Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 101907, 6 pages
http://dx.doi.org/10.1155/2013/101907

Research Article

Security Analysis of HMAC/NMAC by Using Fault Injection

Kitae Jeong,[1] Yuseop Lee,[1] Jaechul Sung,[2] and Seokhie Hong[1]

[1] *Center for Information Security Technologies (CIST), Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Republic of Korea*
[2] *Department of Mathematics, University of Seoul, Jeonnong-dong, Dongdaemun-gu, Seoul 130-743, Republic of Korea*

Correspondence should be addressed to Jaechul Sung; jcsung@uos.ac.kr

Received 18 July 2013; Accepted 22 August 2013

Academic Editor: Jongsung Kim

In Choukri and Tunstall (2005), the authors showed that if they decreased the number of rounds in AES by injecting faults, it is possible to recover the secret key. In this paper, we propose fault injection attacks on HMAC/NMAC by applying the main idea of their attack. These attacks are applicable to HMAC/NMAC based on the MD-family hash functions and can recover the secret key with the negligible computational complexity. Particularly, these results on HMAC/NMAC-SHA-2 are the first known key recovery attacks so far.

1. Introduction

HMAC and NMAC are hash-based message authentication codes proposed in [1]. The construction of HMAC/NMAC is based on a keyed hash function. Let H be an iterated Merkle-Damgård hash function, which defines a keyed hash function H_K by replacing IV with the key K. Then, HMAC and NMAC can be defined as follows:

$$\begin{gathered}\mathrm{HMAC}_K(M) = H\left(\overline{K} \oplus \mathrm{opad} \,\|\, H\left(\overline{K} \oplus \mathrm{ipad} \,\|\, M\right)\right), \\ \mathrm{NMAC}_{K_1,K_2}(M) = H_{K_1}\left(H_{K_2}(M)\right).\end{gathered} \tag{1}$$

Here, M is a message and K and (K_1, K_2) are the secret keys of HMAC and NMAC, respectively; $\overline{K}$ means K padded to a single block, and opad$(= \texttt{0x5c5c}\cdots)$ and ipad$(= \texttt{0x3636}\cdots)$ are two one-block length constants. Until now, many theoretical cryptanalytic results on HMAC/NMAC have been proposed [2–4]. For example, Wang et al. presented key recovery attacks on HMAC/NMAC-MD4 with 2^{72} MAC queries and 2^{77} MD4 computations [4]. On the other hand, McEvoy et al. introduced a differential power analysis on HMAC-SHA-256 in [5]. This attack does not allow the recovery of the secret key, but rather a secret intermediate hash value of SHA-256. It leads to forging the MACs for arbitrary messages. Correlation power analysis on HMAC based on six SHA-3 candidates was presented in [6]. This is also a forgery attack. To our knowledge, there is no key recovery attack on HMAC by using side channel analyss.

Side channel analysis exploits the easily accessible information such as power consumption, running time, and input-output behavior under malfunctions. It is often much more powerful than the classical cryptanalysis such as differential cryptanalysis and linear cryptanalysis. Since Kocher had introduced timing attacks in [7], many side channel analyses such as differential fault analysis [8] and fault injection attack [9] have been proposed [10–12].

Choukri and Tunstall proposed a fault injection attack on AES [13]. The fault injection method used a transient glitch on the power supplied to the smart card. In general, the implementation of a symmetric cryptographic algorithm in the PIC assembly language will have the following format:

```
movlw     0Ah
movwf     RoudCounter
RoudLabel
Call      RoudFunction
decfz     RoudCounter
goto      RoudLabel
```
(2)

The RAM variable (`RoudCounter`) is set to the number of rounds required (in the case of AES, `0A` in hexadecimal). The round function is executed, which has been represented by a call to the function `RoudFunction`. The `RoudCounter`

variable is then decremented, and the round is repeated until `RoudCounter` is equal to zero, at which the loop point exits. It is this loop that we are trying to change so that it exits earlier than expected. The target of the fault is the `decfz` step, which consists of a decrement, a test, and a conditional jump. The conditional jump is presented as a jump of one instruction when the test is positive; otherwise, the next instruction is executed. The aim of the attack is to reduce the algorithm to one round. It is not possible to remove the first round entirely as the first conditional test is after the first round. Thus, the cryptanalysis of the resulting algorithm can be simple and only requires two plaintext/ciphertext pairs.

In this paper, we propose fault injection attacks on HMAC/NMAC. Our fault assumption is based on that of [13]. That is, it is assumed that we can decrease the number of steps in the target compression function by injecting some faults. Our attack can be applied to HMAC/NMAC based on the MD-family hash functions and recover the secret key with the negligible computational complexity. As concrete examples, we apply our attack to HMAC/NMAC based on MD4, MD5 and SHA-2. Our attack results are summarized in Table 1. In the case of HMAC-SHA-256, for any message, we can recover the n-word secret key with $[n/3]$ fault injections and only a negligible computational complexity. Also, we need only $2 \cdot [n/3]$ fault injections to recover the $2n$-word secret key of NMAC-SHA-256. Thus, when n is 4, that is, the 4(8)-word secret key, we require just two(four) fault injections to recover the secret key of HMAC-SHA-256 (NMAC-SHA-256), respectively. Note that the attack results on HMAC/NMAC-SHA-2 are the first known key recovery attacks on them.

This paper is organized as follows: in Section 2, we briefly introduce the MD-family hash functions. Then, we describe the fault injection attacks on HMAC and NMAC in mbox-Sections 3 and 4, respectively. Finally, we give a conclusion in Section 5.

Table 1: Our attack results on HMAC/NMAC.

Algorithm	No. of injected faults	Algorithm	No. of injected faults
HMAC-MD4	$[n/3]$	NMAC-MD4	$2 \cdot [n/3]$
HMAC-MD5	$[n/3]$	NMAC-MD5	$2 \cdot [n/3]$
HMAC-SHA-224	$[n/3]$	NMAC-SHA-224	$[n/3] + [n/2]$
HMAC-SHA-256	$[n/3]$	NMAC-SHA-256	$2 \cdot [n/3]$
HMAC-SHA-384	$[n/3]$	NMAC-SHA-384	$[n/3] + n$
HMAC-SHA-512	$[n/3]$	NMAC-SHA-512	$2 \cdot [n/3]$

Table 2: Parameters of MD4, MD5 and SHA-2.

Hash function	Message block	Chaining value	Hash value	Step function	Word size
MD4	512 bits	128 bits	128 bits	48 steps	32 bits
MD5	512 bits	128 bits	128 bits	64 steps	32 bits
SHA-224	512 bits	256 bits	224 bits	64 steps	32 bits
SHA-256	512 bits	256 bits	256 bits	64 steps	32 bits
SHA-384	1024 bits	512 bits	384 bits	80 steps	64 bits
SHA-512	1024 bits	512 bits	512 bits	80 steps	64 bits

2. MD-Family Hash Function

Since MD4 [14] had been introduced in 1990, the MD-family hash functions such as MD5 [15] and SHA-2 [16], where the design rationale is based on that of MD4, have been proposed. To compute the hash value for a message M of any size, the MD-family hash functions divide M into message blocks $(M_0, \ldots, M_t)$ of fixed length b and obtain the hash value by using a compression function f. A compression function f takes a b-bit message string M_{i-1} and a s-bit chaining variable IHV_{i-1} as input values and outputs an updated s-bit chaining variable IHV_i. IHV_i is computed by iteratively using a step function. It consists of addition, Boolean function, and rotation operations. After operating a step function repeatedly, IHV_i is updated by adding IHV_{i-1}. Table 2 presents the parameters of MD4, MD5, and SHA-2.

As a concrete example, we briefly introduce SHA-2 (SHA-224, SHA-256, SHA-384, and SHA-512), one of the most important MD-family hash functions. In SHA-224/256, the word size is 32 bits. The message string is firstly padded to be a 512-bit multiple and is divided into 512-bit blocks. A compression function f takes a 512-bit message string and a 256-bit chaining variable as input values and outputs an updated 256-bit chaining variable. It consists of a message expansion and a data processing. The message block is expanded by using the following message expansion function. Here, $(m_0, \ldots, m_{15}) = M_i$, "+" denotes the wordwise addition, $\sigma_0(X) = (X^{\ggg 7}) \oplus (X^{\ggg 18}) \oplus (X^{\gg 3})$, and $\sigma_1(X) = (X^{\ggg 17}) \oplus (X^{\ggg 19}) \oplus (X^{\gg 10})$.

Consider

$$\begin{aligned} W_j &= m_j, \quad (0 \le j < 16), \\ W_j &= \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}, \\ &\qquad (16 \le j < 80). \end{aligned} \tag{3}$$

The data processing computes IHV_i as follows. Here, V_j denotes a 256-bit value consisting of eight words A_j, B_j, C_j, D_j, E_j, F_j, G_j, and H_j.

Consider

$$\begin{aligned} V_0 &= \mathrm{IHV}_{i-1}, \\ V_{j+1} &= R_j(V_j, W_j), \quad (j = 0, \ldots, 63), \\ \mathrm{IHV}_i &= \mathrm{IHV}_{i-1} + V_{64}. \end{aligned} \tag{4}$$

A step function R_j is defined as follows. Here, K_j is a constant number for each step; $\mathrm{Ch}(X, Y, Z) = (X \vee Y) \oplus (\neg X \vee Z)$,

$\mathrm{Maj}(X,Y,Z) = (X \vee Y) \oplus (X \vee Z) \oplus (Y \vee Z)$, $\Sigma_0(X) = (X^{\gg 2}) \oplus (X^{\gg 13}) \oplus (X^{\gg 22})$, and $\Sigma_1(X) = (X^{\gg 6}) \oplus (X^{\gg 11}) \oplus (X^{\gg 25})$.

$$\begin{aligned} T_1^j &= H_j + \Sigma_1\left(E_j\right) + \mathrm{Ch}\left(E_j, F_j, G_j\right) + K_j + W_j, \\ T_2^j &= \Sigma_0\left(A_j\right) + \mathrm{Maj}\left(A_j, B_j, C_j\right), \\ A_{j+1} &= T_1^j + T_2^j, \qquad B_{j+1} = A_j, \\ C_{j+1} &= B_j, \qquad D_{j+1} = C_j, \\ E_{j+1} &= D_j + T_1^j, \qquad F_{j+1} = E_j, \\ G_{j+1} &= F_j, \qquad H_{j+1} = G_j. \end{aligned} \tag{5}$$

SHA-224 outputs the left most 224-bit value of IHV_{t+1} as the hash value, and SHA-256 outputs IHV_{t+1} as the hash value.

The structures of SHA-384/512 are similar to those of SHA-224/256. In SHA-384/512, the word size is double that of SHA-224/256. Thus, a message block is 1024 bits, and the size of a chaining value is 512 bits. A compression function consists of 80 steps.

3. Key Recovery Attack on HMAC

Our attack can be applied to HMAC based on the MD-family hash functions. As a concrete example, we introduce a key recovery attack on HMAC-SHA-2. Other cases can be explained similarly. For the detailed attack results, see Table 1.

3.1. Fault Assumption. Recall that the authors reduced the number of rounds in AES to one in [13]. We apply this fault assumption to HMAC. That is, by using several fault injections, we can reduce the number of steps in the last two compression functions (see Figure 1). Similarly to AES, the MD-family hash functions compute the hash value by iteratively using a step function. Moreover, there are some results based on similar fault models [17, 18]. Thus, our fault assumption is reasonable.

For the simplicity, we denote these two compression functions by f_0^* and f_1^*. Thus, we reduce the number of steps in (f_0^*, f_1^*) to some values by using fault injections, respectively, and then recover the secret key K of HMAC-SHA-2. When we reduced the number of steps in f_i^* to j, we denoted this event by $f_{i,j}^*$ in this paper.

3.2. Key Recovery Attack on HMAC-SHA-256/512. As mentioned in the previous section, the structure of SHA-512 is similar to that of SHA-256 excluding parameters such as the word size. Thus, we only propose a key recovery attack on HMAC-SHA-256 in this subsection.

Since the word size of SHA-256 is 32 bits, we assume that the length of $K(= K_0 \| K_1 \| \cdots \| K_{n-1})$ is $32 \cdot n$ bits. Our attack on HMAC-SHA-256 conducts the procedure recovering 96-bit $(K_{3i}, K_{3i+1}, K_{3i+2})$ iteratively $[n/3]$ times $(i = 0, \ldots, [(n-1)/3])$. We can recover (K_0, K_1, K_2) as follows. From an event $(f_{0,3}^*, f_{1,1}^*)$, we compute $\mathrm{HMAC}(= \mathrm{HMAC}_0 \| \mathrm{HMAC}_1 \| \cdots \| \mathrm{HMAC}_7)$. Then, we can construct

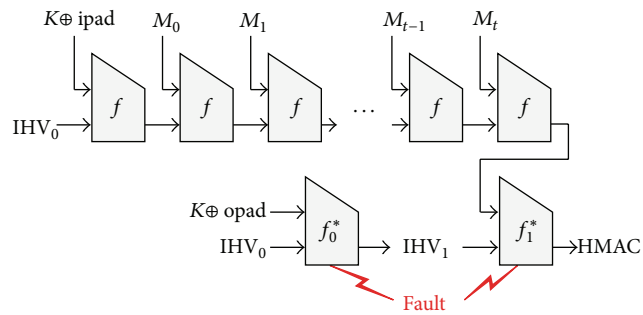

Figure 1: Our fault model on HMAC.

the following six equations (see Figure 2). Here, $(A, \ldots, H) = (\mathrm{IHV}_{0,0}, \ldots, \mathrm{IHV}_{0,7})$;

$$\begin{aligned} (Y + Z) + (A + B) &= \mathrm{HMAC}_1, \\ (X + Y) + (B + C) &= \mathrm{HMAC}_2, \\ X + (A + B + D) &= \mathrm{HMAC}_3, \\ (\beta + \gamma) + (E + F) &= \mathrm{HMAC}_5, \\ (\alpha + \beta) + (F + G) &= \mathrm{HMAC}_6, \\ \alpha + (E + G + H) &= \mathrm{HMAC}_7. \end{aligned} \tag{6}$$

Since $(A, \ldots, H)$ are known values in (6), we can obtain $(X, Y, Z, \alpha, \beta, \gamma)$. With these values, we can compute (K_0, K_1, K_2) by using the following equations:

$$\begin{aligned} K_0 \oplus \mathtt{0x5c} &= X - \left(H + \Sigma_1(E) + \mathrm{Ch}(E, F, G) + C_0 \right. \\ &\qquad \left. + \Sigma_0(A) + \mathrm{Maj}(A, B, C)\right), \\ K_1 \oplus \mathtt{0x5c} &= Y - \left(G + \Sigma_1(\alpha) + \mathrm{Ch}(\alpha, E, F) + C_1 \right. \\ &\qquad \left. + \Sigma_0(X) + \mathrm{Maj}(X, A, B)\right), \\ K_2 \oplus \mathtt{0x5c} &= Z - \left(F + \Sigma_1(\beta) + \mathrm{Ch}(\beta, \alpha, E) + C_2 \right. \\ &\qquad \left. + \Sigma_0(Y) + \mathrm{Maj}(Y, X, A)\right). \end{aligned} \tag{7}$$

By repeating the previous procedure, we can recover K by using $[n/3]$ fault injections. Since this consists of only solving simple equations, the computational complexity is negligible.

3.3. Key Recovery Attack on HMAC-SHA-224/384. Recall that SHA-224/384 outputs the left most 224/384 bits of the resulting 256/512-bit hash value as the hash value, respectively. For example, HMAC-SHA-224 outputs only $\mathrm{HMAC}_0 \| \cdots \| \mathrm{HMAC}_6$ as the hash value. Thus, we can not compute (α, β) in (6) (see Figure 2). However, we can obtain (X, Y, Z) in (6) and compute K_0 in (7). And then, we can obtain α by using K_0. By repeating this procedure, we can recover (K_1, K_2) sequentially. Hence, we can recover K of HMAC-SHA-224/384 with $[n/3]$ fault injections and the negligible computational complexity.

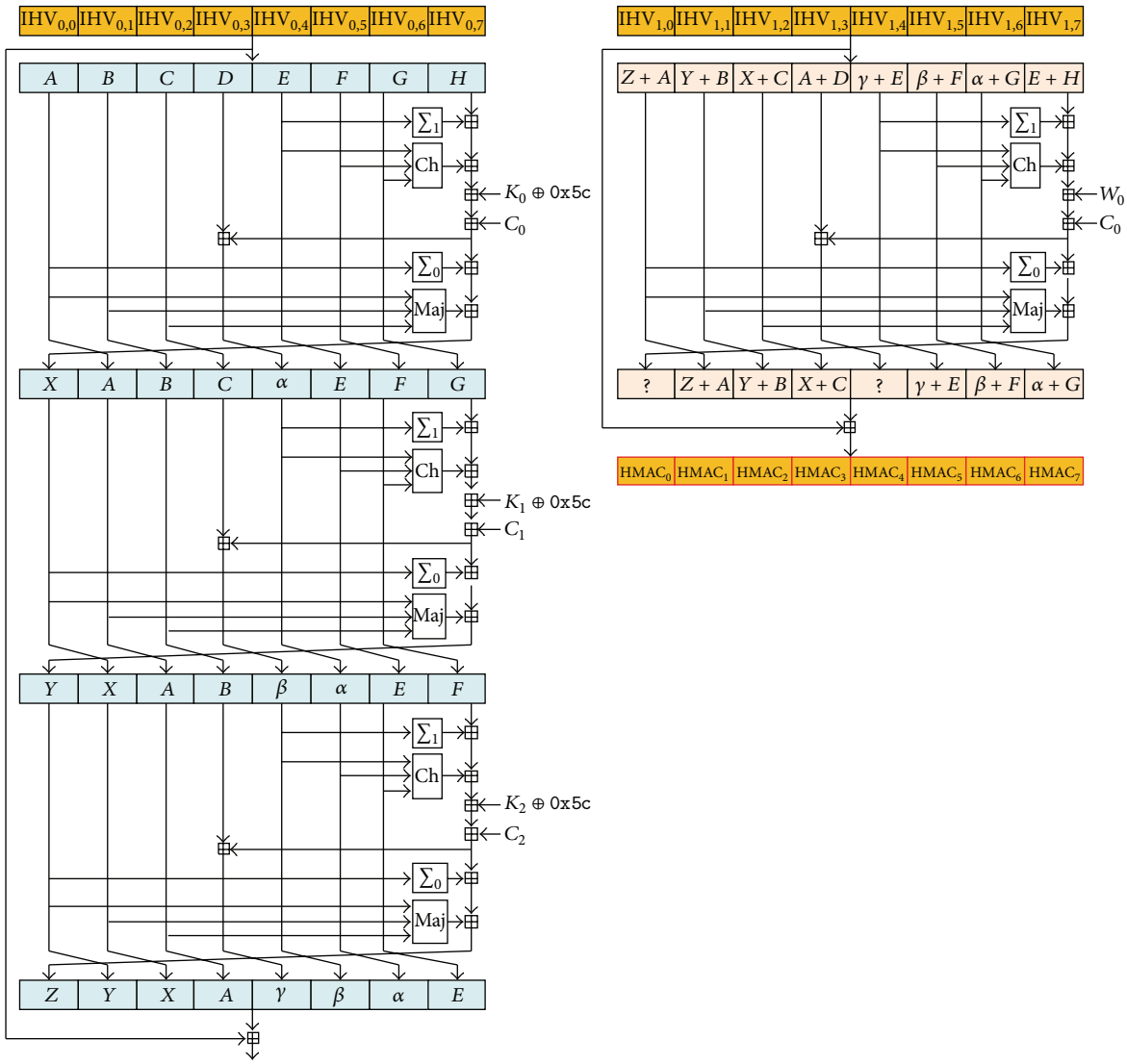

FIGURE 2: An event $(f^*_{0,3}, f^*_{1,1})$.

4. Key Recovery Attack on NMAC

A key recovery attack on NMAC is similar to that on HMAC. This is also applicable to NMAC based on the MD-family hash functions. As a concrete example, we present a key recovery attack on NMAC-SHA-2. In the case of other MD-family hash functions, we can attack in a similar fashion. Table 1 gives the detailed results.

4.1. Fault Assumption. Differently from HMAC, NMAC uses two n-word secret keys (K_1, K_2) (see Figure 3). Thus, our attack on NMAC consists of the following two steps. Firstly, we recover K_2 by using a key recovery attack on HMAC-SHA-2 (Fault$_1$ in Figure 3). Secondly, to compute K_1, we inject faults to (f^*_2, f^*_3, f^*_1) (Fault$_2$ in Figure 3). Note that we assume that a message M is only a single block.

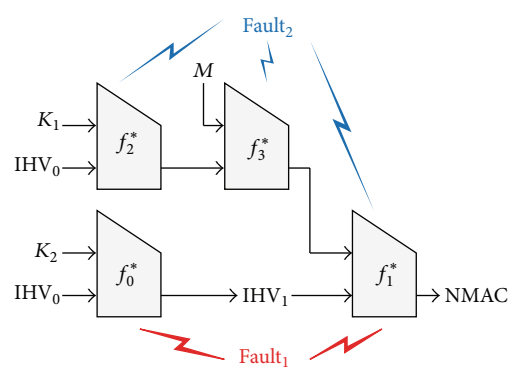

FIGURE 3: Our fault model on NMAC.

4.2. Key Recovery Attack on NMAC-SHA-256/512. Since the structure of SHA-512 is similar to that of SHA-256, we only introduce a key recovery attack on NMAC-SHA-256 in this subsection. We assume that the length of $(K_1\ (= K_{1,0}\|\cdots\|K_{1,n-1}),\ K_2(= K_{2,0}\|\cdots\|K_{2,n-1}))$ is $32 \cdot n$ bits, respectively. By using a key recovery attack on HMAC-SHA-256, we firstly recover K_2 with $\lceil n/3 \rceil$ fault injections and the negligible

Table 3: Recovery of $(K_{1,0}, K_{1,1}, K_{1,2})$.

	Chaining value	Message
$f^*_{2,3}$	(a, b, c, d, e, f, g, h)	$K_{1,0}$
	$(X, a, b, c, ?, e, f, g)$	$K_{1,1}$
	$(Y, X, a, b, ?, ?, e, f)$	$K_{1,2}$
	$(Z, Y, X, a, ?, ?, ?, e)$	—
	$(Z + a, Y + b, X + c, a + d, ?, ?, ?, e + h)$	feed-forward
$f^*_{3,1}$	$(Z + a, Y + b, X + c, a + d, ?, ?, ?, e + h)$	M
	$(W, Z + a, Y + b, X + c, ?, ?, ?, ?)$	—
	$(W + Z + a, Y + Z + a + b, X + Y + b + c, X + a + c + d, ?, ?, ?, ?)$	feed-forward
$f^*_{1,4}$	(A, B, C, D, E, F, G, H)	$W + Z + a$
	$(\alpha_1, A, B, C, \beta_1, E, F, G)$	$Y+Z+a+b$
	$(\alpha_2, \alpha_1, A, B, \beta_2, \beta_1, E, F)$	$X+Y+b+c$
	$(\alpha_3, \alpha_2, \alpha_1, A, \beta_3, \beta_2, \beta_1, E)$	$X+a+c+d$
	$(\alpha_4, \alpha_3, \alpha_2, \alpha_1, \beta_4, \beta_3, \beta_2, \beta_1)$	—
	$(\alpha_4 + A, \alpha_3 + B, \alpha_2 + C, \alpha_1 + D, \beta_4 + E, \beta_3 + F, \beta_2 + G, \beta_1 + H)$	feed-forward

computational complexity. And then, we compute K_1 from an event $(f^*_{2,i}, f^*_{3,j}, f^*_{1,k})$.

Table 3 shows the results of an event $(f^*_{2,3}, f^*_{3,1}, f^*_{1,4})$. From Table 3, we can compute $(K_{1,0}, K_{1,1}, K_{1,2})$ as follows. Since we know HMAC and can compute $\mathrm{IHV}_1(= A\|B\|\cdots\|H)$ by using K_2, we can compute $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3, \beta_4)$ by using the following equations:

$$\begin{aligned} \alpha_1 &= \mathrm{HMAC}_3 - D, & \beta_1 &= \mathrm{HMAC}_7 - H, \\ \alpha_2 &= \mathrm{HMAC}_2 - C, & \beta_2 &= \mathrm{HMAC}_6 - G, \\ \alpha_3 &= \mathrm{HMAC}_1 - B, & \beta_3 &= \mathrm{HMAC}_5 - F, \\ \alpha_4 &= \mathrm{HMAC}_0 - A, & \beta_4 &= \mathrm{HMAC}_4 - E. \end{aligned} \tag{8}$$

By using $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3, \beta_4)$, we can compute the messages of $f^*_1(W + Z + a,\ Y + Z + a + b,\ X + Y + b + c,\ X + a + c + d)$ easily. Since we know $\mathrm{IHV}_0(= a\|b\|\cdots\|h)$, we can obtain (X, Y, Z). Thus, we can recover $(K_{1,0}, K_{1,1}, K_{1,2})$ similarly to a key recovery attack on HMAC-SHA-256.

By repeating the previous procedure, we can recover (K_1, K_2) by using $2 \cdot [n/3]$ fault injections. Its computational complexity is also negligible.

4.3. Key Recovery Attack on NMAC-SHA-224/384. Since SHA-224/384 outputs the left most 224/384 bits of the resulting 256/512-bit hash value as the hash value, respectively, we do not know (β_1, β_2) (see Table 3). In this case, we can not compute K_1. Thus, we consider different events on these algorithms. To recover $(K_{1,0}, K_{1,1})$ of NMAC-SHA-224, we use an event $(f^*_{2,2}, f^*_{3,1}, f^*_{1,3})$. This attack needs $[n/3] + [n/2]$ fault injections and the negligible computational complexity. In the case of NMAC-SHA-384, we consider an event $(f^*_{2,1}, f^*_{3,1}, f^*_{1,2})$ to recover $K_{1,0}$. This attack requires $[n/3] + n$ fault injections with a negligible computational complexity.

5. Conclusion

In this paper, we proposed key recovery attacks on HMAC/NAC by using a fault injection attack. Our attack can be applied to HMAC/NMAC based on the MD-family hash functions and requires a small number of fault injections with a negligible computational complexity. As concrete examples, we applied our attack to HMAC/NMAC based on MD4, MD5, and SHA-2. The results on HMAC/NMAC-SHA-2 are the first known key recovery attacks on them.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the C-ITRC (Convergence Information Technology Research Center) support program (NIPA-2013-H0301-13-3007) supervised by the NIPA (National IT Industry Promotion Agency).

References

[1] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proceedings of the Annual International Cryptology Conference (CRYPTO '96)*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 10–15, August 1996.

[2] S. Contini and Y. L. Yin, "Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions," in *Advances in Cryptology*, vol. 4284 of *Lecture Notes in Computer Science*, pp. 37–53, 2006.

[3] P. A. Fouque, G. Leurent, and P. Q. Nguyen, "Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5," in *Advances in Cryptology*, vol. 4622 of *Lecture Notes in Computer Science*, pp. 13–30, 2007.

[4] L. Wang, K. Ohta, and N. Kunihiro, "New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5," in *Advances in Cryptology (EUROCRYPT 2008)*, vol. 4965 of *Lecture Notes in Computer Science*, pp. 237–253, 2008.

[5] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane, "Differential power analysis of HMAC based on SHA-2, and countermeasures," in *Advances in Cryptology*, vol. 4867 of *Lecture Notes in Computer Science*, pp. 317–332, 2007.

[6] O. Benoît and T. Peyrin, "Side-channel analysis of six sha-3 candidates," in *Advances in Cryptology*, vol. 6225 of *Lecture Notes in Computer Science*, pp. 140–157, 2010.

[7] P. Kocher, "Timing attacks on implementation of Diffie-Hellman," in *Advances in Cryptology*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113, 1996.

[8] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 513–525, 1997.

[9] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, 1997.

[10] K. Jeong, Y. Lee, J. Sung, and S. Hong, "Differential fault analysis on block cipher SEED," *Mathematical and Computer Modelling*, vol. 55, no. 1-2, pp. 26–34, 2012.

[11] K. Jeong and C. Lee, "Differential fault analysis on block cipher LED-64," in *Future Information Technology, Application, and Service*, vol. 164 of *Lecture Notes in Electrical Engineering*, pp. 747–755, 2012.

[12] K. Jeong, J. Sung, S. Hong, and C. Lee, "A new approach of differential fault analysis on block ciphers with S-box," *Information*, vol. 16, no. 3, pp. 1915–1928, 2013.

[13] H. Choukri and M. Tunstall, "Round reduction using faults," in *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '05)*, pp. 13–24, 2005.

[14] R. Rivest, "The MD4 message digest algorithm," RFC 1320, 1992.

[15] R. Rivest, "The MD5 message digest algorithm," RFC 1321, 1992.

[16] National Institute of Standards and Technology, FIPS PUB 180-2: Secure Hash Standard, 2002.

[17] C. H. Kim and J. J. Quisquater, "Fault attacks for CRT based RSA: new attacks, new results, and new countermeasures," in *Proceedings of the Workshop on Information Security Theory and Practices*, vol. 4462 of *Lecture Notes in Computer Science*, pp. 215–228, May 2007.

[18] J. G. J. Van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in *Proceedings of the 8th International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '11)*, pp. 91–99, September 2011.