Research on Differential Power Analysis of HMAC-SM3

Xie Jun , Sun Wei, Gu Dawu, Guo Zheng, Liu Junrong
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China

Bao Sigang, Ma Bo
Shanghai Huahong Integrated Circuit Co., Ltd.
Shanghai, China

Abstract—**The HMAC algorithm is widely used to provide authentication and message integrity in digital communications. However, if the HMAC cryptographic algorithm is implemented in cryptographic circuit, it is vulnerable to side-channel attacks. A typical example is that in 2007, McEvoy proposed an attack strategy for hardware-based implementation of HMAC-SHA2. In this paper, we research on SM3 cryptographic hash algorithm and propose a DPA attack strategy for the software-based implementation of HMAC-SM3. In the experiment, we launch a successful DPA attack on the practical cryptographic circuit and then discuss the security issues about the software-based implementation of HMAC-SM3.**

Keywords—HMAC; SM3; Side Channel Attacks; Differential Power Analysis

I. INTRODUCTION

Nowadays, with the growing demand for e-mail, online banking, online shopping and other sensitive data communications, cryptography has become an important tool for protecting the privacy of data transmission. Therefore, MAC [1] (Message Authentication Code) cryptographic algorithm is often used to ensure the data integrity and consistency between the sender and the recipient. Usually, both the sender and the recipient hold a shared key, which will be used to encrypt the input message and generate a bit string called MAC. HMAC is a class of MAC algorithm which is based on hash functions. It is often applied in a number of internet security protocols, such as IPsec, TLS, etc.

In recent years, it is not uncommon that embedded cryptographic devices are successfully attacked and be made use of by attackers [9]. Among these attacks, side-channel attack [2] (SCA) such as differential power analysis (DPA) is more well known. These non-invasive attacks collect the power consumption information of cryptographic devices leaked while different messages are processed, so as to achieve the purpose of recovering the cryptographic key of embedded cryptography system.

In the area of HMAC security, researchers in China and abroad have successfully performed side-channel attack to the software implementation of HMAC-SHA2 [4]. In this paper, we will implement HMAC by using the SM3 cryptographic hash function delivered by the State Encryption Administration in 2010, and test its security from the perspective of side-channel attack. The Goals of this paper are to fill the gap in the area of power analysis of SM3 cryptographic hash algorithm, and achieve the aims of timely detection of vulnerabilities and research for protective measures, thereby increasing the security of products, which has great application values.

This work is supported by National Natural Science Foundation of China (No. 61402286, 61472250, 61472249), Special Fund Task for Enterprise Innovation Cooperation from Shanghai Municipal Commission of Economy and Informatization (No. CXY-2013-35), Plan of action for the innovation of science and technology of Shanghai (14511100300).

II. BRIEF INTRODUCTION TO HMAC CRYPTOGRAPHIC ALGORITHM

HMAC message authentication scheme was put forward by Bellare and others in CRYPTO’96 [6], whose aim is to increase the security of MAC by using hash function. The MAC is calculated as follows:

$$HMAC_k(m) = h\left(\left(k \oplus opad\right) // h\left(\left(k \oplus ipad\right) // m\right)\right)$$

Where k is the cryptographic key, and m is the message to be authenticated. *Ipad* and *opad* are fixed-length blocks whose sizes equal to the block size of hash function *h*, filled with *0x36* and *0x5c* as the initial value of each byte. // and ⊕ respectively represent the connection and *XOR* operations.

In order to compute $HMAC_k(\mathrm{m})$, the hash function *h* is called for two times. As the two calling of hash function is quite similar, this paper only focuses on the medium MAC generated by the first run of hash function:

$$HMAC_k{}'(m) = h\left(\left(k \oplus ipad\right) // m\right)$$

MD5 and SHA-1 are recommended to be used as hash function h in Bellare’s paper. But as time goes by, weaknesses of those functions are also gradually known by people [7]. Thus, this paper selects the SM3 cryptographic hash algorithm delivered by State Encryption Administration in 2010 as the hash function *h*, whose implementation theory will be briefly introduced in the next section.

III. SM3 CRYPTOGRAPHIC HASH ALGORITHM

SM3 cryptographic hash algorithm is mainly applied in the MAC generation, digital signatures verification, random number generation, etc. in commercial encryption applications. It can meet the security requirements of sender and receiver during the digital communication, improving the reliability and interoperability of relative security products. SM3 cryptographic hash algorithm is capable to receive messages of any length, and finally output a 256-bit-length hash value after a series of padding and iterative compression operations.

A. Padding

Suppose a message m of 8 bit length is received. Firstly, a bit “1” will be appended to the end of the message, and then k bit of “0” will be appended so as to satisfy the equation:

$$l + 1 + k \equiv 448 \mod 512,$$

where k is the minimum non-negative integers satisfying the equation. Lastly, a 64-bit-length string, the binary representation for l, will be appended to the end of the message.

So far, the padding operation is accomplished, generating a new message m' whose length is a multiple of 512 bits, thus ensuring successful execution of the iterative compression process, which will be introduced in the next section.

B. Iterative compression

1) Iterative compression

First, the message m' generated by padding is divided into N parts as:

$$m' = B^0 B^1 \ldots B^{N-1},$$

where each B^i is a 512-bit-length block and $N = (l + k + 65) / 512$. And then N rounds iterative compression operations are performed to m′ as follows:

```
FOR i = 0 TO N-1
    $V^{i+1} = CF(V^i, B^i)$
ENDFOR
```

where CF refers to compression function, which will be introduced in detail in the next section, V^0 is a 256-bit-length initial value defined in official documentation for SM3 and V^n is the 256-bit-length iterative compression final output.

2) Message expansion

Message block B^i is then expended to 132 32-bit-length strings: $W_0, W_1 \ldots W_{67}, W_0', W_1', \ldots, W_{63}'$ by following three steps:

- Divide the 512-bit-length message block B^i as: $B^i = W_0, W_1 \ldots W_{15}$, where each $W_i (0 \le i \le 15)$ is a 32-bit-length string.
- $W_j (16 \le j \le 67)$ is generated as follows:

```
FOR j = 16 TO 67
    $T \leftarrow P_1\left(W_{j-16} \oplus W_{j-9} \oplus \left(W_{j-3} \lll 15\right)\right)$
    $W_j \leftarrow T \oplus \left(W_{j-13} \lll 7\right) \oplus W_{j-6}$
ENDFOR
```

  Where $\lll$ means cyclic left shift, and

$$P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$$

- $W_j' (0 \le j \le 63)$ is generated as follows:

```
FOR j = 0 TO 63
    $W_j' = W_j \oplus W_{j+4}$
ENDFOR
```

So far, 132 strings have been generated to be used for the 64 rounds CF compression function.

3) Compression function

Suppose that $ABCDEFGH$ are all 32-bit registers, and $SS1$, $SS2$, $TT1$, $TT2$ are medium values. Then the compression function $V^{i+1} = CF(V^i, B^i)$ Can be demonstrated in detail as follows :

```
$ABCDEFGH \leftarrow V^i$
FOR j=0 TO 63
    $SS1 \leftarrow ((A \lll 12) + E + (T_i \lll j)) \lll 7$
    $SS2 \leftarrow SS1 \oplus (A \lll 12)$
    $TT1 \leftarrow FF_j(A,B,C) + D + SS2 + W_j'$
    $TT2 \leftarrow GG_j(E,F,G) + H + SS1 + W_j$
    $D \leftarrow C$
    $C \leftarrow B \lll 9$
    $B \leftarrow A$
    $A \leftarrow TT1$
    $H \leftarrow G$
    $G \leftarrow F \lll 19$
    $F \leftarrow E$
    $E \leftarrow P_0(TT2)$
ENDFOR
$V^{i+1} \leftarrow ABCDEFGH \oplus V^i$
```

The detail operations are shown as follows:

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

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

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

$$T_i = \begin{cases} 0x79cc4519 & 0 \le j \le 15 \\ 0x7a879d8a & 16 \le j \le 63 \end{cases}$$

and $\wedge, \vee, \neg$ respectively represent 32-bit AND, OR, NOT operations.

4) Output the hash value

$$ABCDEFGH \leftarrow V^n$$

SM3 cryptographic hash algorithm finally output a 256-bit hash value y:

$$y = ABCDEFGH$$

IV. Differential Power Analysis theory

Side-channel attack collects the physical information leaked during the cryptographic devices perform encryption or decryption process, and make use of such leaked physical information to recover the cryptographic key value in the cryptographic systems. Normally, side-channel attack can be divided into two stages: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). During the SPA stage, attackers locate the positions where specific algorithms are operated by analyzing the features of power traces leaked by the cryptographic devices. Although SPA cannot recover the cryptographic key directly, it really can save plenty of time for the next DPA stage. In the DPA stage, by analyzing the correlation between the power consumption and sensitive information at specific time, attackers can recover the cryptographic key value using relative statistics theories. As large key length will add the complexity of problem, attackers can make use of divide and conquer strategy to reduce the complexity.

Next, we use an example to demonstrate the DPA theory. Suppose the first round of a block cipher is

$$y = S\left(k \oplus x\right)$$

where y is the output of first round, k is a subkey, and x is the input plaintext. There are three steps to implement DPA,

- Attackers should first determine a specific medium value that may leak the cryptographic key information by analyzing the cryptographic algorithm. Then attackers can guess the subkey value k^*, and use the candidate subkey to compute the medium value for each different plaintext x_i. In this instance, it can be computed as:

  $$v_i^{k^*} = S\left(k^* \oplus x_i\right),$$

- The second step is to compute the theoretical power consumption for each medium value. An appropriate power model is needed here. For a software-implemented circuit, Hamming weight (HW) model is always preferred. In this instance, the power consumption can be computed as

  $$h_i^{k^*} = HW\left(v_i^{k^*}\right)$$

- For each candidate subkey k^*, attackers then compute the correlation between the theoretical power consumption values and the true consumption values in the collected power traces. The candidate subkey which leads to the maximum correlation will be determined as the true subkey.

V. DPA of HMAC-SM3

In this section, we will demonstrate how to attack HMAC-SM3 by using DPA.

A. Attacking Target

As is introduced in section II, HMAC is computed as

$$HMAC_k(m) = h\left(\left(k \oplus opad\right) // h\left(\left(k \oplus ipad\right) // m\right)\right)$$

More computation details are demonstrated in figure 1.

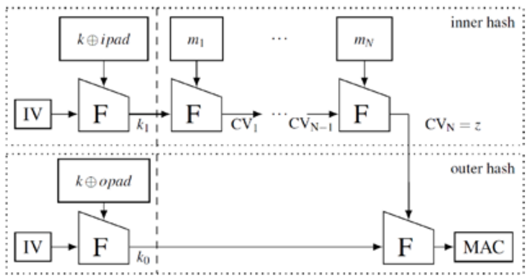

Fig. 1. HMAC computation process

As we can see, for a specific key value k, as *ipad*, *opad* and *IV* are all fixed and known values, the medium values k_1 and k_0, outputs for $k \oplus ipad$ and $k \oplus opad$ in the first round, are also fixed. In order to forge MACs for arbitrary messages, we need either to recover k or to recover both k_1 and k_0. Supposing k_1 is already recovered, the attack of k_0 is absolutely similar to the former attack of k_1. Thus, this paper only focuses on attacking k_1 . In this paper, the F function in figure 1 is SM3 cryptographic hash function. Reviewing the procedure of SM3, we can find that the k_1 is exactly the initial value of $V^1 = ABCDEFGH$ when operating the first message block m_1. Therefore, we focus on the encryption of m_1., and our target is to recover the initial value

$$V^1{}_0 = A_0B_0C_0D_0E_0F_0G_0H_0,$$

which has contained the cryptographic key information.

B. Attacking Strategy

Reviewing the procedure of compression function of SM3, we can see there is 64 rounds iterative operation for each message block. Suppose the output of round i is $A_iB_iC_iD_iE_iF_iG_iH_i$. While computing

$$TT1_0 \leftarrow FF_0(A,B,C) + D + SS2 + W_0{}',$$

as $W_0{}'$ is obtained by operating plaintext, so it is known and variable. And $\theta_0 = FF_0(A_0,B_0,C_0) + D_0 + SS2_0$ is a fixed value, so we can use DPA for modular plus operation to recover θ_0, compute $TT1_0$ and then $A_1 = TT1_0$ can also be known. While computing $FF_1(A_1,B_1,C_1)$, $B_1 = A_0$ and $C_1 = B_0 \lll 9$ are fixed, and A_1 is variable and known, so we can use DPA for *XOR* operation to recover B_1 and C_1, and obtain the value of A_0 and B_0 at the same time. As the value of A_1, B_1 and C_1 are all recovered, while computing

$$TT1_1 \leftarrow FF_1(A_1,B_1,C_1) + D_1 + SS2_1 + W_1{}',$$

$FF_1(A_1,B_1,C_1) + SS2_1 + W_1{}'$ are variable and known, and D_1 is fixed and unknown, so we can use DPA to recover D_1 and at the same time recover C_0. As A_0,B_0,C_0 are all recovered, and $SS2_0$ is also known, we can compute the value of D_0 as:

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

As the iterative operations of E, F, G and H are quite similar to A, B, C and D, we can recover E_0, F_0, G_0, H_0 in the same way. Firstly we can attack

$$TT2_0 \leftarrow GG_0(E,F,G) + H + SS1 + W_0,$$

and recover $\theta'_0 = GG_0(E_0,F_0,G_0) + H_0 + SS1_0$, $TT2_0$ and $E_1 = P_0(TT2_0)$. Secondly, we attack $GG_1(E_1,F_1,G_1)$, recover F_1,G_1 and at the same time obtain the value of E_0 and F_0. Then, we attack

$$TT2_1 \leftarrow GG_1(E_1,F_1,G_1) + H_1 + SS1_1 + W_1,$$

recover H_1, and obtain the value of $G_0=H_1$. At last we can compute H_0 by:

$$H_0=\theta'_0 - GG_0(E_0,F_0,G_0) - SS1_0.$$

So far, the initial value $A_0B_0C_0D_0E_0F_0G_0H_0$ is completely recovered, which means our target is achieved. Thus, we have proved that DPA for HMAC-SM3 software implementation is practical in theory.

VI. Attacking Result

A hardware platform for power trace collection is established. We input random messages of specific length, which ensures that the length of messages generated after padding is 64 bit, to the HMAC-SM3 software implementation card, and collect the power traces.

An autocorrelation operation is performed and the result is shown in figure 2. We can see there are obviously four rounds of iterative operations. As the message generated after padding is 64 bit long, it doesn't need divide and the count of block is $N=1$. As a result, there are exactly four F functions in the whole process showed in figure 1, which is consistent with the four squares in figure 2. Then we locate on the second round. There are obviously 64 rounds iterative operations, and we focus on the first two rounds. The average power trace of first three rounds of the compression function is showed in figure 3.

First, we focus on the first round and attempt to attack

$$TT1_1 \leftarrow \theta_0 + W_1',$$

where $\theta_0 = FF_0(A_0,B_0,C_0) + D_0 + SS2_0$ is fixed and unknown, and W_1' is variable and known. We guess the θ_0 value, compute the $TT1_1$ and furthermore establish a Hamming weight model to compute the power consumption in theory. The candidate key which leads to the maximum correlation between theoretical power consumption and true power consumption showed in the power trace is determined as the true cryptographic key.

As the length of θ_0 is 32 bit, we use divide and conquer strategy and recover 8 bit at a time. The result of the attack of the lowest 8 bit is showed in table 1.

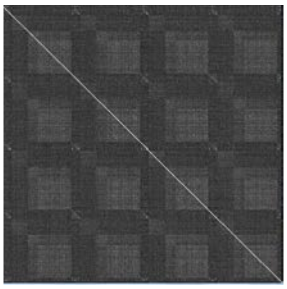
Fig. 2. Autocorrelation of HMAC-SM3 power trace

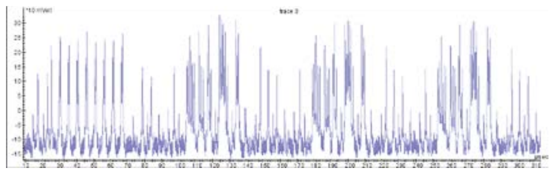
Fig. 3. Average power trace of first three rounds of the compression function

TABLE I. Best correlation of lowest 8bit attack

Number	Subkey	Correlation	Position
0	*0x10*	0.1240	63061
1	*0x90*	0.0916	63061
2	*0x30*	0.0770	63062
3	*0x12*	0.0751	63062
4	*0xD0*	0.0745	63061

We can see the correlation caused by the first candidate subkey *0x10* is obviously far greater than others, so *0x10* is determined as the true value of the lowest 8 bit of θ_0, which has already been verified. Other bits can also be recovered in this way.

Similarly, we can recover $A_0B_0C_0D_0E_0F_0G_0H_0$ using the strategy put forward in section V. It is testified that HMAC-SM3 software implementation circuit can be successfully attacked by DPA.

VII. Conclusion

This paper first introduces the HMAC cryptographic algorithm and SM3 cryptographic hash function, and then put forward a DPA-based power analysis method for HMAC-SM3 software implementation circuit. In the experiment demonstrated in section VI, we perform DPA to true HMAC-SM3 circuit, and the result indicates that we are able to recover the medium value k_1 and k_0, which can be used to forge MACs for arbitrary messages. It can be seen from our research that cryptographic systems without any countermeasures are vulnerable to DPA even though it is mature.

[1] Benoît O, Peyrin T. Side-channel analysis of six SHA-3 candidates[M]//Cryptographic Hardware and Embedded Systems, CHES 2010. Springer Berlin Heidelberg, 2010: 140-157.

[2] Joye M, Quisquater J J. Hessian elliptic curves and side-channel attacks[C]//Cryptographic Hardware and Embedded Systems—CHES 2001. Springer Berlin Heidelberg, 2001: 402-410.

[3] Kocher P, Jaffe J, Jun B. Differential power analysis[C]//Advances in Cryptology—CRYPTO'99. Springer Berlin Heidelberg, 1999: 388-397.

[4] McEvoy R, Tunstall M, Murphy C C, et al. Differential power analysis of HMAC based on SHA-2, and countermeasures[M]//Information security applications. Springer Berlin Heidelberg, 2007: 317-332.

[5] SM3 Cryptographic Hash Algorithm // www.oscca.gov.cn/UpFile/20101222141857786.pdf

[6] Krawczyk H, Canetti R, Bellare M. HMAC: Keyed-hashing for message authentication[J]. 1997.

[7] McEvoy R, Tunstall M, Murphy C C, et al. Differential power analysis of HMAC based on SHA-2, and countermeasures[M]//Information security applications. Springer Berlin Heidelberg, 2007: 317-332.

[8] Sedgwick P. Pearson's correlation coefficient[J]. BMJ: British Medical Journal, 2012, 345.