IEICE TRANS. FUNDAMENTALS, VOL.E91–A, NO.1 JANUARY 2008

PAPER Special Section on Cryptography and Information Security

Collision Resistance of Double-Block-Length Hash Function against Free-Start Attack

Shoichi HIROSE[†a)], ***Member***

SUMMARY In this article, we discuss the security of double-block-length (DBL) hash functions against the free-start collision attack. We focus on the DBL hash functions composed of compression functions of the form $F(x) = (f(x), f(p(x)))$, where f is a smaller compression function and p is a permutation. We first show, in the random oracle model, that a significantly good upper bound can be obtained on the success probability of the free-start collision attack with sufficient conditions on p and the set of initial values. We also show that a similar upper bound can be obtained in the ideal cipher model if f is composed of a block cipher.

key words: *double-block-length hash function, random oracle model, ideal cipher model, collision resistance, free-start collision attack*

1. Introduction

A cryptographic hash function is a function from the set of inputs of arbitrary length to the set of outputs of fixed length. One of the most important security requirements for it is collision resistance. For a cryptographic hash function, it should be intractable to find a pair of distinct inputs both of which correspond to the same output.

A cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^\ell$ usually consists of a compression function $F : \{0,1\}^{\ell+b} \to \{0,1\}^\ell$ as follows:

1. From an input $M \in \{0,1\}^*$, a binary sequence is generated whose length is a multiple of b. Let $(m_1, m_2, \ldots, m_l)$ be the sequence, where $m_i \in \{0,1\}^b$.
2. For $1 \le i \le l$, $h_i = F(h_{i-1}, m_i)$ is evaluated. $h_0 \in \{0,1\}^\ell$ is an initial value. $h_l = H(M)$.

This type of cryptographic hash function is called an iterated hash function. The first step of the above procedure is called padding. It usually involves Merkle-Damgård strengthening [6], [16], which appends the length of M in the last block m_l. In the second step, h_0 is usually a constant.

There are two kinds of methods to construct a compression function: from scratch and using a cryptographic component. For the latter method, typical examples of components are block ciphers and (smaller) compression functions. Suppose that AES or the compression function of MD5 or SHA-1 are used for construction. Then, taking the birthday attack into consideration, we have to construct a hash function whose output length is twice larger than that of the component. Such a kind of hash function is called a double-block-length (DBL) hash function.

For DBL hash functions, Nandi considered a construction of a compression function F of the form $F(x) = (f(x), f(p(x)))$, where f is a smaller compression function and p is a permutation [17]. f may be composed of a block cipher. He showed that, in the random oracle model, the success probability of the collision attack is $O(q/2^n)$ on some DBL hash functions composed of the compression functions of the form shown above, where n is the output length of f and q is the number of the queries to the oracle. On the other hand, the authors gave an upper bound of $O((q/2^n)^2)$ with a sufficient condition on p in the random oracle or the ideal cipher model [9]. Notice that $(q/2^n)^2 \ll q/2^n$ if $q/2^n \ll 1$.

In this article, we discuss the resistance of DBL hash functions given in [9] against the free-start collision attack in the random oracle or the ideal cipher model. This attack is also referred to as the pseudo-collision attack [15]. It is a probabilistic algorithm whose input and output are defined as follows:

Input The description of a DBL hash function $H : \{0,1\}^* \to \{0,1\}^{2n}$. Let IV be the set of initial values.

Output (h_0, M), (h'_0, M') such that

- $H(h_0, M) = H(h'_0, M')$,
- $(h_0, M) \neq (h'_0, M')$ and $h_0, h'_0 \in IV$.

In the definition of the conventional free-start collision attack, $IV = \{0,1\}^{2n}$. If a hash function is collision-resistant against the free-start attack, then one can feed a part of the input instead of an initial value to the hash function. It improves the efficiency especially for short inputs. One may also, for example, personalize the hash function with an initial value of his/her own choice.

The collision resistance against the free-start attack can be measured by the success probability in this setting. We first show that the success probability of the free-start collision attack is $\Theta(q/2^n)$ in the random oracle or the ideal cipher model, where q is the number of the queries to the oracle. Then, we show that the success probability of the free-start collision attacks is $O((q/2^n)^2)$ with sufficient conditions on IV. The condition is quite simple and $\max_{IV} |IV| = 2^{2n-1}$, which does not necessarily hamper the possible usage mentioned above.

As was shown, for example, in [2] and [5], the proof that a scheme is secure in the random oracle or the ideal cipher model does not imply that its particular implementation is secure. However, it is still useful as a test-bed or

Manuscript received March 27, 2007.
Manuscript revised July 13, 2007.

[†]The author is with the Graduate School of Engineering, University of Fukui, Fukui-shi, 910-8507 Japan.
a) URL: http://digcode2.fuee.fukui-u.ac.jp/~hirose/

DOI: 10.1093/ietfec/e91–a.1.74

a sanity check. In particular, the collision resistance of a hash function composed of a block cipher in the ideal cipher model is significant because a collision-resistant hash function cannot be constructed based solely on a black-box pseudorandom permutation [21].

The organization of this article is as follows. In Sect. 2, we provide the definitions for the following discussions and give a short survey of related works. In Sect. 3, we discuss the collision resistance of DBL hash functions composed of smaller compression functions against the free-start attack. In Sect. 4, we discuss the collision resistance of DBL hash functions composed of block ciphers against the free-start attack. Concluding remarks are given in Sect. 5.

2. Preliminaries

2.1 Iterated Hash Function

An iterated hash function $H : \{0,1\}^* \to \{0,1\}^\ell$ usually consists of a compression function $F : \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$, a padding rule Pad, and an initial value $h_0 \in \{0,1\}^\ell$.

For an input $M \in \{0,1\}^*$, Pad first extends it and produces a sequence m whose length is a multiple of b. Let $m = (m_1, m_2, \ldots, m_l)$, where $m_i \in \{0,1\}^b$. Then, $h_i = F(h_{i-1}, m_i)$ is computed successively for $1 \le i \le l$ and $h_l = H(M)$.

In this article, we assume an unambiguous padding with the Merkle-Damgård strengthening [6], [16], which is denoted by $\mathsf{Pad}_{\mathrm{MDS}}$. Let b' be a constant such that $b' \le b$. For an input M, $\mathsf{Pad}_{\mathrm{MDS}}$ produces $M\|0^t\|\mathrm{len}(M)$. $\|$ represents concatenation. $\mathrm{len}(M) \in \{0,1\}^{b'}$ is the binary representation of the length of M. Thus, we assume that H only accepts M whose length is less than $2^{b'}$. $t \ge 0$ has the minimum value such that the length of $M\|0^t\|\mathrm{len}(M)$ is a multiple of b.

We also assume that an initial value is chosen from a set $IV \subseteq \{0,1\}^\ell$. We use the notation $H(h_0, M)$ to represent the output of H for M and the initial value h_0. We also use the notation $H = (F, \mathsf{Pad}, IV)$ to represent an iterated hash function H composed of a compression function F, a padding rule Pad and a set of initial values IV.

2.2 DBL Hash Function

An iterated hash function whose compression function is composed of a block cipher is called a single-block-length (SBL) hash function if its output length is equal to the block length of the block cipher. It is called a double-block-length (DBL) hash function if its output length is twice larger than the block length.

Let F be a compression function composed of a block cipher. For an iterated hash function composed of F, the rate $r = |m_i|/(\sigma n)$ is often used as a measure of efficiency, where σ is the number of block-cipher calls in F and n is the block length of the block cipher.

In this article, we also call an iterated hash function a DBL hash function if its compression function F is composed of a smaller compression function f and its output length is twice larger than that of f.

2.3 Random Oracle Model and Ideal Cipher Model

2.3.1 Random Oracle Model

Let $\boldsymbol{F}_{n',n} = \{f \mid f : \{0,1\}^{n'} \to \{0,1\}^n\}$. In the random oracle model [1], the function f is assumed to be randomly selected from $\boldsymbol{F}_{n',n}$. The oracle f first receives an input x_i as a query. Then, it returns a randomly selected output y_i if the query has never been asked before. It returns the same reply to the same query.

2.3.2 Ideal Cipher Model

A block cipher with the block length n and the key length κ is called an (n,κ) block cipher. Let $e : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be an (n,κ) block cipher. Then, $e(k,\cdot)$ is a permutation for every $k \in \{0,1\}^\kappa$.

Let $\boldsymbol{B}_{n,\kappa}$ be the set of all (n,κ) block ciphers. In the ideal cipher model, e is assumed to be randomly selected from $\boldsymbol{B}_{n,\kappa}$. The encryption e and the decryption e^{-1} are simulated by the following two oracles.

The encryption oracle e first receives a pair of a key and a plaintext as a query. Then, it returns a randomly selected ciphertext. On the other hand, the decryption oracle e^{-1} first receives a pair of a key and a ciphertext as a query. Then, it returns a randomly selected plaintext. The oracles e and e^{-1} share a table of triplets of keys, plaintexts and ciphertexts, which are produced by the queries and the corresponding replies. Referring to the table, they select a reply to a new query under the restriction that $e(k,\cdot)$ is a permutation for every k. They also add the triplet produced by the query and the reply to the table.

2.4 Related Work

Preneel, Govaerts and Vandewalle [19] discussed the security of SBL hash functions against several generic attacks. They considered SBL hash functions composed of compression functions $h_i = e(k,x) \oplus z$, where e is an (n,n) block cipher, $k, x, z \in \{h_{i-1}, m_i, h_{i-1} \oplus m_i, c\}$ and c is a constant. They concluded that 12 out of 64 $(= 4^3)$ hash functions are secure against the attacks. However, they did not provide any formal proofs.

Black, Rogaway and Shrimpton [3] investigated provable security of SBL hash functions given in [19] in the ideal cipher model. The most important result in their paper is that 20 hash functions including the 12 mentioned above is optimally collision-resistant. We use the term "optimally collision-resistant" to mean that any attack to find a collision is at most as effective as the birthday attack.

Knudsen, Lai and Preneel [12] discussed the insecurity of DBL hash functions with the rate 1 composed of (n,n) block ciphers. Hohl, Lai, Meier and Waldvogel [10] discussed the security of compression functions of DBL hash functions with the rate $1/2$. On the other hand, the security

of DBL hash functions with the rate 1 composed of $(n, 2n)$ block ciphers was discussed by Satoh, Haga and Kurosawa [20] and by Hattori, Hirose and Yoshida [7]. These works presented no construction for DBL hash functions with optimal collision resistance.

Many schemes with the rates less than 1 were also presented. Merkle [16] presented three DBL hash functions composed of DES with the rates at most 0.276. They are optimally collision-resistant in the ideal cipher model. MDC-2 and MDC-4 [4] are also DBL hash functions composed of DES with the rates 1/2 and 1/4, respectively. Lai and Massey proposed the tandem/abreast Davies-Meyer [13]. They consist of an $(n, 2n)$ block cipher and their rates are 1/2. It is an open question whether the four schemes are optimally collision-resistant or not.

Recently, some constructions for DBL hash functions were presented by Hirose [9]. They are similar to the tandem Davies-Meyer but simpler. Moreover, they are optimally collision-resistant. This work is largely motivated by the work by Nandi [17]. Nandi generalized the results by Lucks [14] and by Hirose [8].

Nandi, Lee, Sakurai and Lee [18] proposed an interesting construction with the rate 2/3. However, they are not optimally collision-resistant. Knudsen and Muller [11] presented some attacks on it and illustrated its weaknesses, none of which contradicts the security proof in [18].

3. DBL Hash Function in the Random Oracle Model

3.1 Compression Function

In this section, we consider the DBL hash functions composed of compression functions given in the following definition. Their structure is also depicted in Fig. 1.

Definition 1: Let $F : \{0,1\}^{2n} \times \{0,1\}^b \to \{0,1\}^{2n}$ be a compression function such that $(g_i, h_i) = F(g_{i-1}, h_{i-1}, m_i)$, where $g_i, h_i \in \{0,1\}^n$ and $m_i \in \{0,1\}^b$. F consists of $f : \{0,1\}^{2n} \times \{0,1\}^b \to \{0,1\}^n$ and a permutation $p : \{0,1\}^{2n+b} \to \{0,1\}^{2n+b}$ as follows:

$$\begin{cases} g_i = F_{\mathrm{U}}(g_{i-1}, h_{i-1}, m_i) = f(g_{i-1}, h_{i-1}, m_i) \\ h_i = F_{\mathrm{L}}(g_{i-1}, h_{i-1}, m_i) = f(p(g_{i-1}, h_{i-1}, m_i)). \end{cases}$$

p is represented by $p(g, h, m) = (p_{\mathrm{cv}}(g, h), m)$, where $p_{\mathrm{cv}} : \{0,1\}^{2n} \to \{0,1\}^{2n}$. p_{cv} satisfies the following conditions:

- $p_{\mathrm{cv}}(p_{\mathrm{cv}}(\cdot))$ is the identity permutation,
- p_{cv} has no fixed points, that is, $p_{\mathrm{cv}}(g, h) \neq (g, h)$ for any (g, h), and
- $p_{\mathrm{cv}}(g, h) \neq (h, g)$ for any (g, h).

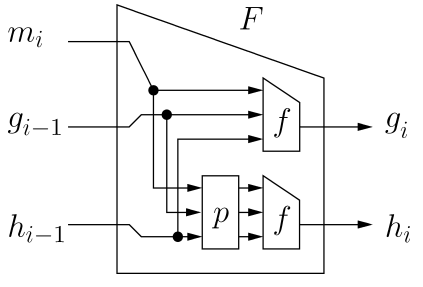

Fig. 1 The compression function in Definition 1.

Example 1: The conditions on p_{cv} in Definition 1 do not necessarily make the compression function F impractical. Here is an example of the permutation p_{cv} satisfying the conditions:

$$p_{\mathrm{cv}}(g, h) = (g \oplus c_1, h \oplus c_2),$$

where c_1 and c_2 are distinct constants in $\{0,1\}^n$.

We will analyze the collision resistance of DBL hash functions composed of F under the assumption that f is a random oracle.

Two queries to the oracle f are required to compute the output of F for an input. For this compression function, a query to f for F_{U} or F_{L} uniquely determines the query to f for the other since p is a permutation. Moreover, for every $w \in \{0,1\}^{2n+b}$, $f(w)$ and $f(p(w))$ are only used to compute $F(w)$ and $F(p(w))$, and $w \neq p(w)$ from the properties of p in Definition 1. Thus, it is reasonable to assume that a pair of queries w and $p(w)$ to f are asked at a time.

Definition 2: A pair of distinct inputs w, w' to F are called a matching pair if $w' = p(w)$. Otherwise, they are called a non-matching pair.

Notice that $w' = p(w)$ iff $w = p(w')$ since $p(p(\cdot))$ is the identity permutation.

3.2 Collision Resistance

3.2.1 Definition

For a set S, let $z \xleftarrow{\mathrm{r}} S$ represent random sampling from S under the uniform distribution. For a probabilistic algorithm $\mathcal{M}$, let $z \xleftarrow{\mathrm{r}} \mathcal{M}$ mean that z is an output of $\mathcal{M}$ and its distribution is based on the random choices of $\mathcal{M}$.

Let $H = (F, \mathsf{Pad}, IV)$ be a DBL hash function, where F is specified in Definition 1. The following experiment $\mathtt{FindColHF}(\mathcal{A}, H)$ is introduced to quantify the collision resistance of H. The adversary $\mathcal{A}$ with the oracle f is a probabilistic collision-finding algorithm for H. $\mathcal{A}$ is for the collision attack if $|IV| = 1$ and for the conventional free-start collision attack if $|IV| = 2^{2n}$.

$\mathtt{FindColHF}(\mathcal{A}, H)$
  $f \xleftarrow{\mathrm{r}} \boldsymbol{F}_{2n+b,n}$;
  $(v_0, M), (v'_0, M') \xleftarrow{\mathrm{r}} \mathcal{A}^f$;
  if $v_0, v'_0 \in IV \wedge (v_0, M) \neq (v'_0, M')$
    $\wedge\, H(v_0, M) = H(v'_0, M')$
    $\wedge\, \mathcal{A}^f$ made all the queries to f necessary to compute both $H(v_0, M)$ and $H(v'_0, M')$
    return 1;
  else return 0;

The collision resistance of H is measured by the probability that $\texttt{FindColHF}(\mathcal{A}, H)$ returns 1 in this setting. Let $\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A})$ be the probability that $\texttt{FindColHF}(\mathcal{A}, H)$ returns 1. It is taken over the uniform distribution on $\boldsymbol{F}_{2n+b,n}$ and random choices of $\mathcal{A}$. It is a function of the number of the queries to f made by $\mathcal{A}$. Without loss of generality, it is assumed that $\mathcal{A}$ does not ask the same query twice. $\mathcal{A}$ can keep pairs of queries and their corresponding answers by himself.

Essentially, we have to consider the time and space complexity of $\mathcal{A}$ and the complexity of the description of H. However, we do not make them explicit in the analysis below. Once f is implemented, for example, by the SHA-1 compression function, it is easy to compute f, n is fixed, and q is the major factor which determines the complexity of $\mathcal{A}$. Moreover, we will see that we can take IV with a simple description as well as p.

3.2.2 Analysis

In this section, we show the collision resistance of hash functions composed of F in Definition 1 against the free-start attack. We first present some lemmas which are used to prove the collision resistance. These lemmas are on the probability that a colliding pair of inputs is found for the compression function F. We say that an algorithm succeeds in finding a colliding pair of inputs w and w' for F only if it made all the queries to f necessary to compute both $F(w)$ and $F(w')$.

Lemma 1: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, \{0,1\}^{2n})$ be a hash function such that $F : \{0,1\}^{2n+b} \to \{0,1\}^{2n}$ is specified in Definition 1. Let $\mathcal{A}$ be a collision-finding algorithm for H with the oracle f. $\mathcal{A}$ asks q pairs of queries to f in total. Then, there exists an algorithm $\mathcal{B}$ with the oracle f which succeeds in finding

1. a colliding pair of non-matching inputs for F, or
2. a colliding pair of matching inputs for F

with the probability $\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A})$. $\mathcal{B}$ asks q pairs of queries to f in total.

Proof: $\mathcal{B}$ first runs $\mathcal{A}$. Suppose that $\mathcal{A}$ finds a colliding pair $((g_0, h_0), M), ((g_0', h_0'), M')$ for H. Then, it is easy to see that $\mathcal{B}$ finds a colliding pair of inputs for F by tracking the computation of H backwards. The colliding pair is either matching or non-matching. During the process, $\mathcal{B}$ needs no other queries than those made by $\mathcal{A}$. □

The following lemmas give upper bounds of the success probabilities of the events listed in Lemma 1.

Lemma 2: Let F be a compression function specified in Definition 1. Let $\mathcal{B}_{\mathrm{c}}$ be any algorithm to find a colliding pair of non-matching inputs for F. Suppose that $\mathcal{B}_{\mathrm{c}}$ asks q pairs of queries to f in total. Then, the success probability of $\mathcal{B}_{\mathrm{c}}$ is at most $q(q-1)/2^{2n}$.

Proof: For $1 \le j \le q$, let w_j and $p(w_j)$ be the j-th pair of queries made by $\mathcal{B}_{\mathrm{c}}$. For $2 \le j \le q$, let C_j be the event that $\mathcal{B}_{\mathrm{c}}$ finds a colliding pair of non-matching inputs for F with the j-th pair of queries. Namely, it is the event that

$$(f(w_j), f(p(w_j))) = (f(w_{j'}), f(p(w_{j'}))) \text{ or } (f(p(w_{j'})), f(w_{j'}))$$

for some $j' < j$. Since both $f(w_j)$ and $f(p(w_j))$ are randomly selected by the oracle,

$$\Pr[\mathsf{C}_j] \le \frac{2(j-1)}{2^{2n}}.$$

Let C be the event that $\mathcal{B}_{\mathrm{c}}$ finds a colliding pair of non-matching inputs. Then,

$$\Pr[\mathsf{C}] = \Pr[\mathsf{C}_2 \vee \mathsf{C}_3 \vee \cdots \vee \mathsf{C}_q] \le \sum_{j=2}^{q} \Pr[\mathsf{C}_j] \le \frac{q(q-1)}{2^{2n}}.$$

□

Lemma 3: Let F be a compression function specified in Definition 1. Let $\mathcal{B}_{\mathrm{c}}'$ be any algorithm to find a colliding pair of matching inputs for F. Suppose that $\mathcal{B}_{\mathrm{c}}'$ asks q pairs of queries to f in total. Then, the success probability of $\mathcal{B}_{\mathrm{c}}'$ is at most $q/2^n$.

Proof: For $1 \le j \le q$, let $\mathsf{C}_j^{\mathrm{m}}$ be the event that $\mathcal{B}_{\mathrm{c}}'$ finds a colliding pair of matching inputs for F with the j-th pair of queries, that is, $f(w_j) = f(p(w_j))$. Thus,

$$\Pr[\mathsf{C}_j^{\mathrm{m}}] = \frac{1}{2^n}.$$

Let C^{m} be the event that $\mathcal{B}_{\mathrm{c}}'$ finds a colliding pair of matching inputs for F. Then,

$$\Pr[\mathsf{C}^{\mathrm{m}}] = \Pr[\mathsf{C}_1^{\mathrm{m}} \vee \mathsf{C}_2^{\mathrm{m}} \vee \cdots \vee \mathsf{C}_q^{\mathrm{m}}] \le \sum_{j=1}^{q} \Pr[\mathsf{C}_j^{\mathrm{m}}] = \frac{q}{2^n}.$$

□

The following theorem is obvious from the above lemmas, and the proof is omitted.

Theorem 1: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, \{0,1\}^{2n})$ be a hash function such that F is a compression function specified in Definition 1. Then, for any collision-finding algorithm $\mathcal{A}$ asking q pairs of queries to f in total,

$$\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A}) \le \frac{q(q-1)}{2^{2n}} + \frac{q}{2^n}.$$

Theorem 1 is valid as long as its upper bound is less than 1. The following theorem shows that the upper bound is almost optimal.

Theorem 2: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, \{0,1\}^{2n})$ be a hash function such that F is a compression function specified in Definition 1. Then, there exists a collision-finding algorithm $\mathcal{A}'$ such that

$$\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A}') \ge \frac{q}{2^n} - \frac{q(q-1)}{2^{2n+1}},$$

where $\mathcal{A}'$ asks at most q pairs of queries to f in total.

Proof: Let $m = \mathsf{Pad}_{\mathrm{MDS}}(\epsilon) \in \{0,1\}^b$, where ϵ is an empty

sequence. Let $\mathcal{A}'$ be a collision-finding algorithm for H whose behaviour is given below.

1. Set $i \leftarrow 1$ and $Q \leftarrow \{0,1\}^{2n}$.
2. While $1 \le i \le q$,
   a. Select $(g,h) \in Q$ at random and ask $((g,h),m)$ and $(p_{\mathrm{cv}}(g,h),m)$ to the oracle f.
   b. If $f((g,h),m) \neq f(p_{\mathrm{cv}}(g,h),m)$, then set $i \leftarrow i+1$ and $Q \leftarrow Q \setminus \{(g,h), p_{\mathrm{cv}}(g,h)\}$, and go to 2.
   c. If $f((g,h),m) = f(p_{\mathrm{cv}}(g,h),m)$, then output $((g,h),m),(p_{\mathrm{cv}}(g,h),m)$ and stop.
3. Output *fail* and stop.

If $f((g,h),m) = f(p_{\mathrm{cv}}(g,h),m)$, then $H((g,h),\epsilon) = H(p_{\mathrm{cv}}(g,h),\epsilon)$. Thus, the pair of inputs produced by $\mathcal{A}'$ is a colliding pair for H. The probability that $\mathcal{A}'$ succeeds in finding a collision is

$$1-\left(1-\frac{1}{2^n}\right)^q \ge \frac{q}{2^n} - \frac{q(q-1)}{2^{2n+1}}.$$

□

We can obtain a better bound than in Theorem 1 if we take the set of initial values appropriately.

Lemma 4: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F is specified in Definition 1, and
- if $(g_0,h_0) \in IV$, then $p_{\mathrm{cv}}(g_0,h_0) \notin IV$.

Let $\mathcal{A}$ be a collision-finding algorithm for H with the oracle f. $\mathcal{A}$ asks q pairs of queries to f in total. Then, there exists an algorithm $\mathcal{B}$ with the oracle f which succeeds in finding

1. a colliding pair of non-matching inputs for F, or
2. a pair of non-matching inputs w and w' for F such that $F(w) = p_{\mathrm{cv}}(F(w'))$

with the probability $\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A})$. $\mathcal{B}$ asks q pairs of queries to f in total.

Proof: $\mathcal{B}$ first runs $\mathcal{A}$. Suppose that $\mathcal{A}$ finds a colliding pair $((g_0,h_0),M)$ and $((g'_0,h'_0),M')$ for H. Let $(m_1,m_2,\ldots,m_l) = \mathsf{Pad}_{\mathrm{MDS}}(M)$ and $(m'_1,m'_2,\ldots,m'_{l'}) = \mathsf{Pad}_{\mathrm{MDS}}(M')$.

Suppose that $|M| \neq |M'|$. Then, (g_{l-1},h_{l-1},m_l) and $(g'_{l'-1},h'_{l'-1},m'_{l'})$ are a colliding pair of non-matching inputs for F since $m_l \neq m'_{l'}$.

On the other hand, suppose that $|M| = |M'|$. Then, it is easy to see that $\mathcal{B}$ also finds a colliding pair of inputs for F by tracking the computation of H backwards. In the remaining part, it is shown that a colliding pair of matching inputs for F is always accompanied by a pair of non-matching inputs w and w' such that $F(w) = p_{\mathrm{cv}}(F(w'))$.

Suppose that $\mathcal{B}$ finds a colliding pair of matching inputs for F by tracking the computation of H. Let (g_{i-1},h_{i-1},m_i) and (g'_{i-1},h'_{i-1},m'_i) be the pair. Then, $(g'_{i-1},h'_{i-1}) = p_{\mathrm{cv}}(g_{i-1},h_{i-1})$. Since $|M| = |M'|$, both of (g_{i-1},h_{i-1}) and (g'_{i-1},h'_{i-1}) are (i) initial values in IV, or (ii) outputs of F. The former case contradicts the assumption on IV that at most only one of (g_{i-1},h_{i-1}) and $p_{\mathrm{cv}}(g_{i-1},h_{i-1})$ is in IV. Thus, $\mathcal{B}$ finds a pair of inputs $(g_{i-2},h_{i-2},m_{i-1})$ and $(g'_{i-2},h'_{i-2},m'_{i-1})$ such that

$$(g_{i-1},h_{i-1}) = F(g_{i-2},h_{i-2},m_{i-1}), \text{ and}$$
$$(g'_{i-1},h'_{i-1}) = F(g'_{i-2},h'_{i-2},m'_{i-1}).$$

This pair is non-matching since $p_{\mathrm{cv}}(g_{i-1},h_{i-1}) = (g'_{i-1},h'_{i-1}) \neq (h_{i-1},g_{i-1})$ from the assumption on p_{cv}.

During the process above, $\mathcal{B}$ needs no other queries than those made by $\mathcal{A}$. □

As in the proof of Theorem 1, the following lemma gives an upper bound of the probability of the second event for F in Lemma 4. Notice that, if the algorithm $\mathcal{B}$ in Lemma 4 succeeds in finding a pair of non-matching inputs w and w' for F such that $F(w) = p_{\mathrm{cv}}(F(w'))$, it made all the queries to f necessary to compute both $F(w)$ and $F(w')$.

Lemma 5: Let F be a compression function specified in Definition 1. Let $\mathcal{B}'_{\mathrm{c}}$ be any algorithm to find a pair of non-matching inputs w and w' for F such that $F(w) = p_{\mathrm{cv}}(F(w'))$. Suppose that $\mathcal{B}'_{\mathrm{c}}$ asks q pairs of queries to f in total and asks all the queries necessary to compute both $F(w)$ and $F(w')$. Then, the success probability of $\mathcal{B}'_{\mathrm{c}}$ is at most $2q(q-1)/2^{2n}$.

Proof: For $2 \le j \le q$, let C'_j be the event that $\mathcal{B}'_{\mathrm{c}}$ finds a pair of non-matching inputs w and w' such that $F(w) = p_{\mathrm{cv}}(F(w'))$ with the j-th pair of queries w_j and $p(w_j)$. Namely, it is the event that

$$F(w_j) = p_{\mathrm{cv}}(F(w_{j'})) \text{ or } p_{\mathrm{cv}}(F(p(w_{j'})))$$

or

$$F(p(w_j)) = p_{\mathrm{cv}}(F(w_{j'})) \text{ or } p_{\mathrm{cv}}(F(p(w_{j'})))$$

for some $j' < j$. Thus,

$$\Pr[\mathsf{C}'_j] \le \frac{4(j-1)}{2^{2n}}.$$

Let C' be the event that $\mathcal{B}'_{\mathrm{c}}$ finds a pair of non-matching inputs w and w' such that $F(w) = p_{\mathrm{cv}}(F(w'))$. Then,

$$\Pr[\mathsf{C}'] \le \sum_{j=2}^{q} \Pr[\mathsf{C}'_j] \le \sum_{j=2}^{q} \frac{4(j-1)}{2^{2n}} = \frac{2q(q-1)}{2^{2n}}.$$

□

Theorem 3: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F is specified in Definition 1, and
- if $(g_0,h_0) \in IV$, then $p_{\mathrm{cv}}(g_0,h_0) \notin IV$.

Then, for any collision-finding algorithm $\mathcal{A}$ asking q pairs of queries to f in total,

$$\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A}) \le \frac{3\,q\,(q-1)}{2^{2n}}.$$

Theorem 3 directly follows from Lemmas 2, 4 and 5. It is valid as long as its upper bound is less than 1.

Remark 1: For $q \le 2^{n-1}$, Theorem 3 gives a smaller upper bound than Theorem 1. Their difference is significant. The upper bound of Theorem 3 is at most $3\,(q/2^n)^2$. On the other hand, the upper bound of Theorem 1 is about $q/2^n$ if $q/2^n \ll 1$. For example, let $n = 128$ and $q = 2^{80}$. Then, the upper bound of Theorem 3 is less than 2^{-94}, while the upper bound of Theorem 1 is about 2^{-48}.

Remark 2: Let $IV \subset \{0,1\}^{2n}$ be a set of initial values satisfying the condition given in Theorem 3. Since $p_{\mathrm{cv}}(p_{\mathrm{cv}}(\cdot))$ is the identity permutation and p_{cv} has no fixed points, $|IV| \le 2^{2n-1}$. We can take, for example,

$$\{(g,h)\,|\,(g,h) \in \{0,1\}^{2n} \wedge (g,h) \prec p_{\mathrm{cv}}(g,h)\}$$

as IV, where $\prec$ represents a lexicographical order with $0 \prec 1$. In this case, $|IV| = 2^{2n-1}$.

Let p_{cv} be the permutation in Example 1, and let $c_1 = 10^{n-1}$ and $c_2 = 0^n$. Then, we can have

$$IV = \{(g,h)\,|\,(g,h) \in \{0,1\}^{2n} \wedge \mathrm{msb}(g) = 0\},$$

where $\mathrm{msb}(g)$ is the most significant bit of g. Thus, we can choose the other $(2n-1)$ bits of the initial value arbitrarily.

4. DBL Hash Function in the Ideal Cipher Model

4.1 Compression Function

In this section, the collision resistance of a DBL hash function composed of a compression function using a block cipher is analyzed. The compression function specified in the following definition is considered.

Definition 3: Let $F : \{0,1\}^{2n} \times \{0,1\}^b \to \{0,1\}^{2n}$ be a compression function such that $(g_i, h_i) = F(g_{i-1}, h_{i-1}, m_i)$, where $g_i, h_i \in \{0,1\}^n$ and $m_i \in \{0,1\}^b$. F consists of an $(n, n+b)$ block cipher e as follows:

$$\begin{aligned} g_i &= F_{\mathrm{U}}(g_{i-1}, h_{i-1}, m_i) = e(h_{i-1}\|m_i, g_{i-1}) \oplus g_{i-1} \\ h_i &= F_{\mathrm{L}}(g_{i-1}, h_{i-1}, m_i) \\ &= e(h_{i-1}\|m_i, g_{i-1} \oplus c) \oplus g_{i-1} \oplus c, \end{aligned}$$

where $c \in \{0,1\}^n \setminus \{0^n\}$ is a constant.

The compression function in Definition 3 is also shown in Fig. 2. It can be regarded as a variant of the compression function specified in Definition 1, where f and p are specified as follows:

$$\begin{aligned} f(g_{i-1}, h_{i-1}, m_i) &= e(h_{i-1}\|m_i, g_{i-1}) \oplus g_{i-1}, \\ p(g_{i-1}, h_{i-1}, m_i) &= (g_{i-1} \oplus c, h_{i-1}, m_i). \end{aligned}$$

F requires two invocations of e to produce an output. However, these two invocations need only one key scheduling of e. If F is implemented using AES with 192-bit key-length, then $n = 128$, $b = 64$ and the rate is 1/4. If implemented using AES with 256-bit key-length, then $n = b = 128$ and the rate is 1/2.

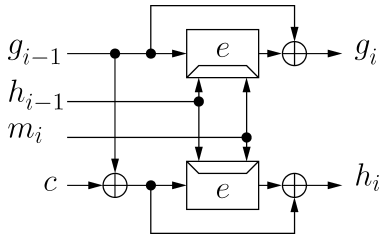

Fig. 2 The compression function in Definition 3.

Two queries to the oracles e and e^{-1} in total are required to compute the output of F for an input. It is apparent from Fig. 2 that a query to e or e^{-1} and the corresponding reply for F_{U} (F_{L}) uniquely determine the query to e for F_{L} (F_{U}). Moreover, these two queries are only used to compute the outputs of F for a matching pair of inputs. Thus, it is assumed that a pair of queries to e, e^{-1} required to compute an output of F are asked at a time.

4.2 Collision Resistance

4.2.1 Definition

The following experiment $\mathtt{FindColHF}(\mathcal{A}, H)$ is similar to the one in Sect. 3 except that the adversary $\mathcal{A}$ is a collision-finding algorithm with the oracles e, e^{-1}.

```
FindColHF(A, H)
  e ←r B_{n,n+b};
  (v_0, M), (v'_0, M') ←r A^{e,e^{-1}};
  if v_0, v'_0 ∈ IV ∧ (v_0, M) ≠ (v'_0, M')
      ∧ H(v_0, M) = H(v'_0, M')
      ∧ A^{e,e^{-1}} made all the queries to e, e^{-1}
          necessary to compute both H(v_0, M)
          and H(v'_0, M')
    return 1;
  else return 0;
```

Let $\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A})$ be the probability that $\mathtt{FindColHF}(\mathcal{A}, H)$ returns 1. It is taken over the uniform distribution on $\boldsymbol{B}_{n,n+b}$ and random choices of $\mathcal{A}$. Without loss of generality, it is assumed that $\mathcal{A}$ asks at most only once on a triplet of a key, a plaintext and a ciphertext obtained by a query and the corresponding reply.

4.2.2 Analysis

In this part, we show the collision resistance of a hash function composed of F in Definition 3 against the free-start attack.

Lemma 6: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F is specified in Definition 3, and
- if $(g_0, h_0) \in IV$, then $(g_0 \oplus c, h_0) \notin IV$.

Let $\mathcal{A}$ be a collision-finding algorithm for H, which asks q

pairs of queries to e, e^{-1} in total. Then, there exists an algorithm $\mathcal{B}$ with the oracles e, e^{-1} which succeeds in finding

1. a colliding pair of non-matching inputs for F, or
2. a pair of non-matching inputs w and w' for F such that $F(w) = (F_{\mathrm{U}}(w') \oplus c, F_{\mathrm{L}}(w'))$

with the probability $\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A})$. $\mathcal{B}$ asks q pairs of queries to e, e^{-1} in total.

The proof of Lemma 6 is omitted since it is similar to that of Lemma 4. For the events listed in Lemma 6, upper bounds of their success probabilities are given in the following lemmas. We say that an algorithm succeeds only if it made all the queries to e, e^{-1} necessary to compute the outputs of F for the inputs produced by it.

Lemma 7: Let F be the compression function specified in Definition 3. Let $\mathcal{B}_{\mathrm{c}}$ be any algorithm to find a colliding pair of non-matching inputs for F. Suppose that $\mathcal{B}_{\mathrm{c}}$ asks q pairs of queries to e, e^{-1} in total. Then, for every $1 \le q \le 2^{n-2}$, the success probability of $\mathcal{B}_{\mathrm{c}}$ is at most $q(q-1)/2^{2(n-1)}$.

Proof: For $1 \le j \le q$, let $(k_j^1 \| k_j^2, x_j, y_j)$ and $(k_j^1 \| k_j^2, x_j \oplus c, z_j)$ represent the triplets of e obtained by the j-th pair of queries and the corresponding answers.

For $2 \le j \le q$, let C_j be the event that $\mathcal{B}_{\mathrm{c}}$ finds a colliding pair of non-matching inputs for F with the j-th pair of queries. Namely, it is the event that, for some $j' < j$,

$$F(x_j, k_j^1, k_j^2) = F(x_{j'}, k_{j'}^1, k_{j'}^2) \text{ or } F(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2)$$

or

$$\begin{aligned} &F(x_j \oplus c, k_j^1, k_j^2) \\ &\quad = F(x_{j'}, k_{j'}^1, k_{j'}^2) \text{ or } F(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2), \end{aligned}$$

which is equivalent to

$$\begin{aligned} &(y_j \oplus x_j, z_j \oplus x_j \oplus c) \\ &\qquad = (y_{j'} \oplus x_{j'}, z_{j'} \oplus x_{j'} \oplus c) \text{ or} \\ &\qquad\quad (z_{j'} \oplus x_{j'} \oplus c, y_{j'} \oplus x_{j'}). \end{aligned}$$

Thus,

$$\begin{aligned} \Pr[\mathsf{C}_j] &\le \frac{2(j-1)}{(2^n - (2j-2))(2^n - (2j-1))} \\ &\le \frac{2(j-1)}{(2^n - (2j-1))^2}. \end{aligned}$$

Let C be the event that $\mathcal{B}_{\mathrm{c}}$ finds a colliding pair of non-matching inputs for F. Then, for $1 \le q \le 2^{n-2}$,

$$\begin{aligned} \Pr[\mathsf{C}] &\le \sum_{j=2}^{q} \Pr[\mathsf{C}_j] \le \sum_{j=2}^{q} \frac{2(j-1)}{(2^n - (2j-1))^2} \\ &\le \sum_{j=2}^{q} \frac{2(j-1)}{2^{2(n-1)}} \le \frac{q(q-1)}{2^{2(n-1)}}. \end{aligned}$$

□

Lemma 8: Let F be the compression function specified in Definition 3. Let $\mathcal{B}'_{\mathrm{c}}$ be any algorithm to find a pair of non-matching inputs w and w' for F such that $F(w) = (F_{\mathrm{U}}(w') \oplus c, F_{\mathrm{L}}(w'))$. Suppose that $\mathcal{B}'_{\mathrm{c}}$ asks q pairs of queries to e, e^{-1} in total. Then, for every $1 \le q \le 2^{n-2}$, the success probability of $\mathcal{B}'_{\mathrm{c}}$ is at most $2q(q-1)/2^{2(n-1)}$.

Proof: Let C'_j be the event that $\mathcal{B}'_{\mathrm{c}}$ finds a pair of non-matching inputs for F such as given above with the j-th pair of queries. Namely, it is the event that, for some $j' < j$,

$$\begin{aligned} &F(x_j, k_j^1, k_j^2) \\ &\quad = (F_{\mathrm{U}}(x_{j'}, k_{j'}^1, k_{j'}^2) \oplus c, F_{\mathrm{L}}(x_{j'}, k_{j'}^1, k_{j'}^2)) \text{ or} \\ &\qquad (F_{\mathrm{U}}(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2) \oplus c, F_{\mathrm{L}}(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2)), \end{aligned}$$

or

$$\begin{aligned} &F(x_j \oplus c, k_j^1, k_j^2) \\ &\quad = (F_{\mathrm{U}}(x_{j'}, k_{j'}^1, k_{j'}^2) \oplus c, F_{\mathrm{L}}(x_{j'}, k_{j'}^1, k_{j'}^2)) \text{ or} \\ &\qquad (F_{\mathrm{U}}(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2) \oplus c, F_{\mathrm{L}}(x_{j'} \oplus c, k_{j'}^1, k_{j'}^2)). \end{aligned}$$

It is equivalent to

$$\begin{aligned} &(y_j \oplus x_j, z_j \oplus x_j \oplus c) \\ &\quad = (y_{j'} \oplus x_{j'} \oplus c, z_{j'} \oplus x_{j'} \oplus c), \\ &\qquad (z_{j'} \oplus x_{j'}, y_{j'} \oplus x_{j'}), \\ &\qquad (z_{j'} \oplus x_{j'} \oplus c, y_{j'} \oplus x_{j'} \oplus c) \text{ or} \\ &\qquad (y_{j'} \oplus x_{j'}, z_{j'} \oplus x_{j'}). \end{aligned}$$

Thus,

$$\Pr[\mathsf{C}'_j] \le \frac{4(j-1)}{(2^n - (2j-1))^2}.$$

Let C' be the event that $\mathcal{B}'_{\mathrm{c}}$ finds a pair of non-matching inputs w and w' for F such that $F(w) = (F_{\mathrm{U}}(w') \oplus c, F_{\mathrm{L}}(w'))$. Then, for $1 \le q \le 2^{n-2}$,

$$\Pr[\mathsf{C}'] \le \sum_{j=1}^{q} \Pr[\mathsf{C}'_j] \le \frac{2q(q-1)}{2^{2(n-1)}}.$$

□

The following theorem is obvious from Lemmas 6, 7 and 8.

Theorem 4: Let $H = (F, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F is specified in Definition 3, and
- if $(g_0, h_0) \in IV$, then $(g_0 \oplus c, h_0) \notin IV$.

Then, for any $1 \le q \le 2^{n-2}$ and any collision-finding algorithm $\mathcal{A}$ asking q pairs of queries to f in total,

$$\mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A}) \le \frac{3q(q-1)}{2^{2(n-1)}}.$$

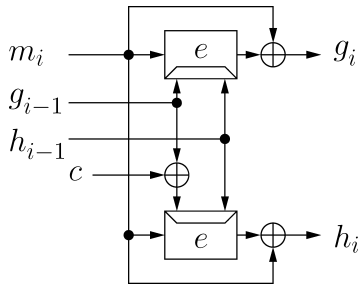

Fig. 3 The compression function F_1.

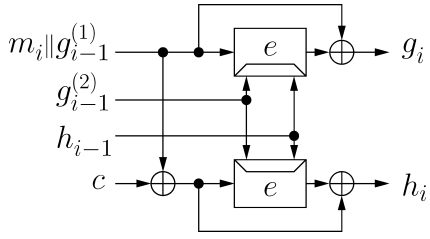

Fig. 4 The compression function F_2.

4.3 Other Schemes

The compression function F given in Definition 3 can be composed of an $(n, 2n)$ or $(n, 3n/2)$ block cipher. The possible drawback of F is that the message block m_i is given to the block cipher as a part of the key. Thus, the key input can be chosen arbitrarily by adversaries.

The following compression function can be composed of an $(n, 2n)$ block cipher. The message block m_i is fed into the block cipher as a plaintext. It is given in Fig. 3. It requires two invocations of key scheduling.

Definition 4: Let $F_1 : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a compression function $(g_i, h_i) = F_1(g_{i-1}, h_{i-1}, m_i)$ such that

$$g_i = e(g_{i-1} \| h_{i-1}, m_i) \oplus m_i$$
$$h_i = e((g_{i-1} \oplus c) \| h_{i-1}, m_i) \oplus m_i,$$

where $g_i, h_i, m_i \in \{0,1\}^n$ and $c \in \{0,1\}^n \setminus \{0^n\}$ is a constant.

The other example of a compression function can be composed of an $(n, 3n/2)$ block cipher. It is depicted in Fig. 4. It requires one invocation of key scheduling. The message block is fed into the block cipher as a part of the plaintext.

Definition 5: Let $F_2 : \{0,1\}^{2n} \times \{0,1\}^{n/2} \to \{0,1\}^{2n}$ be a compression function $(g_i, h_i) = F_2(g_{i-1}, h_{i-1}, m_i)$ such that

$$g_i = e(g_{i-1}^{(2)} \| h_{i-1}, m_i \| g_{i-1}^{(1)}) \oplus (m_i \| g_{i-1}^{(1)})$$
$$h_i = e(g_{i-1}^{(2)} \| h_{i-1}, (m_i \| g_{i-1}^{(1)}) \oplus c) \oplus (m_i \| g_{i-1}^{(1)}) \oplus c,$$

where $g_i, h_i \in \{0,1\}^n$, $g_i^{(1)}, g_i^{(2)}, m_i \in \{0,1\}^{n/2}$, $g_i = g_i^{(1)} \| g_i^{(2)}$, and $c = 0^{n/2} \| c_g \in \{0,1\}^n$ is a constant such that $c_g \in \{0,1\}^{n/2} \setminus \{0^{n/2}\}$.

For the hash functions composed of F_1 or F_2, we can obtain the same upper bound on the adversarial advantage for collision resistance as for F in Definition 3. The proofs are omitted since they are similar to that of Theorem 4.

Theorem 5: Let $H_1 = (F_1, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F_1 is specified in Definition 4, and
- if $(g_0, h_0) \in IV$, then $(g_0 \oplus c, h_0) \notin IV$.

Then, for any $1 \le q \le 2^{n-1}$ and any collision-finding algorithm $\mathcal{A}$ asking q pairs of queries to f in total,

$$\mathbf{Adv}_{H_1}^{\mathrm{coll}}(\mathcal{A}) \le \frac{3\,q\,(q-1)}{2^{2(n-1)}}.$$

Theorem 6: Let $H_2 = (F_2, \mathsf{Pad}_{\mathrm{MDS}}, IV)$ be a hash function such that

- F_2 is specified in Definition 5, and
- if $(g_0, h_0) \in IV$, then $(g_0 \oplus (c_g \| 0^{n/2}), h_0) \notin IV$.

Then, for any $1 \le q \le 2^{n-2}$ and any collision-finding algorithm $\mathcal{A}$ asking q pairs of queries to f in total,

$$\mathbf{Adv}_{H_2}^{\mathrm{coll}}(\mathcal{A}) \le \frac{3\,q\,(q-1)}{2^{2(n-1)}}.$$

Theorem 5 is valid as long as its upper bound is less than 1. It is valid for larger q than Theorems 4 and 6 since F_1 requires two invocations of key scheduling.

5. Concluding Remark

In this article, we have discussed the collision resistance of DBL hash functions against the free-start attack. We have considered DBL hash functions composed of compression functions of the form $F(x) = (f(x), f(p(x)))$. We have given significantly better upper bounds on the success probabilities of the free-start attack with the sufficient conditions on the permutation p and the set of initial values.

Acknowledgements

We would like to thank the anonymous reviewers for their comprehensive comments. This work was supported in part by International Communications Foundation (ICF).

References

[1] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," Proc. 1st ACM Conference on Computer and Communications Security, pp.62–73, 1993.
[2] J. Black, "The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function," Proc. 13th Fast Software Encryption Workshop (FSE 2006), LNCS 4047, pp.328–340, 2006. Also available at "Cryptology ePrint Archive: Report 2005/210," at http://eprint.iacr.org/
[3] J. Black, P. Rogaway, and T. Shrimpton, "Black-box analysis of the block-cipher-based hash-function constructions from PGV," CRYPTO 2002 Proc., LNCS 2442, pp.320–335, 2002.

[4] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, Jr., C.H.W. Meyer, J. Oseas, S. Pilpel, and M. Schilling, "Data authentication using modification detection codes based on a public one-way encryption function," U.S. Patent # 4,908,861, March 1990.
[5] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," J. ACM, vol.51, no.4, pp.557–594, 2004.
[6] I. Damgård, "A design principle for hash functions," CRYPTO'89 Proc., LNCS 435, pp.416–427, 1990.
[7] M. Hattori, S. Hirose, and S. Yoshida, "Analysis of double block length hash functions," Proc. 9th IMA International Conference on Cryptography and Coding, LNCS 2898, pp.290–302, 2003.
[8] S. Hirose, "Provably secure double-block-length hash functions in a black-box model," Proc. 7th Internatinal Conference on Information Security and Cryptology (ICISC 2004), LNCS 3506, pp.330–342, 2005.
[9] S. Hirose, "Some plausible constructions of double-block-length hash functions," Proc. 13th Fast Software Encryption Workshop (FSE 2006), LNCS 4047, pp.210–225, 2006.
[10] W. Hohl, X. Lai, T. Meier, and C. Waldvogel, "Security of iterated hash functions based on block ciphers," CRYPTO'93 Proc., LNCS 773, pp.379–390, 1994.
[11] L. Knudsen and F. Muller, "Some attacks against a double length hash proposal," ASIACRYPT 2005 Proc., LNCS 3788, pp.462–473, 2005.
[12] L.R. Knudsen, X. Lai, and B. Preneel, "Attacks on fast double block length hash functions," J. Cryptol., vol.11, no.1, pp.59–72, 1998.
[13] X. Lai and J.L. Massey, "Hash function based on block ciphers," EUROCRYPT'92 Proc., LNCS 658, pp.55–70, 1993.
[14] S. Lucks, "A failure-friendly design principle for hash functions," ASIACRYPT 2005 Proc., LNCS 3788, pp.474–494, 2005.
[15] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
[16] R.C. Merkle, "One way hash functions and DES," CRYPTO'89 Proc., LNCS 435, pp.428–446, 1990.
[17] M. Nandi, "Towards optimal double-length hash functions," Proc. 6th International Conference on Cryptology in India (INDOCRYPT 2005), LNCS 3797, pp.77–89, 2005.
[18] M. Nandi, W. Lee, K. Sakurai, and S. Lee, "Security analysis of a 2/3-rate double length compression function in the black-box model," Proc. 12th Fast Software Encryption (FSE 2005), LNCS 3557, pp.243–254, 2005.
[19] B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," CRYPTO'93 Proc., LNCS 773, pp.368–378, 1994.
[20] T. Satoh, M. Haga, and K. Kurosawa, "Towards secure and fast hash functions," IEICE Trans. Fundamentals, vol.E82-A, no.1, pp.55–62, Jan. 1999.
[21] D.R. Simon, "Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?," EUROCRYPT'98 Proc., LNCS 1403, pp.334–345, 1998.

Shoichi Hirose received the B.E., M.E. and D.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990 and 1995, respectively. From 1990 to 1998, he was a research associate at Faculty of Engineering, Kyoto University. From 1998 to 2005, he was a lecturer at Graduate School of Informatics, Kyoto University. From 2005, he is an associate professor at Faculty of Engineering, University of Fukui. His current interests include cryptography and information security. He received Young Engineer Award from IEICE in 1997. He is a member of IACR, ACM, IEEE and IPSJ.