Yet Another Side Channel Cryptanalysis on SM3 Hash Algorithm

Christophe Clavier, Leo Reynaud and Antoine Wurcker

eshard, France, surname.name@eshard.com

Abstract. SM3, the Chinese standard hash algorithm inspired from SHA2, can be attacker by similar means than SHA2 up to an adaptation to its differences. But this kind of attack is based on targeting point of interest of different kinds, some are end of computation results, that are stored when others are in intermediate computational data. The leakage effectiveness of the later could be subject to implementation choices, device type or device type of leakage. In this paper, we propose a new approach that targets only the first kind of intermediate data that are more susceptible to appear. As an example, we targeted the HMAC construction using SM3, where our method allows to recover the first half of the secret information. reducing the security of the HMAC protocol.

Keywords: SM3 · Side-channel · Hash functions · HMAC · Chosen input

1 Introduction

Side-channels research field concerns the malicious usage of an involuntary leaked information during the execution of an algorithm, with the objective to retrieve a secret such as a cryptographic key. This was first introduced in [Koc96], where the processing time of operations, that was secret-dependent, was revealing the secret information. Once the side-channels potential discovered, numerous channels where used to extract secret information, such as: power consumption [KJJ98], electromagnetic emissions [GMO01], acoustic emissions [GST14] (extension of preliminary work of 2004), light emission [FH08]. The subject is wide, as in 2014 was shown a new leak source can be used: the ground of a laptop could leak sensitive information along cables (USB, Ethernet, . . .) in [GPT14].

The paper is organized as follows. The remaining of Section 1 describe HMAC and SM3 designs and clas. The Section 2 shows how an attack against SHA2-256 can be adapted to SM3. Our contribution is given in Section 3 showing a new way to attack SM3. Finally we conclude in Section 4.

1.1 HMAC

The HMAC stands for keyed-Hash Message Authentication Code and is a NIST standard that can be found in [NIS08]. It is using one eligible hash function (H) to combine an input text (T) and a key (K) into a HMAC that is used to authenticate the text T by a receiver that shares the key K with the sender.The process is as follow:

1. (K_i, K_o) pair is derived from the key K
2. The text T is concatenated after K_i.
3. The resulting data is hashed.
4. The output of this first hash is then concatenated after K_o.

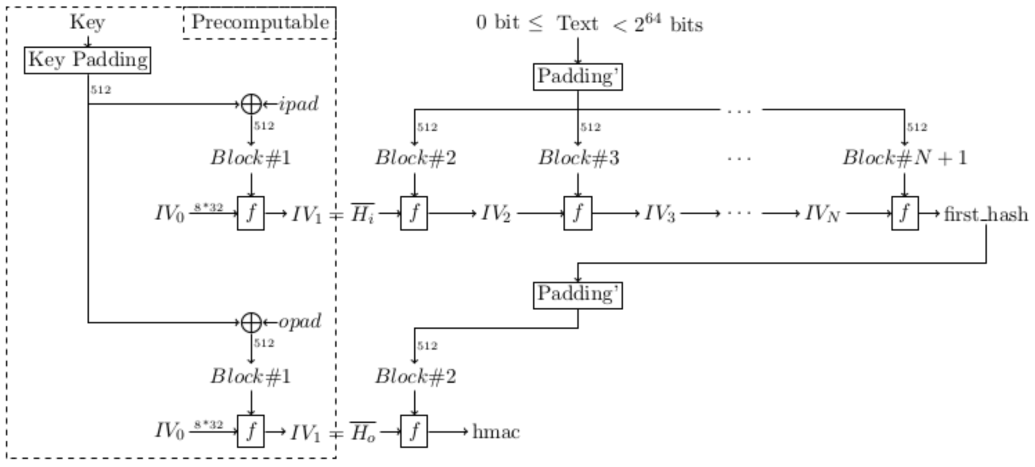

Figure 1: HMAC algorithm structure

5. The resulting data is hashed, giving the HMAC value.

The Figure 1 shows the HMAC of a Hash function that is taking 512-bit message blocks as input. We denote by $\overline{H_i}$ (respectively $\overline{H_o}$) the output of first compression function f of the first (respectively second) hash call. These values are constant (often pre-computed) and their knowledge is sufficent to forge valid HMAC. We consider these values as target of our attack.

1.2 SM3

The SM3 is the Chinese hash standard based on same structure than SHA2-256. The Figure 2 shows the main parts of the SM3 algorithm. A data to be hashed of a bit length between 0 and $2^{64} - 1$ is taken as input. The data is then padded in such a way that it can be split in N blocks of 512 bits (same process than SHA2-256). Each block will be treated the same way: a function $f()$ takes the current block and the current Initialisation Vector (IV) to produce the IV used by next block. The first IV (IV_0) is a constant. The Last IV (IV_N) is the 256-bit hash given as output of the algorithm.

The function $f()$, that differs from the one of SHA2-256, is composed of several parts:

- Function $T()$ convert 512-bit block of padded data into 64 32-bit words pairs, all known to an attacker who knows the input data.

- Each resulting pair (W_i,W_i^*) is used during round R_i that melts it into the 8 32-bit values given as input.

- The resulting 8 32-bit values are given as input of next round.

- Once the 64th round output is computed, a final transformation consists in applying addition (pairwise 32-bit modular addition) with the input used for first round.

In the loop process $IV_i = f(IV_{i-1}, \text{Block}_i)$: We initialize the 8 32-bit local variables

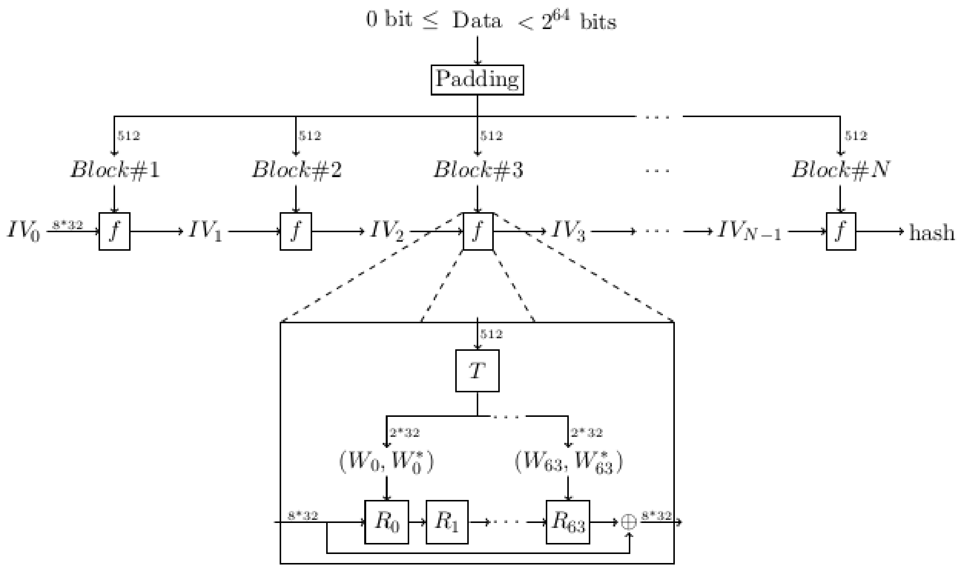

Figure 2: SM3 algorithm structure

named a to h:

$$\begin{aligned}
a_0 &= IV_{i-1,0}\\
b_0 &= IV_{i-1,1}\\
c_0 &= IV_{i-1,2}\\
d_0 &= IV_{i-1,3}\\
e_0 &= IV_{i-1,4}\\
f_0 &= IV_{i-1,5}\\
g_0 &= IV_{i-1,6}\\
h_0 &= IV_{i-1,7}
\end{aligned}$$

For round $r \in [0, 63]$:

$$\begin{aligned}
SS1_r &= ((a_r \lll 12) + e_r + (T_r \lll r)) \lll 7 && (1)\\
SS2_r &= SS1_r \oplus (a_r \lll 12) && (2)\\
TT1_r &= FF_r(a_r, b_r, c_r) + d_r + SS2_r + W_r^* && (3)\\
TT2_r &= GG_r(e_r, f_r, g_r) + h_r + SS1_r + W_r && (4)\\
d_{r+1} &= c_r && (5)\\
c_{r+1} &= b_r \lll 9 && (6)\\
b_{r+1} &= a_r && (7)\\
a_{r+1} &= TT1_r && (8)\\
h_{r+1} &= g_r && (9)\\
g_{r+1} &= f_r \lll 19 && (10)\\
f_{r+1} &= e_r && (11)\\
e_{r+1} &= P_0(TT2_r) && (12)
\end{aligned}$$

- Additions are done modulo 2^{32}.
- $\lll n$ are rotation of n bits to the left of a 32-bit word.

- T_r are constant values depending on the round number.
- $P0()$, $FF()$ and $GG()$ are transformation functions giving 32-bit word as output.

We finish the $f()$ function by the 32-bit xor with initial state:

$$\begin{aligned}
IV_{i,0} &= IV_{i-1,0} \oplus a_{64}\\
IV_{i,1} &= IV_{i-1,1} \oplus b_{64}\\
IV_{i,2} &= IV_{i-1,2} \oplus c_{64}\\
IV_{i,3} &= IV_{i-1,3} \oplus d_{64}\\
IV_{i,4} &= IV_{i-1,4} \oplus e_{64}\\
IV_{i,5} &= IV_{i-1,5} \oplus f_{64}\\
IV_{i,6} &= IV_{i-1,6} \oplus g_{64}\\
IV_{i,7} &= IV_{i-1,7} \oplus h_{64}
\end{aligned}$$

In the following we will denote $\delta_{1,r}$ (respectively $\delta_{2,r}$) a part of the $TT1_r$ (respectively $TT2_r$) computation:

$$\begin{aligned}
\delta_{1,r} &= FF_r(a_r, b_r, c_r) + d_r + SS2_r\\
\delta_{2,r} &= GG_r(e_r, f_r, g_r) + h_r + SS1_r
\end{aligned}$$

One can remark that SM3 cannot be attacked because this is a hash function that processes raw data and then there is no secret to retrieve. But, its usage in protocols, such as HMAC, may introduce a secret information that will be processed and can be targetted by a side-channel attack.

2 Attack Adapted from State-of-the-Art

We can adapt SHA2-256 classical side-channel attack path to SM3, inspired by the one of SHA2-256:

The attack consists in 10 steps, 8 statistical attacks and two of simple computation.

1. Recover $\delta_{1,0}$ constant by targeting the computation of $TT1_0 = \delta_{1,0} + W_0^*$ thanks to the knowledge of varying W_0^*
2. Recover $\delta_{2,0}$ constant by targeting the computation of $TT2_0 = \delta_{2,0} + W_0$ thanks to the knowledge of varying W_0
3. Recover a_0 constant by targeting the computation of $TT1_0 \oplus a_0$ during $TT1_1$ thanks to the knowledge of varying $TT1_0$
4. Recover rotl9(b_0) constant by targeting the computation of $TT1_0 \oplus a_0 \oplus (b_0 \lll 9)$ during $TT1_1$ computation thanks to the knowledge of varying $TT1_0$
5. Recover e_0 constant by targeting the computation of $P_0(TT2_0) \oplus e_0$ during $TT2_1$ computation thanks to the knowledge of varying $TT2_0$
6. Recover rotl19(f_0) constant by targeting the computation of $P_0(TT2_0) \oplus e_0 \oplus (f_0 \lll 19)$ during $TT2_1$ computation thanks to the knowledge of varying $TT2_0$
7. Recover c_0 constant by targeting the computation of $TT1_1 = c_0 + \ldots$ thanks to the knowledge of varying $TT1_0$ and W_1^*
8. Recover g_0 constant by targeting the computation of $TT2_1 = g_0 + \ldots$ thanks to the knowledge of varying $TT2_0$ and W_1
9. Recover d_0 constant by simply computing equation $d_0 = \delta_{1,0} - (a_0 \oplus b_0 \oplus c_0) - SS2_0$
10. Recover h_0 constant by simply computing equation $h_0 = \delta_{2,0} - (e_0 \oplus f_0 \oplus g_0) - SS1_0$

3 Our Contribution

We considered that several step (steps 3 to 8) of the attack path are targeting intermediate computation results that leakage is very sensitive to choices in implementations and chosen device. Contrarily to step 1 and 2 that target computational result ($\delta_{1,0}$ and $\delta_{2,0}$) that must be stored for the next round and seems then more resilient as point of interest. We then designed an chosen text attack process that rely only on deltas leakages.

3.1 Attack Set 1

We let all byte of message vary and recover $\delta_{1,0}$ and $\delta_{2,0}$ as described in previous attack.

3.2 Attack Set 2

This time we let all bytes of message vary except the ones of W_0 and W_4 (32-bit words) that are fixed to any value. This choice will fix W_0 and W_0^* values and then all outputs of first loop are fixed too ($TT1_0$,$TT2_0$, a_1, b_1, c_1, d_1, e_1, f_1, g_1, h_1). Even if some remains unknown, the attacker knows that they are fixed and then induce the delta values of second round ($\delta_{1,1}$ and $\delta_{2,1}$) to be constant too. As they are constant, they can be attacked by statistical attacks using conserved variability of W_1 and W_1^*.

3.3 Attack Set 3

The same process than previously is applied to attack third round: W_0 and W_4 are kept fixed to have constancy in first round, and now W_1 and W_5 are fixed too to keep constancy in second round. This induce that $\delta_{1,2}$ and $\delta_{2,2}$ are now constants and then can be targeted using conserved variability of W_2 and W_2^*.

3.4 Attack Set 4

Again the same process than previously is applied to attack fourth round: W_0, W_1, W_4 and W_5 are kept fixed to have constancy in first and second rounds, and now W_2 and W_6 are fixed too to keep constancy in third round. This induce that $\delta_{1,3}$ and $\delta_{2,3}$ are now constants and then can be targeted using conserved variability of W_3 and W_3^*.

One can remark we could not get further and recover next round deltas ($\delta_{1,4}$ and $\delta_{2,4}$) by this methodology as it would require to have W_4 both fixed (to keep constancy of first round) and variable (to be able to attack delta values) that is impossible. But the acquired information is enough to gain the secret value involved.

Indeed knowing pairs $(\delta_{1,i}, \delta_{2,i}), \forall i \in [0, 3]$ of SM3 hash lead to knowledge of Initial Vector (IV) used. First of all, the knowledge of such delta values $\{\delta_{j,i}\}_{j\in[1,2],i\in[0,3]}$ induce the knowledge of corresponding $TT1$ and $TT2$ values. We take the equation of the fourth set when $\{TTj_i\}_{j\in[1,2],i\in[0,2]}$ (of three first rounds only) are fixed constants and known. In the following equations we indicate in red the values that are unknown and in black the value that are known by the attacker:

$$
\begin{aligned}
\delta_{1,0} =& (a_0 \oplus b_0 \oplus c_0) + d_0 + (((a_0 \lll 12) + e_0 + \mathrm{Cte}_0) \oplus (a_0 \lll 12))\\
\delta_{2,0} =& (e_0 \oplus f_0 \oplus g_0) + h_0 + ((a_0 \lll 12) + e_0 + \mathrm{Cte}_0)\\
\delta_{1,1} =& (TT1_0 \oplus a_0 \oplus (b_0 \lll 9)) + c_0 + (((TT1_0 \lll 12) + P0(TT2_0) + \mathrm{Cte}_1) \oplus (TT1_0 \lll 12))\\
\delta_{2,1} =& (P0(TT2_0) \oplus e_0 \oplus (f_0 \lll 19)) + g_0 + ((TT1_0 \lll 12) + P0(TT2_0) + \mathrm{Cte}_1)\\
\delta_{1,2} =& (TT1_1 \oplus TT1_0 \oplus (a_0 \lll 9)) + (b_0 \lll 9) + (((TT1_1 \lll 12) + P0(TT2_1) + \mathrm{Cte}_2) \oplus (TT1_1 \lll 12))\\
\delta_{2,2} =& (P0(TT2_1) \oplus P0(TT2_0) \oplus (e_0 \lll 19)) + (f_0 \lll 19) + ((TT1_1 \lll 12) + P0(TT2_1) + \mathrm{Cte}_2)\\
\delta_{1,3} =& (TT1_2 \oplus TT1_1 \oplus (TT1_0 \lll 9)) + (a_0 \lll 9) + (((TT1_2 \lll 12) + P0(TT2_2) + \mathrm{Cte}_3) \oplus (TT1_2 \lll 12))\\
\delta_{2,3} =& (P0(TT2_2) \oplus P0(TT2_1) \oplus (P0(TT2_0) \lll 19)) + (e_0 \lll 19) + ((TT1_2 \lll 12) + P0(TT2_2) + \mathrm{Cte}_3)
\end{aligned}
$$

As we can see, the number of unknown values in equations decrease. The two last equations allows to recover the values of a_0 and e_0. Then, these values can be used in the two previous equations to recover b_0 and f_0 and so on to the full recovery of all sought values $\{a_0, \ldots, h_0\}$.

The main limitation of this methodology is that the attacker do not have control (only knowledge) over the second hash input and then cannot pursue the secret recovery by the same methodology.

4 Conclusion

SM3 scheme is sensitive to this attack due to its design particularities as this attack do not seems to be applicable "as is" on SHA2-256.

Even if this attack is limited to the first half of the secret of a HMAC, it still reduces its security. Furthermore, any scheme using only one hash would be subject to total recovery of the secret by this attack.

As a further work, we are currently working on adaptations of this attack to enable the recovery of the second part of the secret of HMAC protocol. Considering others hash functions would also be of interest.

References

[FH08] Julie Ferrigno and Martin Hlavác. When AES blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '01*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.

[GPT14] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES '14*, volume 8731 of *Lecture Notes in Computer Science*, pages 242–260. Springer, 2014.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer, 2014.

[KJJ98] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Introduction to Differential Power Analysis and Related Attacks. Technical report, Cryptography Research Inc., 1998. http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[NIS08] NIST. Fips pub 198-1, the keyed-hash message authentication code (hmac). Technical report, 2008.