PAPER *Special Section on Discrete Mathematics and Its Applications*

A Note on the Strength of Weak Collision Resistance*

Shoichi HIROSE[†a)], ***Member***

SUMMARY NMAC is a function for message authentication based on cryptographic hash functions such as SHA. It is shown to be a secure message authentication code if its compression function with fixed input length is a secure message authentication code and its iterated hash function with variable input length constructed with the compression function is weakly collision resistant. In this article, two results are shown on the strength of the weak collision resistance of the iterated hash function in NMAC. First, it is shown that the weak collision resistance of the iterated hash function in NMAC is not implied by the pseudorandomness of its compression function even if the MD-strengthening is assumed. Second, the weak collision resistance of the iterated hash function in NMAC implies the collision resistance of its compression function if the compression function is pseudorandom.

key words: *hash function, weak collision resistance, message authentication code, NMAC*

1. Introduction

NMAC [2] is a function for message authentication based on cryptographic hash functions such as SHA. It is a theoretical basis of HMAC [2], [4], which is used in practice. NMAC is shown to be a secure message authentication code if its compression function with fixed input length is a secure message authentication code and its iterated hash function with variable input length constructed with the compression function is weakly collision resistant. In this article, two results are shown on the weak collision resistance of the iterated hash function in NMAC.

Related Work. A function for message authentication should be a function with variable input length for practical use and such a function is composed by iterated applications of a function with fixed input length. In this case, it is preferable that the security of the function with variable input length is implied by as weak assumptions as possible on the security of the function with fixed input length.

Bellare, Kilian and Rogaway [5] showed that the CBC-MAC is a secure message authentication code (MAC) if the function with fixed input length used in the construction is pseudorandom. The pseudorandomness is, however, a stronger notion than the MAC security. An and Bellare [1] presented a scheme to construct a secure MAC function with variable input length from a secure MAC function with fixed input length. They call this scheme the NI construction. To prove that the NI construction is a secure MAC, they showed that the weak collision resistance of the iterated hash function in the NI construction is implied by the weak collision resistance of the compression function with fixed input length used in this construction. They also showed that the weak collision resistance is implied by the MAC security with fixed input length.

For other security notions such as pseudorandomness and collision resistance, following results are known. A pseudorandom function with variable input length can be produced by cascade construction of a pseudorandom function with fixed input length [3]. This construction needs a secret key to be appended to inputs or some prefix-free encoding of inputs. Collision resistance for variable input length is implied by collision resistance for fixed input length [7].

Our Contribution. In the iterated hash function in the NI construction, each compression function has the same secret key as a part of the input. On the other hand, in the iterated hash function in NMAC, only the initial value is the secret key, that is, only the first compression function has the secret key as a part of the input. It is an open question how strong the assumption is that the iterated hash function in NMAC is weakly collision resistant. In this article, two results are presented on this question. These results show that there may exist a large gap between the weak collision resistance of the iterated hash function in the NI construction and that of the iterated hash function in NMAC.

First, it is shown that the weak collision resistance of the iterated hash function in NMAC is not implied by the pseudorandomness of the compression function used in the iteration even if the MD-strengthening is used for padding. This result implies that the weak collision resistance of the iterated hash function in NMAC is a stronger notion of security than the weak collision resistance of a function with fixed input length. This is because the latter security notion is implied by the pseudorandomness. Second, it is shown that the weak collision resistance of the iterated hash function in NMAC implies the collision resistance of the compression function used in the iteration if this compression function is pseudorandom. Together with Simon's result [8], this result implies that there may exist a large gap between the weak collision resistance of the iterated hash function in NMAC and that of a function with fixed input length. Si-

Manuscript received August 22, 2003.
Manuscript revised November 6, 2003.
Final manuscript received December 26, 2003.

[†] The author is with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.

a) E-mail: hirose@i.kyoto-u.ac.jp

* This work is supported in part by Grant-in-Aid for Young Scientists (B) KAKENHI 14780209 of Japan Society for the Promotion of Science (JSPS).

mon showed that no provable construction of a collision resistant hash function can exist based solely on a "black box" one-way permutation, namely, a one-way permutation as an oracle. On the other hand, it is easy to see that a weakly collision resistant function with fixed input length can be constructed from a "black box" one-way permutation.

The remainder of this article is organized as follows. Definitions and some notations are presented in Sect. 2. Two results are shown in Sect. 3 on the strength of the weak collision resistance of the iterated hash function in NMAC. Sect. 4 is the concluding remark.

2. Preliminaries

2.1 Definitions

Let F be a function such that $F : \mathfrak{K}(F) \times \mathfrak{D}(F) \to \mathfrak{R}(F)$, where $\mathfrak{K}(F)$, $\mathfrak{D}(F)$, $\mathfrak{R}(F)$ are the set of the keys, that of the inputs and that of the outputs of F, respectively. For $k \in \mathfrak{K}(F)$, let $F_k(\cdot)$ represent $F(k, \cdot)$. F can also be regarded as a family of functions $\{F_k \,|\, k \in \mathfrak{K}(F)\}$.

Four notions of security of a function are defined below. They are collision resistance, weak collision resistance, pseudorandomness and a secure message authentication code. They are defined in the similar way as in [1].

(1) Collision Resistance (CR).

Collision resistance (CR) is defined for a family of functions F as in [6], [7]. To define the CR of F, the following experiment FindCol$(\mathcal{A}, F)$ is introduced, where $\mathcal{A}$ is a probabilistic algorithm which gets a key $k \in \mathfrak{K}(F)$ as an input and works as a collision finder of F_k.

```
FindCol($\mathcal{A}, F$)
    $k \leftarrow \mathfrak{K}(F); (m, m') \leftarrow \mathcal{A}(k)$;
    if m $\neq m' \wedge F_k(m) = F_k(m')$ then return 1;
    else return 0;
```

In the above description of FindCol$(\mathcal{A}, F)$, $k \leftarrow \mathfrak{K}(F)$ means that k is randomly selected from the set $\mathfrak{K}(F)$ and the distribution is uniform. On the other hand, $(m, m') \leftarrow \mathcal{A}(k)$ means that (m, m') is an output of the probabilistic algorithm $\mathcal{A}$ with an input k. The distribution of the output is based on the random choices of $\mathcal{A}$ and the distribution of the input to $\mathcal{A}$. FindCol$(\mathcal{A}, F)$ returns 1 iff $\mathcal{A}(k)$ finds a collision of F_k, that is, a pair of different inputs of F_k which give the same output. Let $\mathbf{Succ}_F^{\mathrm{CR}}(\mathcal{A})$ denote the probability that FindCol$(\mathcal{A}, F)$ returns 1.

The CR of F is quantified by the maximum probability that any collision finder with at most t steps succeeds in finding a collision of F. This value is denoted by $\mathbf{Insec}_F^{\mathrm{CR}}(t)$ and is defined as follows.

$$\mathbf{Insec}_F^{\mathrm{CR}}(t) \overset{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathbf{Succ}_F^{\mathrm{CR}}(\mathcal{A})\},$$

where the number of the steps of $\mathcal{A}$ is at most t.

(2) Weak Collision Resistance (WCR).

The following experiment FindWeakCol$(\mathcal{A}, F)$ is introduced to define the weak collision resistance (WCR) of a function F, where $\mathcal{A}$ is a probabilistic algorithm which takes F_k as an oracle and works as a collision finder of F_k. $\mathcal{A}$ makes a chosen message attack to F_k. The difference between FindWeakCol$(\mathcal{A}, F)$ and FindCol$(\mathcal{A}, F)$ is that the key k of F is not given to $\mathcal{A}$ in FindWeakCol$(\mathcal{A}, F)$. Thus, it is obvious that the WCR of F is implied by the CR of F.

```
FindWeakCol($\mathcal{A}, F$)
    $k \leftarrow \mathfrak{K}(F); (m, m') \leftarrow \mathcal{A}^{F_k}$;
    if m $\neq m' \wedge F_k(m) = F_k(m')$ then return 1;
    else return 0;
```

FindWeakCol$(\mathcal{A}, F)$ returns 1 iff $\mathcal{A}^{F_k}$ finds a pair of different inputs of F_k which give the same output. Let $\mathbf{Succ}_F^{\mathrm{WCR}}(\mathcal{A})$ denote the probability that FindCol$(\mathcal{A}, F)$ returns 1. The WCR of F is quantified by the maximum probability that any collision finder such that the number of its steps, that of its queries and the total length of its queries are at most t, q and μ respectively succeeds in finding a collision of F_k. This value is denoted by $\mathbf{Insec}_F^{\mathrm{WCR}}(t, q, \mu)$ and is defined as follows.

$$\mathbf{Insec}_F^{\mathrm{WCR}}(t, q, \mu) \overset{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathbf{Succ}_F^{\mathrm{WCR}}(\mathcal{A})\},$$

where $\mathcal{A}$ is a collision finder with an oracle F_k such that the number of its steps, that of its queries and the total length of the queries are at most t, q and μ, respectively. For simplicity, a collision finder with an oracle F_k is called a weak collision finder of F in the followings.

(3) Pseudorandomness (PR).

The following experiment Distinguish$(\mathcal{A}, F)$ is introduced to define the pseudorandomness of a function F. $\mathcal{A}$ is a probabilistic algorithm which works as a distinguisher of F with an oracle O, to which $\mathcal{A}$ makes a chosen message attack. In this experiment, a randomly chosen bit $s \in \{0, 1\}$ is given to the oracle first. If $s = 1$, then O chooses $k \in \mathfrak{K}(F)$ randomly in advance. O returns $F_k(x)$ to each query x from $\mathcal{A}$. If $s = 0$, then O chooses a function $R : \mathfrak{D}(F) \to \mathfrak{R}(F)$ randomly in advance. O returns $R(x)$ to each query x from $\mathcal{A}$.

```
Distinguish($\mathcal{A}, F$)
    $s \leftarrow \{0, 1\}; s' \leftarrow \mathcal{A}^{O(s)}$;
    if $s = s'$ then return 1; else return 0;
```

Distinguish$(\mathcal{A}, F)$ returns 1 iff $s = s'$, that is, $\mathcal{A}$ judges correctly which one of F_k and R is used by O to compute the answers to the queries of $\mathcal{A}$. Let $\mathbf{Succ}_F^{\mathrm{PR}}(\mathcal{A})$ be the probability that Distinguish$(\mathcal{A}, F)$ returns 1. We only consider the case where $\mathbf{Succ}_F^{\mathrm{PR}}(\mathcal{A}) \geq 1/2$ because the probability that $s = s'$ is 1/2 even if $\mathcal{A}$ chooses s' randomly. Let

$$\mathbf{Adv}_F^{\mathrm{PR}}(\mathcal{A}) \overset{\mathrm{def}}{=} \mathbf{Succ}_F^{\mathrm{PR}}(\mathcal{A}) - \frac{1}{2}.$$

Then, the pseudorandomness of F is quantified by the maximum of $\mathbf{Adv}_F^{\mathrm{PR}}(\mathcal{A})$ for any $\mathcal{A}$ such that the number of its steps, that of its queries and the total length of its queries are at most t, q and μ, respectively. This value is denoted by $\mathbf{Insec}_F^{\mathrm{PR}}(t,q,\mu)$ and is defined as follows.

$$\mathbf{Insec}_F^{\mathrm{PR}}(t,q,\mu) \overset{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathbf{Adv}_F^{\mathrm{PR}}(\mathcal{A})\}.$$

(4) Message Authentication Code (MAC).

The following experiment $\mathtt{Forge}(\mathcal{A},F)$ is introduced to define the notion that a function F is a secure message authentication code (MAC). $\mathcal{A}$ is a probabilistic algorithm which works as a forger of F_k. $\mathcal{A}$ takes F_k as an oracle and makes a chosen message attack to it.

```
Forge($\mathcal{A}$, $F$)
    $k \leftarrow \Re(F); (m,a) \leftarrow \mathcal{A}^{F_k}$;
    if $a = F_k(m)$ then return 1;
    else return 0;
```

$\mathtt{Forge}(\mathcal{A},F)$ returns 1 iff $\mathcal{A}$ succeeds in forging a pair $(m, F_k(m))$, where m is not included in the queries of $\mathcal{A}$ to F_k. Let $\mathbf{Succ}_F^{\mathrm{MAC}}(\mathcal{A})$ denote the probability that $\mathtt{Forge}(\mathcal{A},F)$ returns 1.

The MAC security of F is quantified by the maximum of $\mathbf{Succ}_F^{\mathrm{MAC}}(\mathcal{A})$ of any forger $\mathcal{A}$ such that the number of its steps, that of its queries and the total length of its queries are at most t, q and μ, respectively. This value is denoted by $\mathbf{Insec}_F^{\mathrm{MAC}}(t,q,\mu)$ and is defined as follows.

$$\mathbf{Insec}_F^{\mathrm{MAC}}(t,q,\mu) \overset{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathbf{Succ}_F^{\mathrm{MAC}}(\mathcal{A})\}.$$

(5) Notations.

The Hamming distance between two binary strings x, y with equal length is denoted by $d_{\mathrm{H}}(x,y)$. The length of a binary string x is denoted by $|x|$. The number of the steps to compute a function f is denoted by $T(f)$.

2.2 Hash Function

A hash function is a function which outputs a string of fixed length for a given input string of arbitrary length. A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ is computed by iterated applications of a compression function $f : \{0,1\}^\ell \times \{0,1\}^b \rightarrow \{0,1\}^\ell$ to a given input x.

A hash function H consists of a compression function f, an initial value $v_0 \in \{0,1\}^\ell$, and a padding algorithm Pad. Pad produces $x_1, x_2, \ldots, x_m$ for a given input x, where $x_i \in \{0,1\}^b$, $|x| \le mb$. For $i = 1,\ldots,m$, $v_i = f(v_{i-1}, x_i)$ and $H(x) = v_m$. This kind of hash function is called an iterated hash function.

The padding algorithm Pad is often based on the following algorithm called MD-strengthening. In this algorithm, a given input x is divided into blocks $x_1, x_2, \ldots, x_n$, each of whose length is b, and a new block $x_{n+1} = |x|_{\mathrm{bin}}$ is added, where $(n-1)b < |x| \le n\,b$ and $|x|_{\mathrm{bin}}$ is a b-bit binary representation of $|x|$. If $|x_n| < b$, then $(b - |x_n|)$ 0's are appended to x_n.

2.3 NMAC

NMAC is a MAC function based on a hash function. NMAC is a provably secure MAC function. NMAC is constructed from a hash function H as follows:

$$\mathrm{NMAC}_k(x) \overset{\mathrm{def}}{=} H_{k_1}(H_{k_2}(x)),$$

where $k = (k_1, k_2)$ is a secret key, $k_1, k_2 \in \{0,1\}^\ell$ and H_{k_i} represents H with the initial value k_i.

Suppose that H is a hash function such as SHA and MD5. In this case, H_{k_1} is computed with only one application of f since the length of the input to H_{k_1}, which is generated from $H_{k_2}(x)$ with the padding algorithm, is b. Taking this fact into account, we assume that H_{k_1} is computed with one application of the compression function f of the hash function H. Let NMAC-IT(f) represent the iterated hash function H_{k_2} with the compression function f. Let NMAC(f) represent NMAC composed with f.

The following theorem is on the security of NMAC.

Theorem 1: [2] Let $f : \{0,1\}^\ell \times \{0,1\}^b \rightarrow \{0,1\}^\ell$ be a compression function used in NMAC. Then, for every t, q, μ,

$$\begin{aligned}&\mathbf{Insec}_{\mathrm{NMAC}(f)}^{\mathrm{MAC}}(t,q,\mu)\\ &\quad \le \mathbf{Insec}_f^{\mathrm{MAC}}(t,q,q\,b) + \mathbf{Insec}_{\mathrm{NMAC\text{-}IT}(f)}^{\mathrm{WCR}}(t,q,\mu).\end{aligned}$$

◊

Theorem 1 shows that NMAC(f) is a secure MAC if f is a secure MAC and NMAC-IT(f) is WCR. In the next section, two results are presented on the strength of the WCR of NMAC-IT(f).

3. The Strength of the WCR of the Iterated Hash Function in NMAC

For a compression function $f : \{0,1\}^\ell \times \{0,1\}^b \rightarrow \{0,1\}^\ell$ used in NMAC, the set of keys of f, $\Re(f)$, is assumed to be $\{0,1\}^\ell$.

First, it is shown that the PR of the compression function f does not imply the WCR of NMAC-IT(f) even if MD-strengthening is assumed.

Theorem 2: Let g be a function such that $g : \{0,1\}^\ell \times \{0,1\}^b \rightarrow \{0,1\}^\ell$. For every $k \in \{0,1\}^\ell$ and $w \in \{0,1\}^\ell$ such that $d_{\mathrm{H}}(k,w) \le 1$, suppose that

$$g(k, w\|0^{b-\ell}) = w.$$

Then, there exists a weak collision finder of NMAC-IT(g) such that the number of its steps, that of its queries and the total length of its queries are at most $O(\ell\,b + \ell\,T(g))$, 1 and b, respectively, even if MD-strengthening is assumed for padding. ◊

(Proof) Let $\mathcal{A}$ be an algorithm whose behaviour is described below.

1. $\mathcal{A}$ chooses a query $x_1 \in \{0,1\}^b$ to its oracle arbitrarily. Let v_2 be the answer of the oracle to the query x_1. Since MD-strengthening is assumed, the input after padding is (x_1, x_2), where $x_2 = |x_1|_{\mathrm{bin}}$.
2. $\mathcal{A}$ chooses $x_3, x_4, \ldots, x_{\ell+2} \in \{0,1\}^b$ arbitrarily such that $x_3 \notin \{v_2\|0^{b-\ell}, (v_2 \oplus e_1)\|0^{b-\ell}\}$. For $i = 2, 3, \ldots, \ell+1$, $\mathcal{A}$ computes $v_{i+1} = g(v_i, x_{i+1})$.
3. $\mathcal{A}$ computes $x'_3, x'_4, \ldots, x'_{\ell+2}$ in the following way.

```
$v'_2 = v_2$;
    for $j = 1$ to $\ell$ {
    if $\langle v'_2 \oplus v_{\ell+2}\rangle_j = 0$ then $x'_{j+2} = v'_{j+1}\|0^{b-\ell}$;
    else $x'_{j+2} = (v'_{j+1} \oplus e_j)\|0^{b-\ell}$;
    $v'_{j+2} = g(v'_{j+1}, x'_{j+2})$;
    }
```

$\langle v'_2 \oplus v_{\ell+2}\rangle_j$ is the j-th element of $v'_2 \oplus v_{\ell+2} \in \{0,1\}^\ell$ and $e_j \in \{0,1\}^\ell$ such that

$$\langle e_j\rangle_u = \begin{cases} 1 & \text{if } u = j \\ 0 & \text{otherwise} \end{cases}$$

for $u = 1, 2, \ldots, \ell$.
4. $\mathcal{A}$ outputs $x = (x_1, x_2, x_3, \ldots, x_{\ell+2})$ and $x' = (x_1, x_2, x'_3, \ldots, x'_{\ell+2})$.

It is clear that NMAC-IT(g) produces the same output for the inputs x and x'. $x \neq x'$ since $x_3 \neq x'_3$. The number of the queries of $\mathcal{A}$ is 1, the total length of the queries is b, and the number of the steps is at most $O(\ell\, b + \ell\, T(g))$. □

The following theorem shows that there exists a PR function satisfying the property given in Theorem 2 if there exists a PR function.

Theorem 3: Let $g : \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ be defined with $f : \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ as follows:

$$g(k, x) = \begin{cases} w & \text{if } x = w\|0^{b-\ell} \text{ and } d_{\mathrm{H}}(k, w) \le 1 \\ f(k, x) & \text{otherwise.} \end{cases}$$

Then,

$$\mathbf{Insec}_g^{\mathrm{PR}}(t, q, \mu) \le \frac{3}{2}\mathbf{Insec}_f^{\mathrm{PR}}(t + q(\ell+1)\,T(f), q, \mu) + \frac{q(\ell+1)}{2^{\ell+1}}.$$

◊

(Proof) Let $\mathcal{A}$ be a distinguisher of g with the maximum success probability such that the number of its steps, that of its queries and the total length of its queries are at most t, q and μ, respectively.

Let $\mathcal{I}$ be a probabilistic algorithm which works as an interface between $\mathcal{A}$ and an oracle O. First of all, a randomly chosen $s \in \{0,1\}$ is given to O. If $s = 1$, then O chooses $k \in \{0,1\}^\ell$ randomly and uses f_k to produce answers to the queries. If $s = 0$, then O chooses a function from $\{0,1\}^b$ to $\{0,1\}^\ell$ randomly and uses it. $\mathcal{I}$ is a distinguisher of f using $\mathcal{A}$ as a subroutine. The behaviour of $\mathcal{I}$ is described below.

1. For $1 \le i \le q$,
   a. $\mathcal{I}$ gives a query x_i from $\mathcal{A}$ to the oracle O and receives the answer y_i to this query.
   b. If $x_i = w_i\|0^{b-\ell}$ for some $w_i \in \{0,1\}^\ell$, then $\mathcal{I}$ checks whether $y_i = f_{w_i}(x_i)$ and $y_i = f_{w_i \oplus e_j}(x_i)$ for $1 \le j \le \ell$. If at least one equation holds, then $\mathcal{I}$ chooses $s' \in \{0,1\}$ randomly, outputs s' and halts. Otherwise, $\mathcal{I}$ gives y_i to $\mathcal{A}$ as an answer of the oracle to x_i and go to (a).
2. $\mathcal{I}$ receives the output of $\mathcal{A}$, and outputs it as s'.

The number of the steps, that of the queries and the total length of the queries of $(\mathcal{A}, \mathcal{I})$ are at most $t + q(\ell+1)T(f)$, q and μ, respectively. Let Δ be the event that at least one equation holds in 1(b) in the above algorithm. The success probability of this algorithm is evaluated as follows:

$$\begin{aligned}
\Pr[s' = s] &= \Pr[s' = s \wedge \neg\Delta] + \Pr[\Delta]\Pr[s' = s \mid \Delta] \\
&= \Pr[\mathcal{A} \text{ succeeds} \wedge \neg\Delta] + \frac{1}{2}\Pr[\Delta] \\
&\ge \Pr[\mathcal{A} \text{ succeeds}] - \Pr[\Delta] + \frac{1}{2}\Pr[\Delta] \\
&= \frac{1}{2} + \mathbf{Insec}_g^{\mathrm{PR}}(t, q, \mu) - \frac{1}{2}\Pr[\Delta].
\end{aligned}$$

Thus,

$$\mathbf{Insec}_g^{\mathrm{PR}}(t, q, \mu) \le \mathbf{Insec}_f^{\mathrm{PR}}(t + q(\ell+1)\,T(f), q, \mu) + \frac{1}{2}\Pr[\Delta].$$

The rest of the proof is the evaluation of $\Pr[\Delta]$.

Let $\mathcal{I}'$ be a probabilistic algorithm which works as an interface between $\mathcal{A}$ and the oracle O. $\mathcal{I}'$ is a distinguisher of f using $\mathcal{A}$ as a subroutine. The behaviour of $\mathcal{I}'$ is described below.

1. For $1 \le i \le q$,
   a. $\mathcal{I}'$ gives a query x_i from $\mathcal{A}$ to the oracle O and receives the answer y_i to this query.
   b. If $x_i = w_i\|0^{b-\ell}$ for some $w_i \in \{0,1\}^\ell$, then $\mathcal{I}'$ checks whether $y_i = f_{w_i}(x_i)$ and $y_i = f_{w_i \oplus e_j}(x_i)$ for $1 \le j \le \ell$. If at least one equation holds, then $\mathcal{I}'$ outputs $s' = 1$ and halts. Otherwise, $\mathcal{I}'$ gives y_i to $\mathcal{A}$ as an answer of the oracle to x_i and go to (a).
2. $\mathcal{I}'$ outputs $s' = 0$.

It is clear from the description above that the number of the steps, that of the queries and the total length of the queries of $(\mathcal{A}, \mathcal{I}')$ are at most $t + q(\ell+1)T(f)$, q and μ, respectively. The success probability of $(\mathcal{A}, \mathcal{I}')$ is evaluated as follows:

$$\begin{aligned}\Pr[s' = s] &= \frac{1}{2}\Pr[s' = 0|s = 0] + \Pr[s' = 1 \wedge s = 1]\\ &= \frac{1}{2}\Pr[\neg\Delta|s = 0] + \Pr[\Delta \wedge s = 1]\\ &= \frac{1}{2}\Pr[\neg\Delta|s = 0] + \Pr[\Delta] - \frac{1}{2}\Pr[\Delta|s = 0]\\ &= \Pr[\neg\Delta|s = 0] - \frac{1}{2} + \Pr[\Delta]\\ &\geq \left(1 - \frac{\ell+1}{2^\ell}\right)^q - \frac{1}{2} + \Pr[\Delta]\\ &\geq \frac{1}{2} - \frac{q(\ell+1)}{2^\ell} + \Pr[\Delta].\end{aligned}$$

Using this fact, we can obtain

$$\Pr[\Delta] \leq \mathbf{Insec}_f^{\mathrm{PR}}(t + q(\ell+1)T(f), q, \mu) + \frac{q(\ell+1)}{2^\ell}.$$

Thus, we can obtain

$$\begin{aligned}&\mathbf{Insec}_g^{\mathrm{PR}}(t, q, \mu)\\ &\quad\leq \frac{3}{2}\mathbf{Insec}_f^{\mathrm{PR}}(t + q(\ell+1)\,T(f), q, \mu) + \frac{q(\ell+1)}{2^{\ell+1}}.\end{aligned}$$

□

From Theorems 2 and 3, it is straightforward that the WCR of the iterated hash function in NMAC is not implied by the PR of its compression function.

The following theorem shows that the CR of the compression function f is implied by the WCR of NMAC-IT(f) if f is PR.

Theorem 4: For a compression function f and NMAC-IT(f),

$$\begin{aligned}\mathbf{Insec}_f^{\mathrm{CR}}(t) \leq 2\,&\mathbf{Insec}_f^{\mathrm{PR}}(t + O(1), 1)\\ &+\mathbf{Insec}_{\text{NMAC-IT}(f)}^{\mathrm{WCR}}(t + O(1), 1, b).\end{aligned}$$

◊

To prove this theorem, two lemmas are presented.

Lemma 1: If there exists a collision finder of $f : \{0,1\}^\ell \times \{0,1\}^b \rightarrow \{0,1\}^\ell$ with at most t steps, then there exists a weak collision finder of NMAC-IT(f) with at most $t + O(1)$ steps and 1 query of length b. ◊

(Proof) Let $\mathcal{A}$ be a collision finder of f with at most t steps. Let $\mathcal{I}$ be a probabilistic algorithm which works as an interface between $\mathcal{A}$ and an oracle NMAC-IT(f). The behaviour of $\mathcal{I}$ is described below. $\mathcal{I}$ is a weak collision finder of NMAC-IT(f) using $\mathcal{A}$ as a subroutine.

1. $\mathcal{I}$ gives a query $z \in \{0,1\}^b$ to the oracle and gives the answer returned by the oracle to $\mathcal{A}$ as an input.
2. $\mathcal{I}$ obtains the output x, x' from $\mathcal{A}$, outputs (z, x) and (z, x').

It is obvious that the number of the steps, that of the queries and the total length of the queries of $(\mathcal{A}, \mathcal{I})$ are $t + O(1)$, 1 and b, respectively. □

In Lemma 1, the success probability is not mentioned. This is because the success probability of $\mathcal{A}$ depends on the distribution of the key of f given to $\mathcal{A}$ as an input. The following lemma shows that, if the difference is not negligible between the success probability of $\mathcal{A}$ with the key randomly selected and that of $\mathcal{A}$ with the interface $\mathcal{I}$ and the oracle NMAC-IT(f) in Lemma 1, then it is able to be used for distinguishing f from a random function.

Lemma 2: Let $\mathcal{A}$ be a collision finder of f with at most t steps. Let ε_1 be the success probability of $\mathcal{A}$ when the key of f given to $\mathcal{A}$ is randomly selected. Let ε_2 be the success probability of $\mathcal{A}$ with the interface $\mathcal{I}$ and the oracle NMAC-IT(f) as in Lemma 1. Then, a distinguisher of f is able to be constructed with $\mathcal{A}$ and the number of its steps and that of its queries are $t + O(1)$ and 1, respectively, and its success probability is at least $\frac{1}{2} + |\varepsilon_1 - \varepsilon_2|$. ◊

(Proof) Suppose that $\varepsilon_1 \leq \varepsilon_2$. Let $\mathcal{I}'$ be a probabilistic algorithm which works as an interface between $\mathcal{A}$ and an oracle $\mathcal{O}$. First of all, a randomly chosen bit $s \in \{0,1\}$ is given to $\mathcal{O}$. If $s = 1$, then $\mathcal{O}$ chooses $k \in \{0,1\}^\ell$ randomly and uses f_k to produce answers to the queries. If $s = 0$, then $\mathcal{O}$ chooses a function from $\{0,1\}^b$ to $\{0,1\}^\ell$ randomly and uses it. $\mathcal{I}'$ is a distinguisher of f using $\mathcal{A}$ as a subroutine. The behaviour of $\mathcal{I}'$ is described below.

1. $\mathcal{I}'$ chooses a query $z \in \{0,1\}^b$ to the oracle in the same way as $\mathcal{I}$ in Lemma 1 and gives the answer from the oracle to $\mathcal{A}$ as an input.
2. $\mathcal{I}'$ determines s' in the following way and outputs it.

$$s' = \begin{cases} 0 & \text{if } \mathcal{A} \text{ fails to find a collision,}\\ 1 & \text{if } \mathcal{A} \text{ succeeds in finding a collision.}\end{cases}$$

It is obvious from the above description that the number of the steps, that of the queries and the total length of the queries of $(\mathcal{A}, \mathcal{I}')$ are $t + O(1)$, 1 and b, respectively. The success probability of this algorithm is evaluated as follows:

$$\begin{aligned}&\Pr[s' = s]\\ &\quad= \Pr[s' = 0 \wedge s = 0] + \Pr[s' = 1 \wedge s = 1]\\ &\quad= \Pr[s{=}0]\Pr[s'{=}0|s{=}0] + \Pr[s{=}1]\Pr[s'{=}1|s{=}1]\\ &\quad= \frac{1}{2}\Pr[\mathcal{A}\text{ fails}|s = 0] + \frac{1}{2}\Pr[\mathcal{A}\text{ succeeds}|s = 1]\\ &\quad= \frac{1}{2}(1 - \varepsilon_1) + \frac{1}{2}\varepsilon_2\\ &\quad= \frac{1}{2} + \frac{1}{2}(\varepsilon_2 - \varepsilon_1).\end{aligned}$$

Suppose that $\varepsilon_2 \leq \varepsilon_1$. Then, in the behaviour of $\mathcal{I}'$described above, $\mathcal{I}'$ determines s' as follows:

$$s' = \begin{cases} 0 & \text{if } \mathcal{A} \text{ succeeds in finding a collision,}\\ 1 & \text{if } \mathcal{A} \text{ fails to find a collision.}\end{cases}$$

Then, the success probability of $(\mathcal{A}, \mathcal{I}')$ is $\frac{1}{2} + \frac{1}{2}(\varepsilon_1 - \varepsilon_2)$. □

Theorem 4 is led from Lemmas 1 and 2 in the following way.

Proof of Theorem 4.

Since ε_2 is the success probability of $\mathcal{A}$ with the interface $\mathcal{I}$ and the oracle NMAC-IT(f) in Lemma 1,

$$\varepsilon_2 \leq \mathbf{Insec}^{\mathrm{WCR}}_{\text{NMAC-IT}(f)}(t + O(1), 1, b).$$

On the other hand, since $|\varepsilon_1 - \varepsilon_2| \leq 2\,\mathbf{Insec}^{\mathrm{PR}}_{f}(t + O(1), 1)$ from Lemma 2,

$$\varepsilon_1 \leq \varepsilon_2 + 2\,\mathbf{Insec}^{\mathrm{PR}}_{f}(t + O(1), 1).$$

Thus, we are able to obtain

$$\mathbf{Insec}^{\mathrm{CR}}_{f}(t) \leq 2\,\mathbf{Insec}^{\mathrm{PR}}_{f}(t + O(1), 1) + \mathbf{Insec}^{\mathrm{WCR}}_{\text{NMAC-IT}(f)}(t + O(1), 1, b)$$

by assuming that $\mathcal{A}$ is a collision finder of f with the maximum success probability. □

4. Conclusion

For a compression function with fixed input length, the WCR is implied by the MAC security. In contrast, it is shown in this article that the CR of a compression function with fixed input length can be implied by the WCR of the iterated hash function in NMAC with variable input length. These facts show that there may be a gap between the WCR of a function with fixed input length and that of a function with variable input length.

References

[1] J.H. An and M. Bellare, “Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions,” Proc. CRYPTO’99, pp.252–269, Lect. Notes Comput. Sci. 1666, 1999.

[2] M. Bellare, R. Canetti, and H. Krawczyk, “Keying hash functions for message authentication,” Proc. CRYPTO’96, Lect. Notes Comput. Sci. 1109, pp.1–15, 1996.

[3] M. Bellare, R. Canetti, and H. Krawczyk, “Pseudorandom functions revisited: The cascade construction and its concrete security,” Proc. 37th IEEE Symposium on Foundations of Computer Science, pp.514–523, 1996.

[4] M. Bellare, R. Canetti, and H. Krawczyk, “HMAC: Keyed-hashing for message authentication,” Network Working Group RFC 2104, Lect. Notes Comput. Sci. 1109, 1997.

[5] M. Bellare, J. Kilian, and P. Rogaway, “The security of the cipher block chaining message authentication code,” J. Comput. Syst. Sci., vol.61, no.3, pp.362–399, 2000.

[6] M. Bellare and P. Rogaway, “Collision-resistant hashing: Towards making UOWHFs practical,” Proc. CRYPTO’97, pp.470–484, Lect. Notes Comput. Sci. 1294, 1997.

[7] I. Damgård, “A design principle for hash functions,” Proc. CRYPTO’89, pp.416–427, Lect. Notes Comput. Sci. 435, 1990.

[8] D.R. Simon, “Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?” Proc. EUROCRYPT’98, pp.334–345, Lect. Notes Comput. Sci. 1403, 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 1997, he was a research associate at Faculty of Engineering, Kyoto University. From 1998, he is a lecturer at the Graduate School of Informatics, Kyoto University. His current interests include cryptography, information security and computational complexity. He received Young Engineer Award from IEICE in 1997. He is a member of ACM, IEEE, IACR and IPSJ.