2008－CSEC－42 (31)
2008 / 7 / 25

SHA-256圧縮関数の擬似ランダム関数性に関する解析

桑門 秀典† 廣瀬 勝一††

† 神戸大学大学院工学研究科 〒657–8501 兵庫県神戸市灘区六甲台町 1–1
†† 福井大学工学研究科 〒910–8507 福井県福井市文京 3–9–1

あらまし HMACなどのアプリケーションでは，ハッシュ関数の圧縮関数の擬似ランダム関数性が重要である．本論文では，SHA-256圧縮関数の擬似ランダム関数性について検討した．その結果，step 22までのSHA-256圧縮関数は，ランダム関数と容易に識別できることが判明した．

キーワード SHA-256，圧縮関数，擬似ランダム関数性

Analysis on the Pseudorandom-Function Property of the SHA-256 Compression Function

Hidenori KUWAKADO† and Shoichi HIROSE††

† Graduate School of Engineering, Kobe University 1-1 Rokkodai-cho Nada-ku Kobe, 657-8501, Japan
†† Graduate School of Engineering, The University of Fukui 3-9-1 Bunkyo Fukui City, 910-8507, Japan

Abstract Applications of an iterated hash function such as HMAC require that the compression function of the hash function is a pseudorandom function, that is, it is computationally infeasible to distinguish between the compression function and a random function. This paper shows that it is easy to distinguish between the 22 step-reduced SHA-256 compression function and the random function.

Key words SHA-256, compression function, pseudorandom function

1. Introduction

As the SHA-256 hash function [6] has gotten attention recently by the cryptanalysis community, the analysis on the SHA-256 hash function has developed remarkably. For example, Mendel *et al.* [5] showed an 18-step collision in 2006. Sanadhya and Sarkar [7] presented differential paths for 19 – 23 steps of SHA-256 in 2008. Indesteege *et al.* [4] showed collision attacks on SHA-256 reduced to 23 and 24 steps with complexities 2^{18} and 2^{50} in 2008. In addition, they also pointed out the non-random behavior of SHA-256 in the form of pseudo-near collision for up to 31 steps [4]. Thus, previous results are related to the security of collision-resistance or that of non-random behavior based on the collision. We note that the collision-resistant property is important, but it is not all.

Applications such as HMAC require that a (keyed) hash function behaves as if it is a random function when the key is unknown. This property is called the *pseudorandom-function property*, which is closely related to the security of such applications. The pseudorandom-function property and the collision-resistant property are independent in the sense that there is no general reduction between them. Indeed, the attack model on the collision-resistant property differs from the attack model on the pseudorandom-function property. In the attack model on the collision-resistant property, an adversary can search it only by himself, without other's help. Besides, in the attack model on the pseudorandom-function property, an adversary cannot obtain a hashed value without making queries to an oracle that knows a secret key. Accordingly, a collision-resistant hash function is not necessarily a hash function with the pseudorandom-function property, and vice versa. In particular, the pseudorandom-function property of SHA-256 is not studied from the viewpoint of actual attacks.

On the other hand, the pseudorandom-function property of a hash function has been discussed in the context of domain extension [1] [2] [3]. Specifically, under the assumption that an underlying compression function is a pseudorandom function, methods for constructing a hash function that behaves as if it is the pseudorandom function have been studied. Hence, since the pseudorandom-function property of such a hash function is reduced to that of the underlying compression function, it is worth studying the pseudorandom-function property of the compression function.

In this paper, we show that the 22 step-reduced SHA-256 compression function with the key-via-IV strategy is distinguishable from the random function. This is the first result on the pseudorandom-function property of the SHA-256 compression function. Our distinguishing attack is practical in the sense that the probability that the attack succeeds in distinguishing them is large with the reasonable number of queries and low complexity.

This paper is organized as follows. Section 2 describes the algorithm of the SHA-2 compression function and the key-via-IV strategy to transform a non-keyed compression function into a keyed compression function, and defines the prf-advantage to measure the indistinguishability of a function. Section 3 shows an algorithm for distinguishing between a 22 step-reduced SHA-256 compression function and a random function. We demonstrate that the prf-advantage of this algorithm is large, that is, this algorithm is computationally feasible. Furthermore, we show the differential path used by this algorithm and evaluate the probability that the algorithm succeeds in distinguishing them. Section 4 concludes this paper.

2. Preliminaries

2.1 SHA-256 Compression Function

We here describe the algorithm of the SHA-256 compression function (Fig. 1). In the following, all variables are 32 bits, an operation '+' denotes an addition modulo 2^{32}, an operation '$\oplus$' denotes the bitwise exclusive-or, and N represents the number of steps. The SHA-256 compression function consists of 64 steps, that is, $N = 64$.

First, prepare expanded message blocks w_i for given message blocks $m_0, m_1, \ldots, m_{15}$.

$$w_i = \begin{cases} m_i & \text{if } 0 \leq i \leq 15, \\ \sigma_1(w_{i-2}) + w_{i-7} + \sigma_0(w_{i-15}) + w_{i-16} & \\ & \text{if } 16 \leq i \leq N-1. \end{cases} \quad (1)$$

In Eq. (1), functions σ_0, σ_1 are defined as

$$\begin{aligned} \sigma_0(x) &= \mathrm{ROTR}(7, x) \oplus \mathrm{ROTR}(18, x) \oplus \mathrm{SHR}(3, x), \\ \sigma_1(x) &= \mathrm{ROTR}(17, x) \oplus \mathrm{ROTR}(19, x) \oplus \mathrm{SHR}(10, x), \end{aligned}$$

where $\mathrm{ROTR}(n, x)$ is a circular shift of x by n positions to the right and $\mathrm{SHR}(n, x)$ is a shift of x by n positions to the right.

Next, suppose that $(a_{-1}, b_{-1}, \ldots, h_{-1})$ are given, for example, as initial values. For $i = 0$ to $N-1$, compute the variables as rules where α and β are intermediate variables.

$$\begin{aligned} &\alpha = \Sigma_0(a_{i-1}) + \mathrm{Maj}(a_{i-1}, b_{i-1}, c_{i-1}), \\ &\beta = h_{i-1} + \Sigma_1(e_{i-1}) + \mathrm{Ch}(e_{i-1}, f_{i-1}, g_{i-1}) + k_i + w_i, \\ &a_i = \alpha + \beta, \quad b_i = a_{i-1}, \quad c_i = b_{i-1}, \quad d_i = c_{i-1}, \\ &e_i = d_{i-1} + \beta, \quad f_i = e_{i-1}, \quad g_i = f_{i-1}, \quad h_i = g_{i-1}, \end{aligned} \quad (2)$$

where k_i is a constant value and functions $\Sigma_0, \Sigma_1, \mathrm{Ch}$, and Maj are defined as

$$\begin{aligned} &\Sigma_0(x) = \mathrm{ROTR}(2, x) \oplus \mathrm{ROTR}(13, x) \oplus \mathrm{ROTR}(22, x), \\ &\Sigma_1(x) = \mathrm{ROTR}(6, x) \oplus \mathrm{ROTR}(11, x) \oplus \mathrm{ROTR}(25, x), \\ &\mathrm{Ch}(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z), \\ &\mathrm{Maj}(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z). \end{aligned} \quad (3)$$

Finally, add the initial values to them.

$$\begin{aligned} &a_N = a_{N-1} + a_{-1}, \quad b_N = a_{N-1} + b_{-1}, \\ &c_N = c_{N-1} + c_{-1}, \quad d_N = d_{N-1} + d_{-1}, \\ &e_N = e_{N-1} + e_{-1}, \quad f_N = f_{N-1} + f_{-1}, \\ &g_N = g_{N-1} + g_{-1}, \quad h_N = h_{N-1} + h_{-1}. \end{aligned}$$

The result $(a_N, b_N, \ldots, h_N)$ is output of the compression function.

In this paper, we discuss the step-reduced compression functions of SHA-256. We denote by SHA-256/N the SHA-256 compression function reduced to N steps. Note that the step-reduced compression function includes the final addition of the initial values, which is often ignored in collision attacks.

Although SHA-256/N is not a keyed compression function, it is possible to use SHA-256/N as the keyed compression function by replacing $(a_{-1}, b_{-1}, \ldots, h_{-1})$ with a 256-bit key. The replacement is often called the *key-via-IV strategy*. This paper argues this type of keyed SHA-256/N.

2.2 Pseudorandom-Function Property

Let $\mathcal{F}_{\ell,n}$ be the set of all functions from $\{0,1\}^\ell$ to $\{0,1\}^n$. A function f is called a *random function* if f is randomly chosen from $\mathcal{F}_{\ell,n}$. Consider a function $\phi(k, x) : \{0,1\}^\kappa \times \{0,1\}^\ell \rightarrow \{0,1\}^n$. After k was fixed, $\phi(k, x)$ can be considered as a function in $\mathcal{F}_{\ell,n}$. Such a function is denoted by $\phi_k(x)$. The function $\phi(k, x)$ is called a *pseudorandom function* if it is infeasible for an adversary who does not know k to distinguish between $\phi_k(x)$ and a random function $f(x)$ in $\mathcal{F}_{\ell,n}$. Formally, the indistinguishability is measured by the *prf-advantage* of an adversary A that is defined as

$$\begin{aligned} \mathbf{Adv}^{\mathrm{prf}}_{\phi_k}(A) = \; & \Pr\left[k \xleftarrow{\$} \{0,1\}^\kappa; A^{\phi_k} \Rightarrow 1\right] \\ & - \Pr\left[f \xleftarrow{\$} \mathcal{F}_{\ell,n}; A^f \Rightarrow 1\right] \end{aligned} \quad (4)$$

where A has access to ϕ_k (or f) and returns a bit [1]. It should be noted that $\phi(k, x)$ is public, that is, anyone including A knows the description of $\phi(k, x)$ and is capable for given values k, x of computing $\phi(k, x)$, but A does not know k. If the prf-advantage is negligibly small, then $\phi(k, x)$ is called the pseudorandom function.

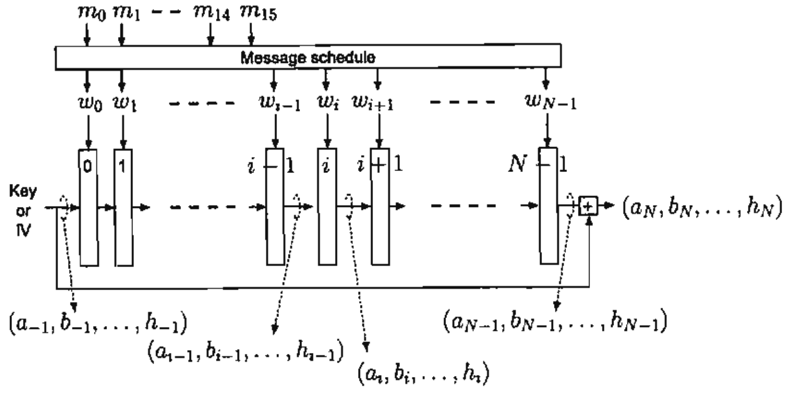

Fig. 1 SHA-256/N.

3. 22-Step Reduced SHA-256 Compression Function

3.1 Distinguishing Algorithm

Suppose that an adversary A has access to an oracle G that is the keyed SHA-256/22 ϕ_k or a random function f in $\mathcal{F}_{512,256}$. The goal of A is to determine whether G is ϕ_k or f. We define an algorithm of A as follows.

1. Set a counter c to 0.
2. For $i = 0, 1, \ldots, 11, 14, 15$, choose a message block m_i randomly and independently. Set another message block m'_i to m_i for $i = 0, 1, \ldots, 11, 14, 15$.
3. For $i = 12, 13$, set m_i, m'_i as follows.

$$\begin{aligned} m_{12} &= \mathsf{00000000}, & m'_{12} &= \mathsf{80000000}, \\ m_{13} &= \Sigma_1(\mathsf{80000000}), & m'_{13} &= \mathsf{00000000}, \end{aligned} \quad (5)$$

where sans-serif fonts are used to represent hex digits.

4. Send message blocks $(m_0, m_1, \ldots, m_{15})$ to the oracle G, and receive its hash blocks $(a, b, \ldots, h)$.
5. Send another message blocks $(m'_0, m'_1, \ldots, m'_{15})$ to G, and receive its hash blocks $(a', b', \ldots, h')$.
6. Compute the modular difference of h' and h, that is,

$$\delta h = h' - h \bmod 2^{32}. \quad (6)$$

7. If the following conditions are satisfied, then increment c by one.

$$\delta h[i] = \begin{cases} 0 & \text{if } 0 \leq i \leq 2, \\ 1 & \text{if } i = 3, \end{cases} \quad (7)$$

where $\delta h[i]$ denotes the i-bit value of δh. Note that the least significant bit is the 0-th bit.

8. If the number of queries is less than q, then go to step 2, otherwise go to step 9.
9. If $c \geq \lceil (q/2)c_t \rceil$ where $c_t = (2^{-4} + 2^{-3.91})/2$, then A outputs 1, otherwise A outputs 0.

Section 3.2 will show that the probability of Eq. (7) is $2^{-3.91}$ if G is SHA-256/22, and it is 2^{-4} if G is the random function. Let $\mathrm{E_{SHA}}$ and $\mathrm{E_{RF}}$ be an event that G is SHA-256/22 and an event that G is the random function, respectively. For each case, the probability that A outputs 1 is calculated as follows.

$$\Pr\left[A^G \Rightarrow 1 | \mathrm{E_{SHA}}\right] = \sum_{c=\lceil (q/2)c_t \rceil}^{q/2} {}_{q/2}\mathrm{C}_c (2^{-3.91})^c (1 - 2^{-3.91})^{q/2-c}$$

$$\Pr\left[A^G \Rightarrow 1 | \mathrm{E_{RF}}\right] = \sum_{c=\lceil (q/2)c_t \rceil}^{q/2} {}_{q/2}\mathrm{C}_c (2^{-4})^c (1 - 2^{-4})^{q/2-c}$$

Therefore, the prf-advantage of A is given by

$$\mathbf{Adv}^{\mathrm{prf}}_{\phi_k}(A) = \Pr\left[A^G \Rightarrow 1 | \mathrm{E_{SHA}}\right] - \Pr\left[A^G \Rightarrow 1 | \mathrm{E_{RF}}\right]. \quad (8)$$

We here give a numerical example to demonstrate the prf-advantage of the adversary A. Suppose that the number of queries q is 2^{17}, that is, A obtains 2^{16} difference δh's. Since $\lceil (q/2)c_t \rceil = 4228$, probabilities are calculated as follows.

$$\Pr\left[A^G \Rightarrow 1 | \mathrm{E_{SHA}}\right] = \sum_{c=4228}^{2^{16}} {}_{2^{16}}\mathrm{C}_c (2^{-3.91})^c (1 - 2^{-3.91})^{2^{16}-c} \approx 0.981203,$$

$$\Pr\left[A^G \Rightarrow 1 | \mathrm{E_{RF}}\right] = \sum_{c=4228}^{2^{16}} {}_{2^{16}}\mathrm{C}_c (2^{-4})^c (1 - 2^{-4})^{2^{16}-c} \approx 0.0172595.$$

Hence, the prf-advantage of A is approximated by

$$\mathbf{Adv}^{\mathrm{prf}}_{\phi_k}(A) \approx 0.981203 - 0.0172595 = 0.9639435,$$

which means that distinguishing between SHA-256/22 and the random function is easy.

3.2 Analysis

This subsection explains why Eq. (7) holds with probability $2^{-3.91}$ when the oracle G is SHA-256/22. Suppose that

Table 1 Differential path for SHA-256/22.

Step i	Δw_i	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	Δf_i	Δg_i	Δh_i	Prob.
0-11	0	0	0	0	0	0	0	0	0	–
12	Δw_{12}	Δw_{12}	0	0	0	Δw_{12}	0	0	0	1
13	Δw_{13}	$\boxed{\Delta a_{13}}$	Δw_{12}	0	0	$\boxed{0}$	Δw_{12}	0	0	$2^{-4.268}$
14	0	Δa_{14}	Δa_{13}	Δw_{12}	0	$\boxed{0}$	0	Δw_{12}	0	2^{-1}
15	0	*	Δa_{14}	Δa_{13}	Δw_{12}	$\boxed{0}$	0	0	Δw_{12}	2^{-1}
16	0	*	*	Δa_{14}	Δa_{13}	0	0	0	0	1
17	0	*	*	*	Δa_{14}	Δe_{17}	0	0	0	–
18	0	*	*	*	*	Δe_{18}	Δe_{17}	0	0	–
19	*	*	*	*	*	*	Δe_{18}	Δe_{17}	0	–
20	*	*	*	*	*	*	*	Δe_{18}	Δe_{17}	–
21	*	*	*	*	*	*	*	*	Δe_{18}	–

G is SHA-256/22. The algorithm in Sect. 3.1 is based on the differential path shown in Table 1. Unlike δ of Eq. (6), Δ in Table 1 means the bitwise exclusive-or difference. For example, Δw_i denotes the bitwise exclusive-or difference of w_i and w'_i, that is,

$$\Delta w_i = w_i \oplus w'_i,$$

where w_i is computed from message blocks m_j ($j = 0, 1, \ldots, 15$) according to Eq. (1) and w'_i is done from m'_j similarly. Since $w_{12} = m_{12}$ and $w'_{12} = m'_{12}$, Eq. (5) yields

$$\begin{aligned}\Delta w_{12} &= m_{12} \oplus m'_{12}\\ &= \mathsf{00000000} \oplus \mathsf{80000000}\\ &= \mathsf{80000000}.\end{aligned}$$

Similarly, Δw_{13} is given by

$$\begin{aligned}\Delta w_{13} &= m_{13} \oplus m'_{13}\\ &= \Sigma_1(\mathsf{80000000}).\end{aligned}$$

First, consider step 12 in Table 1. The intermediate variables α, β in Eq. (2) are calculated by

$$\begin{aligned}\alpha &= \Sigma_0(a_{11}) + \mathrm{Maj}(a_{11}, b_{11}, c_{11}),\\ \alpha' &= \Sigma_0(a'_{11}) + \mathrm{Maj}(a'_{11}, b'_{11}, c'_{11}),\\ \beta &= h_{11} + \Sigma_1(e_{11}) + \mathrm{Ch}(e_{11}, f_{11}, g_{11}) + k_{12} + w_{12},\\ \beta' &= h'_{11} + \Sigma_1(e'_{11}) + \mathrm{Ch}(e'_{11}, f'_{11}, g'_{11}) + k_{12} + w'_{12}.\end{aligned}$$

Since $a_{11} = a'_{11}$, $b_{11} = b'_{11}$, etc., a_{12} and a'_{12} are given by

$$a_{12} = \gamma + w_{12}, \quad a'_{12} = \gamma + w'_{12},$$

where

$$\begin{aligned}\gamma = {} & \Sigma_0(a_{11}) + \mathrm{Maj}(a_{11}, b_{11}, c_{11})\\ & + h_{11} + \Sigma_1(e_{11}) + \mathrm{Ch}(e_{11}, f_{11}, g_{11}) + k_{12}.\end{aligned}$$

Since $w_{12} = \mathsf{00000000}$ and $w'_{12} = \mathsf{80000000}$, Δa_{12} is

$$\begin{aligned}\Delta a_{12} &= a_{12} \oplus a'_{12}\\ &= \gamma \oplus (\gamma + \mathsf{80000000})\\ &= \mathsf{80000000} = \Delta w_{12}.\end{aligned}$$

Note that the modular addition of 80000000 is identical to the bitwise exclusive-or of 80000000. In a similar way, we obtain $\Delta e_{12} = \Delta w_{12}$. Hence, the difference in step 12 occurs with probability 1.

Second, let us consider Δe_{13} in Table 1. Using the intermediate variable β, e_{13} and e'_{13} are given by

$$e_{13} = d_{12} + \beta, \quad e'_{13} = d'_{12} + \beta',$$

where intermediate variables are

$$\begin{aligned}\beta &= h_{12} + \Sigma_1(e_{12}) + \mathrm{Ch}(e_{12}, f_{12}, g_{12}) + k_{13} + w_{13},\\ \beta' &= h'_{12} + \Sigma_1(e'_{12}) + \mathrm{Ch}(e'_{12}, f'_{12}, g'_{12}) + k_{13} + w'_{13}.\end{aligned}$$

Notice that the following equalities hold.

$$\begin{aligned}& d_{12} = d'_{12}\\ & h_{12} = h'_{12}\\ & \Sigma_1(e'_{12}) = \Sigma_1(e_{12}) \oplus \Sigma_1(\mathsf{80000000})\\ & \mathrm{Ch}(e'_{12}, f'_{12}, g'_{12}) = \mathrm{Ch}(e_{12} \oplus \mathsf{80000000}, f_{12}, g_{12})\\ & w_{13} = \Sigma_1(\mathsf{80000000})\\ & w'_{13} = \mathsf{00000000}\end{aligned}$$

Hence, if the following equations hold, then Δe_{13} becomes 0 because $\beta = \beta'$.

$$\mathrm{Ch}(e_{12}, f_{12}, g_{12}) = \mathrm{Ch}(e_{12} \oplus \mathsf{80000000}, f_{12}, g_{12}) \quad (9)$$

$$\begin{aligned}& \Sigma_1(e_{12}) + \Sigma_1(\mathsf{80000000})\\ & \quad = \Sigma_1(e_{12}) \oplus \Sigma_1(\mathsf{80000000})\end{aligned} \quad (10)$$

Here, Eq. (9) is equivalent to

$$f_{12}[31] = g_{12}[31], \quad (11)$$

where $\vartheta[i]$ denotes the i-bit value of a variable ϑ, and Eq. (10) is equivalent to

$$\Sigma_1(e_{12})[25] = \Sigma_1(e_{12})[20] = \Sigma_1(e_{12})[6] = 0. \quad (12)$$

If Eq. (11) and Eq. (12) hold, then Δe_{13} becomes 0. Suppose that the computation from step 0 to step 11 in SHA-256/22 behaves as if it is the random function. Denoting by $E_{e_{13}}$ the event that Eq. (11) and Eq. (12) hold, we have

$$\Pr[E_{e_{13}}|E_{SHA}] = 2^{-4}, \tag{13}$$

where E_{SHA} is the event that G is SHA-256/22. Notice that Eq. (11) and Eq. (12) are sufficient for $\Delta e_{13} = 0$, but are not necessary. However, our computer experiment showed that 2^{-4} was a close approximation to $\Pr[\Delta e_{13} = 0|E_{SHA}]$.

Third, consider Δa_{13} under the assumption of Eq. (11) and Eq. (12). The difference Δa_{13} is computed from

$$a_{13} = \Sigma_0(a_{12}) + \mathrm{Maj}(a_{12}, b_{12}, c_{12}) + \beta,$$
$$a'_{13} = \Sigma_0(a'_{12}) + \mathrm{Maj}(a'_{12}, b'_{12}, c'_{12}) + \beta',$$

where $a'_{12} = a_{12} \oplus 80000000$ and $\beta = \beta'$. However, it may be difficult to obtain the formula of Δa_{13} because the value of Δa_{13} depends on a_{12}, Maj, and β. Hence, we consider only bits of Δa_{13} that obviously affect the four least significant bits of Δe_{18}. Note that the four least significant bits of Δe_{18} is used in Eq. (7). The four least significant bits of $\Sigma_1(\Delta e_{17})$ affect those of Δe_{18}. Supposing that Δa_{13} gives a approximation to Δe_{17}, we see that the 6-th bit of Δa_{13} affect the 0-th bit of Δe_{18}. In a similar manner, we pick up bits that affect those of Δe_{18} through the function Σ_1, which are shown in the upper half of Table 2. In addition, since the four least significant bits of Δa_{14} affect those of Δe_{18}, Δa_{13} affects those of Δe_{18} through the function Σ_0. For example, the 2-nd bit of Δa_{13} affects the 0-th bit of Δe_{18} through the function Σ_0. In a similar manner, we pick up bits that affect those of Δe_{18} through the function Σ_0, which are shown in the lower half of Table 2. We impose the following conditions on Δa_{13}. Using x instead of Δa_{13},

$$x[6] \oplus x[12] \oplus x[25] \oplus x[2] \oplus x[13] \oplus x[22] = 0$$
$$x[7] \oplus x[13] \oplus x[26] \oplus x[3] \oplus x[14] \oplus x[23] = 0$$
$$x[8] \oplus x[14] \oplus x[27] \oplus x[4] \oplus x[15] \oplus x[24] = 0$$
$$x[9] \oplus x[15] \oplus x[28] \oplus x[4] \oplus x[16] \oplus x[25] = 1$$

where values in the right hand correspond to Eq. (7). Notice that we expect that the four least significant bits in δh is equal to those of Δe_{18}. The probability that all the above equations hold was $2^{-0.268}$ by our computer experiment. Formally, denoting by $E_{a_{13}}$ the event that all the above equations hold, we have

$$\Pr[E_{a_{13}}|E_{SHA} \wedge E_{e_{13}}] = 2^{-0.268}. \tag{14}$$

Fourth, consider Δe_{14} in Table 2 under the assumption of $E_{e_{13}}$ and $E_{a_{13}}$. From the definition of Ch (i.e., Eq. (3)), $\Delta e_{14} = 0$ if and only if $e_{13}[31] = e'_{13}[31] = 0$ because

Table 2 Influential bits.

Δe_{18}	3	2	1	0
Σ_1	9	8	7	6
	15	14	13	12
	28	27	26	25
Σ_0	5	4	3	2
	16	15	14	13
	25	24	23	22

$\Delta e_{13} = 0$. Denoting by $E_{e_{14}}$ the event of $\Delta e_{14} = 0$, we have

$$\Pr[E_{e_{14}}|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}}] \tag{15}$$
$$= \Pr\left[e_{13}[31] = e'_{13}[31] = 0|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}}\right]$$
$$= 2^{-1}, \tag{16}$$

where we assume that the computation from step 0 to step 11 in SHA-256/22 behaved as if it is the random function.

Finally, consider Δe_{15} in Table 2 under the assumption of $E_{e_{13}}$, $E_{a_{13}}$, and $E_{e_{14}}$. Under this assumption, $\Delta e_{15} = 0$ if and only if $e_{14}[31] = e'_{14}[31] = 1$. Denoting by $E_{e_{15}}$ the event of $\Delta e_{15} = 0$, we have

$$\Pr[E_{e_{15}}|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}} \wedge E_{e_{14}}]$$
$$= \Pr\left[e_{14}[31] = e'_{14}[31] = 1|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}} \wedge E_{e_{14}}\right]$$
$$= 2^{-1}. \tag{17}$$

Let E_{dif} be the event of $E_{e_{13}} \wedge E_{a_{13}} \wedge E_{e_{14}} \wedge E_{e_{15}}$. Combining Eqs. (13)–(17) gives the probability $\Pr[E_{dif}|E_{SHA}]$ that all the four events occur.

$$\Pr[E_{dif}|E_{SHA}] = \Pr[E_{e_{13}} \wedge E_{a_{13}} \wedge E_{e_{14}} \wedge E_{e_{15}}|E_{SHA}]$$
$$= \Pr[E_{e_{13}}|E_{SHA}] \cdot \Pr[E_{a_{13}}|E_{SHA} \wedge E_{e_{13}}]$$
$$\cdot \Pr[E_{e_{14}}|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}}]$$
$$\cdot \Pr[E_{e_{15}}|E_{SHA} \wedge E_{e_{13}} \wedge E_{a_{13}} \wedge E_{e_{14}}]$$
$$= 2^{-6.268}$$

Let E_{ok} be the event that Eq. (7) holds. The probability of E_{ok} when G is SHA-256/22 is expressed as

$$\Pr[E_{ok}|E_{SHA}] = \Pr[E_{ok}|E_{SHA} \wedge E_{dif}]\Pr[E_{SHA} \wedge E_{dif}]$$
$$+\Pr\left[E_{ok}|E_{SHA} \wedge \overline{E_{dif}}\right]\Pr\left[E_{SHA} \wedge \overline{E_{dif}}\right], \tag{18}$$

where $\overline{E_{dif}}$ is the complement of E_{dif}. Our computer experiment showed that

$$\Pr[E_{ok}|E_{SHA} \wedge E_{dif}] = 2^{-1.441}.$$

Supposing that SHA-256/22 behaves as if it is the random function when not all the four events do occur, we have

$$\Pr\left[E_{ok}|E_{SHA} \wedge \overline{E_{dif}}\right] = 2^{-4},$$

which compares well with the result of our computer experiment. Substituting these probabilities into Eq. (18) yields the probability of E_{ok} when the oracle G is SHA-256/22 as follows.

$$\Pr[E_{ok}|E_{SHA}] = 2^{-1.441} \cdot 2^{-6.268} + 2^{-4}(1 - 2^{-6.268}) \approx 2^{-3.91}.$$

On the other hand, if G is the random function, then the probability that Eq. (7) holds is 2^{-4}, that is,

$$\Pr[E_{ok}|E_{RF}] = 2^{-4}.$$

These two probabilities provide the validity to the algorithm described in Sect. 3.1.

As shown by the numerical example of Sect. 3.1, distinguishing between SHA-256/22 and the random function is reduced to distinguishing between the binomial distribution with probability $2^{-3.91}$ and that with probability 2^{-4}. In the algorithm of Sect. 3.1, the output '1' implicitly means that G is SHA-256/22, and the output '0' means that G is the random function. Hence, in order to distinguish them with high probability, the judgment condition in step 6 of the algorithm should be chosen so that the following error probability is minimum.

$$\Pr[E_{err}] = \Pr[A \Rightarrow 1|E_{RF}] + \Pr[A \Rightarrow 0|E_{SHA}] \quad (19)$$

Notice that minimizing the error probability of Eq. (19) is not equivalent to maximizing the prf-advantage of Eq. (8).

4. Concluding Remarks

The previous analysis of hash functions focused on collision-resistance. However, applications often require that hash functions have not only the collision-resistant property but also the pseudorandom-function property. These two properties are independent in the sense that the collision-resistant property does not follow the pseudorandom-function property and vice versa. The collision-resistant property of the SHA-256 compression function has been extensively studied, but its pseudorandom-function property has not been done.

This paper provided the first result on the pseudorandom-function property of the SHA-256 compression function. We showed the practical attack for distinguishing between the 22 step-reduced SHA-256 compression function and the random function. Since the attack is based on the differential path, the success probability of the attack can probably be evaluated from the differential path theoretically. Since the differential path, however, involves complicated conditions, we partially used the result of computer experiments to evaluate the success probability. Additionally, a similar distinguishing attack is applicable to the step-reduced SHA-512 compression function.

Acknowledgments

Fruitful discussions with Dr. Yoshida and Dr. Ideguchi at Hitachi, Ltd. and Prof. Ohta at The University of Electro-Communications are greatly appreciated. This research was supported by the National Institute of Information and Communications Technology, Japan.

References

[1] M. Bellare and T. Ristenpart, "Multi-property-preserving hash domain extension and the EMD transform," Advances in Cryptology - ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4248, pp. 299–314, 2006.

[2] M. Bellare and T. Ristenpart, "Hash functions in the dedicated-key setting: Design choices and MPP transforms," International Colloquium on Automata, Languages and Programming, ICALP 2007, Lecture Notes in Computer Science, vol. 4596, pp. 399–410, 2007. Cryptology ePrint Archive, Report 2007/271, `http://eprint.iacr.org/`.

[3] S. Hirose, J. H. Park, and A. Yun, "A simple variant of the Merkle-Damgård scheme with a permutation," Advances in Cryptology - ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833, pp. 113–129, 2007.

[4] S. Indesteege, F. Mendel, B. Preneel, C. Rechberger, and V. Rijmen, "Collisions and other non-random properties for step-reduced SHA-256," Cryptology ePrint Archive, Report 2008/131, 2008. `http://eprint.iacr.org/`.

[5] F. Mendel, N. Pramstaller, C. Rechberger, and V. Rijmen, "Analysis of step-reduced SHA-256," Fast Software Encryption, FSE 2006, Lecture Notes in Computer Science, vol. 4047, pp. 126–143, 2006.

[6] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publication 180-2, August 2002. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`.

[7] S. K. Sanadhya and P. Sarkar, "Non-linear reduced round attacks against SHA-2 hash family," Cryptology ePrint Archive, Report 2008/174, 2008. `http://eprint.iacr.org/`.