Research Article

Security Analysis of HMAC/NMAC by Using Fault Injection

Kitae Jeong,¹ Yuseop Lee,¹ Jaechul Sung,² and Seokhie Hong¹

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

Copyright © 2013 Kitae Jeong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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:

\[ \text{HMAC}_K(M) = H\left(K \oplus \text{opad} \parallel H\left(K \oplus \text{ipad} \parallel M\right)\right) , \]
\[ \text{NMAC}_{K_1,K_2}(M) = H_{K_1}\left(H_{K_2}(M)\right) . \]

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 $\text{opad}(=0\times5c5c\cdots)$ and $\text{ipad}(=0\times3636\cdots)$ 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 analysis.

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
   Call RoudLabel
   decfz RoudCounter
   goto RoudLabel
```

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 RoundCounter 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 desc2 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 secret key of HMAC-SHA-256. Thus, when \(n = 4\), 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 sections 3 and 4, respectively. Finally, we give a conclusion in Section 5.

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 \((M_0, \ldots, M_l)\) 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 is computed by iteratively using a step function. It consists of addition, Boolean function, and rotation operations. After operating a step function repeatedly, IHV 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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. of injected faults</th>
<th>Algorithm</th>
<th>No. of injected faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC-MD4</td>
<td>([n/3])</td>
<td>HMAC-MD4</td>
<td>(2 \cdot [n/3])</td>
</tr>
<tr>
<td>HMAC-MD5</td>
<td>([n/3])</td>
<td>HMAC-MD5</td>
<td>(2 \cdot [n/3])</td>
</tr>
<tr>
<td>HMAC-SHA-224</td>
<td>([n/3])</td>
<td>HMAC-SHA-224</td>
<td>([n/3]) + ([n/2])</td>
</tr>
<tr>
<td>HMAC-SHA-256</td>
<td>([n/3])</td>
<td>HMAC-SHA-256</td>
<td>([n/3]) + ([n/2])</td>
</tr>
<tr>
<td>HMAC-SHA-384</td>
<td>([n/3])</td>
<td>HMAC-SHA-384</td>
<td>([n/3]) + ([n/2])</td>
</tr>
<tr>
<td>HMAC-SHA-512</td>
<td>([n/3])</td>
<td>HMAC-SHA-512</td>
<td>([n/3]) + ([n/3])</td>
</tr>
</tbody>
</table>

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, \((m_0, \ldots, m_{15}) = M_i\), “\(\oplus\)” denotes the wordwise addition, \(\sigma_0(X) = (X_{267} \oplus X_{198}) \oplus (X_{123})\), and \(\sigma_1(X) = (X_{197}) \oplus (X_{198} \oplus X_{123})\).

Consider

\[
W_j = m_j, \quad (0 \leq j < 16),
\]

\[
W_j = σ_1(W_{j-2}) + W_{j-7} + σ_0(W_{j-15}) + W_{j-16}, \quad (16 \leq j < 80).
\]

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

Consider

\[
V_0 = IHV_{i-1},
\]

\[
V_{j+1} = R_j(V_j, W_j), \quad (j = 0, \ldots, 63),
\]

\[
IHV_i = IHV_{i-1} + V_{64}.
\]

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

<table>
<thead>
<tr>
<th>Hash function</th>
<th>Message block</th>
<th>Chaining value</th>
<th>Hash value</th>
<th>Step function</th>
<th>Word size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD4</td>
<td>512 bits</td>
<td>128 bits</td>
<td>128 bits</td>
<td>48 steps</td>
<td>32 bits</td>
</tr>
<tr>
<td>MD5</td>
<td>512 bits</td>
<td>128 bits</td>
<td>128 bits</td>
<td>64 steps</td>
<td>32 bits</td>
</tr>
<tr>
<td>SHA-224</td>
<td>512 bits</td>
<td>256 bits</td>
<td>224 bits</td>
<td>64 steps</td>
<td>32 bits</td>
</tr>
<tr>
<td>SHA-256</td>
<td>512 bits</td>
<td>256 bits</td>
<td>256 bits</td>
<td>64 steps</td>
<td>32 bits</td>
</tr>
<tr>
<td>SHA-384</td>
<td>1024 bits</td>
<td>512 bits</td>
<td>384 bits</td>
<td>80 steps</td>
<td>64 bits</td>
</tr>
<tr>
<td>SHA-512</td>
<td>1024 bits</td>
<td>512 bits</td>
<td>512 bits</td>
<td>80 steps</td>
<td>64 bits</td>
</tr>
</tbody>
</table>
Maj(𝑋,𝑌,𝑍) = (𝑋 ∨ 𝑌) ⊕ (𝑋 ∨ 𝑍) ⊕ (𝑌 ∨ 𝑍), Σ0(𝑋) = (𝑋 ⊙ 2) ⊕ (𝑋 ⊙ 13) ⊕ (𝑋 ⊙ 22), and Σ1(𝑋) = (𝑋 ⊙ 6) ⊕ (𝑋 ⊙ 11) ⊕ (𝑋 ⊙ 25).

SHA-224 outputs the left most 224-bit value of IHV+t, as the hash value, and SHA-256 outputs IHV+t, 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 fault 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∗ 1 to 𝑗, we denoted this event by 𝑓∗1 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 = 𝐾0|𝐾1|⋯|𝐾𝑛−1) is 32 · 𝑛 bits. Our attack on HMAC-SHA-256 conducts the procedure recovering 96-bit (𝐾3, 𝐾13, 𝐾31, 𝐾31, 2) iteratively [𝑛/3] times (𝑖 = 0, …, [(𝑛−1)/3]). We can recover (𝐾0, 𝐾1, 𝐾2) as follows. From an event (𝑓∗0, 𝑓∗1), we compute HMAC(= HMAC0||HMAC1|⋯|HMAC7). Then, we can construct the following six equations (see Figure 2). Here, (𝐴, …, 𝐻) = (IHV0, 0, …, IHV0, 7):

(𝑌 + 𝑍) + (𝐴 + 𝑃) = HMAC1, (𝑋 + 𝑌) + (𝐵 + 𝐶) = HMAC2, 𝑋 + (𝐴 + 𝐵 + 𝐷) = HMAC3, (𝛽 + 𝐸) + (𝐹 + 𝐺) = HMAC4, (𝛼 + 𝑃) + (𝐹 + 𝐺 + 𝐻) = HMAC5.

Since (𝐴, …, 𝐻) are known values in (6), we can obtain (𝑌, 𝑍, 𝑋, 𝛽, 𝛼, 𝛾). With these values, we can compute (𝐾0, 𝐾1, 𝐾2) by using the following equations:

(𝑌 + 𝑋) + (𝐴 + 𝑃) = HMAC1, (𝑋 + 𝑌) + (𝐵 + 𝐶) = HMAC2, 𝑋 + (𝐴 + 𝐵 + 𝐷) = HMAC3, (𝛽 + 𝐸) + (𝐹 + 𝐺) = HMAC4, (𝛼 + 𝑃) + (𝐹 + 𝐺 + 𝐻) = HMAC5.

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 HMAC0|⋯|HMAC7 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 𝐾0 in (7). And then, we can obtain 𝛼 by using 𝐾0. By repeating this procedure, we can recover (𝐾1, 𝐾2) sequentially. Hence, we can recover 𝐾 of HMAC-SHA-224/384 with [𝑛/3] fault injections and the negligible computational complexity.

![Figure 1: Our fault model on HMAC.](image-url)
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 $(f_3^*, f_3', f_2^*)$ (Fault $2$ in Figure 3). Note that we assume that a message $M$ is only a single block.

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
computational complexity. And then, we compute $K_1$ from an event $(f_{2,3}^*, f_{3,1}^*, f_{4,1}^*)$.

Table 3 shows the results of an event $(f_{2,3}^*, f_{3,1}^*, f_{4,1}^*)$. From Table 3, we can compute $(K_{1,0}, K_{1,1}, K_{1,2})$ as follows. Since we know HMAC and can compute 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:

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

By using $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3, \beta_4)$, we can compute the messages of $f_{1}^*((W + Z + a, Y + Z + a + b, X + Y + b + c, X + a + c + d)\cdots)$ easily. Since we know IHV$_2(= a||B||\cdots||H)$, 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 $(\beta_1, \beta_2)$ (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 $(f_{2,2}^*, f_{3,1}^*, f_{4,1}^*)$. 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 $(f_{2,3}^*, f_{3,1}^*, f_{4,1}^*)$ 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


Submit your manuscripts at http://www.hindawi.com