A First-Order Differential Power Analysis Attack on HMAC-SM3

Limin Guo, Lihui Wang, Qing Li, Zhimin Zhang, Dan Liu, Weijun Shan,
Shanghai Fudan Microelectronics Group Company Limited, Shanghai, 200433, China
E-mail: guolimin@fmsh.com.cn

Abstract—**HMAC algorithm is one of the most famous keyed hash functions, and widely utilized. And SM3 is the only standard hash algorithm of China. However, most cryptographic algorithms implementations are vulnerable against side channel attacks. But specific side channel attacks on HMAC-SM3 have not been given so far. This paper presents a first-order DPA attack on HMAC-SM3. HMAC-SM3 hash algorithm is based on the mixing of different algebraic operations, such as XOR and addition modulo 2^{32}, thus the proposed DPA attack is mainly against these basic group operations. Experimental results are given by attacking an implementation of HMAC-SM3 in a smart card, which demonstrate the practicability of such attacks described in this paper.**

Keywords—HMAC;SM3; DPA;

I. INTRODUCTION

Various cryptographic algorithm are prevalent applied in economy, military, government and so on to provide information security. Currently, many attack methods are widely used to cryptographic equipment to recover the secret key. Attacks on cryptographic equipment are usually classified into four categories: logical attacks where the attackers exploits a weak software implementation; invasive attacks where the device is physically and irreversibly modified; side channel attacks like SPA, DPA and CPA; and fault attacks where external perturbations are used to induce faults on the execution of a given sensitive program or on the sensitive data being manipulated. Many cryptanalysts have focused on side channel attacks, since they are simple to implement and require rather low cost equipment. The side-channel information which leaks the secret key includes the executing time of the algorithm, the power consumption of the device, or even the electromagnetic radiation, faulty output, and so on. Based on various side-channel information, power analysis is the most commonly used attack methods.

HMAC (Hash-based Message Authentication code) is one of the most famous keyed hash functions, and widely utilized. HMAC is based on a cryptographic hash function such as SM3. SM3 [1] hash algorithm is released by China's Office of Security Commercial Code Administration, and certificated as the only standard hash algorithm of China in 2010. The previous work are mainly about the different implementations of SM3 [2] [3] [4] [5]. Many side channel attacks on HMAC algorithm have been reported already. Lemke *et al.* [6] introduced a side channel attack against HMAC algorithm based on the hash function RIPEMD-160, and it may be applicable to HMAC-SHA-1. Okeya *et al.* [7] [8] evaluated the security of HMAC algorithm based on block-cipher based hash functions. McEvoy *et al.* [9] discussed a differential side-channel attack on an implementation of the HMAC algorithm that uses the SHA-2 hash function family. But as far as we know, the resistance of SM3 to side channel attacks still remains uncertain and thus a potential risk for software or hardware implementations. In this paper, we propose a first order DPA attack against HMAC-SM3. The rest of this paper is organized as follows. Section II introduces the background theory regarding the SM3 algorithm, the HMAC algorithm, and DPA attacks. Section III presents a DPA attack on HAMC-SM3. Attack results are given in Section IV. Finally, we conclude in Section V.

II. BACKGROUND

The following section will briefly cover the background theory, required for understanding this work. First, we give an overview of SM3 algorithm, HMAC algorithm and followed by a brief introduction to differential power analysis.

A. SM3 hash algorithm descryption

Full description of the SM3 hash algorithm can be found in the official OSCCA standard [1]. SM3 maps a message of length l ($l < 2^{64}$) bits to produce a message digest of 256 bits. The SM3 algorithm essentially consists of three stages: (i) message padding and parsing; (ii) message expansion; (iii)message compression.

Message Padding and Parsing. The binary message to be processed is appended with a single bit "1" followed by zeros until the padded message bit length equivalent to $448 \bmod 512$. The original message length is then appended as a 64-bit binary number. The resultant message is then parsed into n 512-bit blocks, denoted as $B^{(i)}$, for $0 \le i \le n-1$. These $B^{(i)}$ message blocks are iteratively passed to the message expansion stage.

Message Expansion. In this stage, each $B^{(i)}$ is expanded into 132 32-bit words W_j, for $0 \le j \le 67$ and W_j', for $0 \le j \le 63$. Each 512-bit $B^{(i)}$ block from the message padding and parsing stage is viewed as sixteen 32-bit words denoted as W_j, for $0 \le j \le 15$ and generate other 32-bit words according to equations given by:

$$W_j = P_1\left(W_{j-16} \oplus W_{j-9} \oplus \left(W_{j-3} \lll 15\right)\right) \oplus \left(W_{j-13} \lll 7 \oplus Wj-6, 16 \le j \le 67 \right. \quad (1)$$

$$W_j = W_j \oplus W_{j+4}, 0 \le j \le 63 \quad (2)$$

where $P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$ and $x \lll k$ denotes a circular rotations of x by k positions to the left.

Message Compression. The SM3 compression function *CF* utilizes eight 32-bit word registers labeled A, B, C, D, E, F, G, H, which are initialized to the 256-bit initial value at the beginning of each call to the hash function. Sixty-four iterations of following operations are sequentially utilized for computing the hash value:

$$SS1 \leftarrow \left((A \lll 12) + E + (T_j \lll j)\right) \lll 7 \tag{3}$$

$$SS2 \leftarrow SS1 \oplus (A \lll 12) \tag{4}$$

$$TT1 \leftarrow FF_j(A, B, C) + D + SS2 + W_j \tag{5}$$

$$TT2 \leftarrow GG_j(E, F, G) + H + SS1 + W_j \tag{6}$$

$$D \leftarrow C \tag{7}$$

$$C \leftarrow B \lll 9 \tag{8}$$

$$B \leftarrow A \tag{9}$$

$$A \leftarrow TT1 \tag{10}$$

$$H \leftarrow G \tag{11}$$

$$G \leftarrow F \lll 19 \tag{12}$$

$$F \leftarrow E \tag{13}$$

$$E \leftarrow P_0(TT2) \tag{14}$$

where T_j are constants defined as below:

$$T_j = \begin{cases} 79cc4519 & 0 \le j \le 15 \\ 7a879d8a & 16 \le j \le 63 \end{cases}$$

and FF_j, GG_j, P_0 are given by:

$$FF_j(X, Y, Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \land Y) \lor (X \land Z) \lor (Y \land Z) & 16 \le j \le 63 \end{cases}$$

$$GG_j(X, Y, Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \land Y) \lor (X \land Z) & 16 \le j \le 63 \end{cases}$$

$$P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$$

A 256-bit intermediate hash value $\mathrm{V}^{(i+1)}$ is calculated after 64 iterations:

$$V^{(i+1)} \leftarrow ABCDEFGH \oplus V^{(i)}$$

The SM3 algorithm iteratively processes all N data blocks, and computes a fixed length data, namely the final 256-bit message digest.

B. HMAC scheme

HMAC is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combinations with a secret cryptographic key. It may be used to simultaneously verify both the data integrity and the authentication of a message. In HMAC, for a given message m, using the Key *K*, the message authentication code is calculated as follows:

$$HMAC(m, K) = h\left((K \oplus opad) || h\left((K \oplus ipad) || m\right)\right)$$

where *h* is a cryptographic hash function, *K* is the secret key padded to the right with extra zeros to the input block size of the hash function, or the hash of the original key if it's longer than that block size, and m is the message to be authenticated. *opad* is the outer padding ($\mathrm{0x5C5C5C \cdots 5C5C}$, one-block-long hexadecimal constant), and *ipad* is the inner padding ($\mathrm{0x363636 \cdots 3636}$, one-block-long hexadecimal constant). ‖ denotes concatenation, $\oplus$ denotes exclusive or.

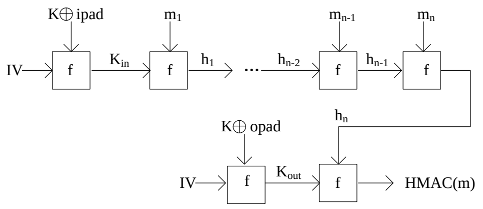

Fig. 1. HMAC

Fig.1. shows the algorithm of HMAC, where *f* stands for a compression function. In order to calculate $\mathrm{HMAC(m, K)}$, the hash function *h* must be invoked twice. At first, the message *m* is divided into the fixed-length blocks $\mathrm{m_1, \cdots, m_n}$. Then the compression function *f* is utilized $\mathrm{n + 3}$ times. $\mathrm{m_1, \cdots, m_n}$ are part of message inputs of the first application of the hash function *h*. The output h_n is inputted as a part of message input of the second application of the hash function *h*.

In this paper, we use SM3 to instantiate the hash function *h* and use the terminology "HMAC-SM3" to denote the HMAC algorithm that uses SM3 to instantiate *h*. We denote by *IV* the initial vector of SM3. Using the compression function *f*, K_{in} and K_{out} are defined as follows, where f stands for compression function *CF* here:

$$K_{in} = f(IV, K \oplus ipad), \quad K_{out} = f(IV, K \oplus opad)$$

In HMAC, K_{in} and K_{out} are fixed and unknown secret values. If an attacker know these two values, he can create MACs of his choice. Consequently, the goal of the attacker is to recover these two secret hash values, instead of the secret key *K* itself.

C. DPA

Because the amount of power used by a device is influenced by the data being processed, power consumption measurements contain information about calculations of a cryptographic implementation. An attacker can exploit the dependency between the power consumption of a device and the processed data in order to recover secret intermediates, such as keys of cryptographic algorithms. Differential Power Analysis (DPA) computes hypotheses of the power consumption for each input and key candidate and compares them to the recorded power consumption of the device. Such a hypothesis can be computed by calculating the Hamming weight (HW) of a processed value. Finding a suited intermediate value, which reveals information about the key, is specified as leakage analysis. In order to compare the calculated hypothesis to the measured power consumption, methods like the Pearson correlation or the difference in means can be used [10].

III. Attacking HMAC-SM3

In this section, we present an attack on HMAC-SM3 using DPA. We can only recover the intermediate state required for forging a HMAC, i.e. the inner keyed state and outer keyed state resulting from the digestion of the key XORed with the inner padding and outer padding. In this paper, we focus on power analysis, especially DPA. However, the attack is not limited to DPA, other side channel attacks, such as timing attack or electromagnetic analysis, are also applicable.

A. Goal of the attack

In order to explain the attacks, we will state the hypothesis functions used in the conducted DPAs. We assume m are public and changeable. The block size of SM3 is 512 bits, therefore, without loss of generality, we can assume that $|K|$ = $|ipad|$ = $|opad|$ = 512 and the size of the message m satisfies $|m|$ = 256. Meanwhile, on the first call of SM3, the device will run the compression function twice. We can launch a DPA attack on the second calculation when the variable m is introduced and combined with K_{in}. Considering that an attacker has access to the output of HMAC and there exists an XOR operation in the final compression function, a reverse DPA [7] can be utilized to recover K_{out}. Consequently, the goal of the attacker is to recover K_{in} and K_{out}, instead of the secret key itself.

B. Recovery of K_{in}

We use the subscript t, $0 \le t \le 63$ to signify the round number, e.g. A_0 refers to the value of A at the beginning of round 0 of the compression function *CF*, etc. At the very start of *CF*, eight 32-bit word registers A, B, C, D, E, F, G, H are initialized by the secret intermediate value K_{in}. Thus, the goal of this DPA attack is to recover the eight values $A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0$. The detailed attack process is described as follows.

1. According to equations (3) to (7), the variable $TT1_0$ must be calculated in round 0. $TT1_0$ is a sum with 4 items, and can be rewritten as:

$$TT1_0 = \theta_0 + W_0'$$

where

$$\theta_0 = FF_0(A_0, B_0, C_0) + D_0 + SS2_0$$

Note that θ_0 is fixed and unknown, W_0' is known and changeable, therefore, a DPA attack is applicable. By selecting $\mathrm{HW}(TT1_0) = \mathrm{HW}(\theta_0 + W_0^{'})$ as hypothesis function and making hypotheses about θ_0, we can recover θ_0 and calculate the corresponding values of $TT1_0$. Then, by making hypotheses about A_0, the attacker correlates the power traces with hypothesis for $\mathrm{HD}(A_0, A_1) = \mathrm{HD}(A_0, TT1_0)$. This allows the attacker to recover A_0.

In the mean time, with reference to equations (10), we can get $A_1 = TT1_0$. Alternatively, another attack strategy is selecting $\mathrm{HD}(A_0, A_1) = \mathrm{HW}(A_0 \oplus TT1_0)$ as hypothesis function and making hypothesis about A_0 and θ_0 in the mean time. Thus, we can recover A_0 and θ_0 meanwhile.

$TT2_0$ can be rewritten as:

$$TT2_0 = \eta_0 + W_0$$

where

$$\eta_0 = GG_0(E_0, F_0, G_0) + H_0 + SS1_0$$

Note that η_0 is fixed and unknown, W_0 is known and changeable. Similarly, we can recover η_0 and E_0.

2. According to the above attack, we know A_0 and E_0 before round 1, and can computing the values of A_1 (i.e. $TT1_0$) and E_1 (i.e. $P_0(TT2_0)$). According to equations (8) and (9), we know $C_1 = (B_0 \lll 9)$ and $B_1 = A_0$. Using equation (5), $FF_1(A_1, B_1, C_1) = A_1 \oplus B_1 \oplus C_1$ must be calculated in round 1, where only C_1 is unknown. Now, we can make hypotheses on the bits of C_1, using $\mathrm{HW}(FF_1)$ as hypothesis function. In this way, the attacker can recover C_1 and then calculate B_0.

Similarly, using $GG_1(E_1, F_1, G_1) = E_1 \oplus F_1 \oplus G_1$, the attacker can recover G_1 and F_0.

3. A_0、E_0、B_0、F_0、A_1、B_1、C_1、E_1、F_1、G_1 can be gained from the previous attack stage before round 1. From equation (5), i.e. $TT1_1 = FF_1(A_1, B_1, C_1) + D_1 + SS2_1 + W_1^{'}$, we observe that only D_1, equivalent to C_0, is unknown to the attacker. Therefore, the attacker can generate hypotheses about the unknown values D_1, and attack the output of $FF_1(A_1, B_1, C_1) + D_1$. Recovering D_1 means that the attacker can get C_0.Note that $SS2_1$ is variable and known by the attacker. Alternatively, another attack strategy is selecting $HW(TT1_1)$ or $HD(TT1_0, TT1_1)$ as hypothesis function.

Note that $TT2_1 = GG_1(E_1, F_1, G_1) + H_1 + SS1_1 + W_1$, using a similar approach to above, the attacker can recover H_1, i.e. G_0.

4. By following the above attack process, the attacker can gain A_0、E_0、B_0、F_0、C_0、G_0. The last two remaining secret variables D_0 and H_0 can be found by going back to round 0. Note that $\theta_0 = FF_0(A_0, B_0, C_0) + D_0 + SS2_0$, where the only unknown value is that of D_0, the attacker can compute D_0. Using the same method, H_0 can be calculated from $\eta_0 = GG_0(E_0, F_0, G_0) + H_0 + SS1_0$. The eight 32-bit secret values, i.e. K_{in} are thus recovered, using eight first-order DPA attacks.

C. Recovery of K_{out}

Considering the second call of SM3 in the HMAC algorithm, i.e. $HMAC(m, K) = \mathrm{SM3}\big((K \oplus opad)|h_n\big)$, involving running the compression function twice. From the second calculation, we can observe that the output $h_n = \mathrm{SM3}\big((K \oplus ipad)|m\big)$ of the first application of the hash function SM3 is inputted as message input. Meanwhile, K_{out} is another input. According to the DPA attack described above, the attacker reveals the key K_{in}. For the revealed K_{in}, the attacker computes the output h_n. Reapplying the first-order DPA attack on the second invocation of SM3 in the HMAC algorithm would allow the attacker to recover K_{out}.

On the other hand, there exists an XOR operation $HMAC(m, K) = V^{(2)} \oplus K_{out}$ in the final compression function in the second application of SM3, where $V^{(2)}$ is the output value of sixty-four iterations of K_{out} and h_n. Note that

$HMAC(m, K)$ is public and variable, one of the inputs K_{out} is secret and constant, and the other input $V^{(2)}$ is secret, so the reverse DPA is applicable. Thus, the attacker can reveal the secret K_{out}.

IV. ATTACK ON SOFTWARE IMPLEMENTATION

A. *Testing platform*

To examine the proposed DPA attack in section III, we conducted experiments on a software implementation of HMAC-SM3. Our experimental setup consists of a PC, a power tracer, a smart card and a LeCroy oscilloscope. The smart card is an 8-bit 80251 microprocessor with a software implementation of HMAC-SM3 in it, and does not include any countermeasures against side channel analysis..

The working frequency of the smart card is 4MHz and the sampling frequency is 500MHz. Traces for the first two rounds of the first application of SM3 were captured while 5,000 random messages were being processed. In addition, another 5,000 traces for the last round of the second invocation of SM3 were collected as well. The aligning and re-sampling of the data preprocessing are done after data sampling. Fig.2. and Fig.3. represent for traces for K_{in} and K_{out} respectively.

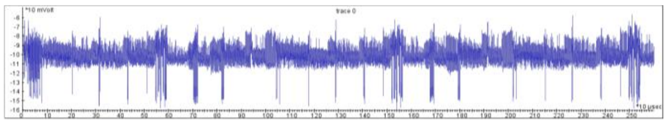

Fig. 2. Traces for K_{in}

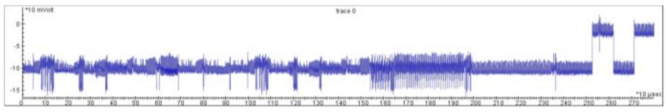

Fig. 3. Traces for K_{out}

B. *Attack results*

We have realized our attack in Inspector 4.5, including the acquisition module and the analysis module. The power model was built using an 8-bit key guess at a time. This choice was motivated by a practical search space (256 different key guesses). There is a preferred direction for DPA starting from the least significant bit.

Fig.4. shows the result of our attack on the least significant byte of θ_0. The least significant byte of θ_0 is 0x91. It only takes 5000 traces that 0x91 comes top in the 256 candidate values.

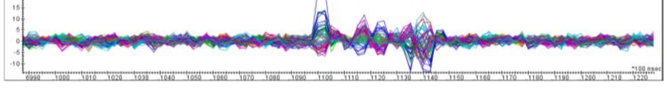

Fig. 4. Result of our attack on the least significant byte of θ_0

The attack results on K_{out} using reverse DPA are described below. Fig.5. shows the result of our attack on the second byte of K_{out}. The second byte of K_{out} is 0x94. It takes 5000 traces that 0x94 comes top in the 256 candidate values.

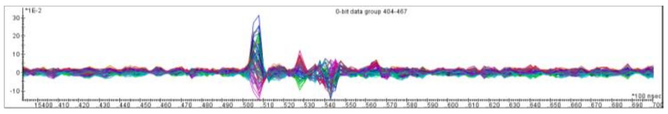

Fig. 5. Result of our attack on the 2nd byte of K_{out}

V. CONCLUSIONS

A first-order DPA attack on HMAC-SM3 has been proposed, and the attacks have been verified with a software implementation of HMAC-SM3. We have shown that the attacker can recover the inner keyed state K_{in} and outer keyed state K_{out}, and forge a message authentication code by using these two intermediate values. Further work will focus on designing other forms of side channel attack, such as higher-order DPA and chosen-plaint attack. Besides, secure HMAC-SM3 algorithm against side channel attacks also needs further research.

Acknowledgment

This work is supported by National Science and Technology Major Project on “Core Electronic Devices, High-end General Chips and Fundamental Software Product” (No. 2014ZX01032401-001) .

References

[1] China’s Office of Security Commercial Code Administration: Sepecification of SM3 Cryptographic Hash Function (2010) (in Chinese), http://www.oscca.gov.cn/UpFile/20101222141857786.pdf

[2] D.P. Ding and X.W. Gao, “Design and implementation of SM3 algorithm on FPGA,” Microcomputer & Its Applications, vol. 31(5),pp. 26-28, 2012.

[3] X.Y. Wang and X.W. Yang, “Optimization design and implementation of SM3 algorithm based on FPGA,” Computer Engineering, vol. 38(6), pp. 244-246, 2012.

[4] Q. Zheng, N. Gao and L.C. Zhang, “Design and implementation of dynamic password card based on SM3 algorithm,” Computer Applications and Software, vol. 30(2), pp. 14-17, 2013.

[5] Y. Hu, A. Wang, L.J. Wu and B.B. Wang, “Hardware design and implementation of SM3 hash algorithm for financial IC card,” 10th International Conference on Computational Intelligence and Security (2014).

[6] K. Lemke, K. Schramm and C. Paar, “DPA on n-bit sized boolean and arithmetic operations and its application to IDEA, RC6,and the HMAC-Construction,” Cryptographic Hardware and Embedded Systems, vol. 3156, pp. 205-218, 2004.

[7] K. Okeya, “Side channel attsacks against HMACs based on block-cipher based hash functions,” Information Security and Privacy, vol. 4058, pp.432-443, 2006.

[8] K. Okeya and T. Iwata, “Side channel attsacks on message authentication codes,” Security and Privacy in Ad-hoc and Sensor Networks, vol. 3813, pp.205-217, 2005.

[9] R. McEvoy, M. Tunstall, C.C. Murphy, and W.P. Marnane, “Differential power analysis of HMAC based on SHA-2, and countermeasures,” Information Security Applications, vol. 4867, pp. 317-332, 2007.

[10] S. Mangard, E. Oswald, and T. Popp, Power anlysis attacks – revealing the secrets of smart cards. Springer, 2007.