# The Dangerous Message/Key Swap in HMAC

Antoine Wurcker and David Marçais

Serma Safety and Security
Pessac, France
a.wurcker@serma.com
d.marcais@serma.com

**Abstract.** Classical usage of Hash-based Message Authentication Code (HMAC) is known to be subject to Side-Channel Attacks (SCA). In case physical leakage of information occurs, and without proper countermeasures, the variability of known message endangers the secret key through statistical attacks. These attacks are not always applicable due to strict constraints on attacker model. However, some protocols can choose not to use the HMAC the way it was designed by swapping the roles of message and key, and therefore modify the condition of applicability of such attacks.

In this paper, we describe a new attack method that takes advantage of the reduction of constraints induced by the choice to swap roles. Moreover, we describe two methods that can optimize previously existing attacks against HMAC. One allowing to reduce the trace cost of some attacks by $\sim 25\%$ , the other allowing to adapt existing attacks even in presence of partial countermeasures.

**Keywords:** HMAC · SHA-2 · SCA · Side-Channel · DPA · Masking · HKDF

## 1 Introduction

*Side-Channel Attacks* (hereafter called SCA) are a type of attack that use an unplanned source of information (side-channel) leaking from hardware component. When such a component runs an algorithm, fluctuations can appear on physical measurements that are related to the values processed by the chip. If algorithms are processing data related to secret information, such as in cryptographic algorithms, the related leakages can lead to disclosure of secret information. The first paper to consider these information leakages was focusing on execution time of operations [11]. Once the potential of SCA discovered, other leakage mediums were exploited such as: power consumption [12], electromagnetic emissions [8], light emissions [6], acoustic emissions [9]. These sources of information can since be exploited by attackers, allowing to target secrets during their processing in an hardware component.

*Differential Power Analysis* (hereafter called DPA) is a statistical SCA described in [12]. Other enhancements of this attack exist such as the *Correlation Power Analysis* [5]. In an attack of this family (hereafter called *DPA-like*), the attacker performs numerous executions of code on a physical component that involve the targeted secret information. For each execution, a measurement through a side-channel is performed. Attack will then be possible depending on the presence of leakage resulting from the combination of i) a known variable (e.g. message) and ii) an unknown constant (e.g. key). The variations of the measured leakages are expected to correlate the best when the known variable is combined with the correct guess for the unknown constant.

In order to retrieve the used secret key, DPA-like attacks are known to be able to target hash based algorithms such as *Hash-based Message Authentication Code* (hereafter called HMAC) described in [4]. Several publications show how to target HMAC with different leakages assumptions [14, 17, 3, 19].

As HMAC needs an underlying hash function, *Secure Hash Algorithm 2* (hereafter called SHA-2), detailed in [15], is the most targeted algorithm in these publications. However, these attacks are often also applicable on *Secure Hash Algorithm 1* (hereafter called SHA-1) as this algorithm partially shares the same structures as SHA-2. Our contribution also focus on SHA-2 as main example, however, we will show later that it can also be applied to SHA-1.

In these publications, state-of-the-art methods only consider classical usages of HMAC. However, there exists protocols where the role of the message and the key registers are swapped, such as *HMAC-based extract-and-expand Key Derivation Function* (hereafter called HKDF) or *HMAC based Deterministic Random Bit Generator* (hereafter called HMAC-DRBG). For example, in HKDF specification, the usage of a known value (called salt) as the HMAC key and a secret fixed value (called IKM for Input Key Material) as the HMAC message is depicted. We cannot consider usages where the salt value is fixed [18, 1]. This is due to DPA-like attacks that require a variable known input and are therefore not applicable. We only consider protocols where the salt can be variable as a diversifier, as suggested in [13, 10, 7]. Our contributions intend to show that such a swap in the roles of message and key registers of HMAC can be dangerous as it may remove a lot of constraints on attacker model.

In this paper, notations are detailed in Section 2, then the attack methods of state-of-the-art and the evolution of the constraints on attacker model are described in Section 3. Finally, our contributions are detailed in 4.

## 2 Notations

**SHA-2** is in fact a set of several hash functions based on the same algorithm but with several buffer sizes and different initial values, as described in [15]. All these functions are based on the process described in Figure 1. The message is padded and cut in equal chunks depending on the chosen version of algorithm block size. Then, starting with a constant known Initial Value ($IV_0$), IV value is mixed together with a message block by function $f$ to produce the next block IV. Function $f$ processes 64 rounds $R_i$ where IV is progressively mixed with message block tranformed into 64 words ($W_i$). Finally, IV is added to the state before being exited as the next IV. The output of last $f$ is the hash value. $'+'$ means addition performed modulo 2 to the size of the word in bits corresponding to the chosen algorithm. $'\oplus'$ means bitwise exclusive-or operation. $'\wedge'$ means bitwise $'$and$'$ operation. $'\neg'$ means bitwise $'$not$'$ operation.

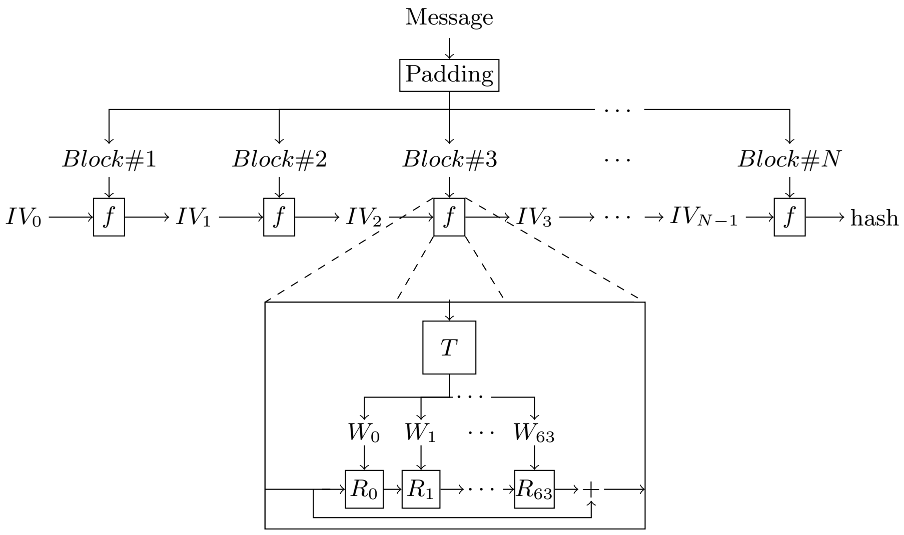

**Fig. 1.** Scheme of SHA-2 process.

Transformed message parts ($W_i$), have the property that the first 16 words are the message block split in 16 words when the last 48 words are computed as combinations of the previous ones. As this is not relevant in this paper, the exact computation of later value is omitted but can be found in specifications [15]. This means that an attacker controlling the message has control over the first 16 $W_i$ values and has knowledge of the next 48.

Entering $f$ function, IV value is split in 8 words $(A_0, \ldots, H_0)$, updated through rounds to produce 8 words as output of the loop $(A_{64}, \ldots, H_{64})$. A final word by word addition between $(A_{64}, \ldots, H_{64})$ and initial $(A_0, \ldots, H_0)$ is performed to produce the IV of next block. Focusing on round $R_i$, equations are:

$$\begin{aligned}
T1_i &= H_i + \Sigma_1(E_i) + \mathscr{C}(E_i, F_i, G_i) + K_i + W_i && (1)\\
T2_i &= \Sigma_0(A_i) + \mathscr{M}(A_i, B_i, C_i) && (2)\\
A_{i+1} &= T1_i + T2_i && (3)\\
B_{i+1} &= A_i && (4)\\
C_{i+1} &= B_i && (5)\\
D_{i+1} &= C_i && (6)\\
E_{i+1} &= T1_i + D_i && (7)\\
F_{i+1} &= E_i && (8)\\
G_{i+1} &= F_i && (9)\\
H_{i+1} &= G_i && (10)
\end{aligned}$$

With some sub-operations ($'$Choice$'$ $\mathscr{C}$ and $'$Majority$'$ $\mathscr{M}$):

$$\begin{aligned}
\mathscr{C}(E_i, F_i, G_i) &= (E_i \wedge F_i) \oplus (\neg E_i \wedge G_i)\\
\mathscr{M}(A_i, B_i, C_i) &= (A_i \wedge B_i) \oplus (A_i \wedge C_i) \oplus (B_i \wedge C_i)
\end{aligned}$$

$\Sigma_0$ and $\Sigma_1$ are not detailed here. The data passed as input of these transformation functions do not undergo changes in properties required for our attack. E.g. a known value passed though these functions will remains known (the same stands for properties: unknown, fix and variable).

$T1_i$ and $T2_i$ are temporary variables used to combine current round inputs $(A_i, \ldots, H_i)$ with round constant $K_i$ and round message part $W_i$. Then, these temporary variables are used to compute the current round outputs, denoted $(A_{i+1}, \ldots, H_{i+1})$.

To ease further explanations, we set up several intermediate data references that are not dependent on known constants $K_i$ and on message part $W_i$:

$$\begin{aligned}
\delta E_{i+1} &= H_i + \Sigma_1(E_i) + \mathscr{C}(E_i, F_i, G_i) + D_i\\
\delta A_{i+1} &= H_i + \Sigma_1(E_i) + \mathscr{C}(E_i, F_i, G_i) + \Sigma_0(A_i) + \mathscr{M}(A_i, B_i, C_i)
\end{aligned}$$

These references are involved in the following equations:

$$\begin{aligned}
A_{i+1} &= \delta A_{i+1} + K_i + W_i\\
E_{i+1} &= \delta E_{i+1} + K_i + W_i
\end{aligned}$$

**HMAC SHA-2** is composed of two successive calls to SHA-2. The first is called *inner* hash when the second is called *outer* hash. The key is processed through a padding process to be of the size of one block. Then, the result is combined with a constant called *ipad* by exclusive-or. Finally, the concatenation of the modified key and the message to process is hashed. The result is used as a message for a second pass of the same process except that *ipad* is now replaced by another constant called *opad*. This process is described in Figure 2. In this figure, message padding method is replaced by Padding′ as the message bit length has a role in padding and must then take into account the concatenation of first block from key part.

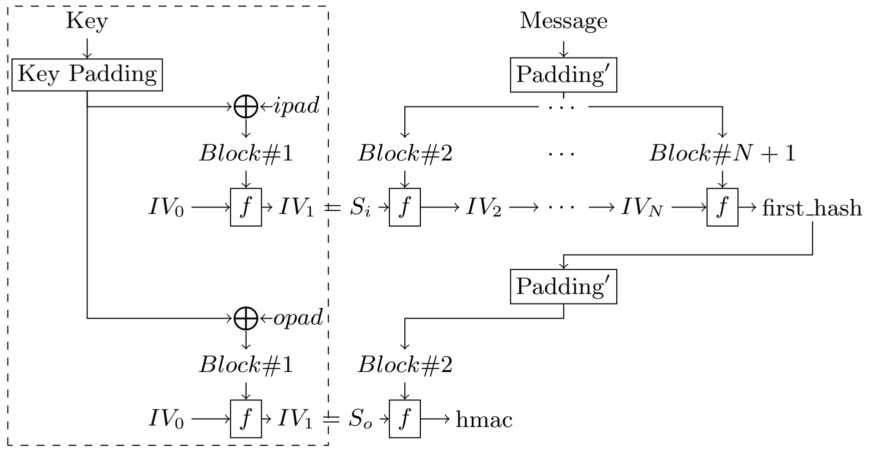

**Fig. 2.** Scheme of HMAC- SHA-2 process.

In a classical usage of HMAC, the key is constant and therefore the dashed area on Figure 2 identifies a part of computation that is constant and does not depend on the message. Most attacks from state-of-the-art aim at retrieving $(S_i, S_o)$. This pair of constant values allows an attacker to forge a valid HMAC output by performing the non framed part of algorithm on Figure 2 with any message.

**HKDF** is a key derivation function based on HMAC. The used inputs are the secret source named IKM (e.g. shared secret during Diffie-Hellman key exchange) and a publicly known salt. The key derivation consists in two steps:

– **Extract** that produces a Pseudo-Random Key (PRK) by using a HMAC with the salt as the key and the secret IKM as the message:

$$\mathrm{PRK} = \mathrm{HMAC}(\mathrm{salt}, \mathrm{IKM})$$

– **Expand**[1] that uses obtained PRK as a key in a HMAC with a chosen message (called info) to produce a derived key:

$$\mathrm{Key}_{\mathrm{info}} = \mathrm{HMAC}(\mathrm{PRK}, \mathrm{info})$$

[1] Details about the *Expand* part of the computation are omitted here as we only focus on *Extract* part. For more details, please consult [13].

## 3 State-of-The-Art Attacks

During attacks explanations below, the known constants $K_i$ will often be associated with known variables (e.g. $W_i$) as it does not have an impact (e.g. $K_i + W_i$ is also a known variable if $W_i$ is). It could also have been included in targeted constant and then been subtracted once the whole constant is recovered.

### 3.1 Early Attacks

**Short description:** Early attacks are described in [14, 2] where authors propose the first attacks allowing to recover secret information of HMAC-SHA-2 through a DPA-like attacks. Both does not apply constraints on attacker model, as only a known message is required. However, specific constraints on the leakage model must be met in order to be able to apply the attacks.

In [14], authors propose an attack in a context where Hamming distance leakages appear on intermediate values of $T1_i$, and on results of bitwise-and operations inside of $\mathscr{C}$ and $\mathscr{M}$ functions.

In [2], authors propose a variation that needs Hamming weight leakages of the same intermediate values.

The attack principle consists in using message variability ($W_i$) to attack $S_i$ value when used as an IV ($(A_0, \dots, H_0)$) during second block absorption of first hash in HMAC. Once recovered, $S_i$ allows to compute the rest of inner hash. The output produced is the input message of outer hash, then the exact same attack can be conducted on $S_o$.

**Step-by-step description:** Is given hereafter, an example derived from early attacks principle. At each step a constant, called here $\chi_i$, is recovered through a DPA-like attack.

1. Target $E_1$ leakage, use the variability of $W_0$ to recover the value of $\chi_1 = \delta E_1$ constant:

$$E_1 = \underbrace{D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0)}_{\delta E_1\text{:Unknown Constant}} + \underbrace{K_0 + W_0}_{\text{Known Variable}}$$

   From now on, $E_1$ becomes a known variable.
2. Target $A_1$ leakage, use the variability of $W_0$ to recover the value of $\chi_2 = \delta A_1$ constant:

$$A_1 = \underbrace{H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + \Sigma_0(A_0) + \mathscr{M}(A_0, B_0, C_0)}_{\delta A_1\text{:Unknown Constant}} + \underbrace{K_0 + W_0}_{\text{Known}}$$

   From now on, $A_1$ becomes a known variable.
3. Target $E_1 \wedge E_0$ leakage, use the variability of $E_1$ to recover the value of $\chi_3 = E_0$ constant:

$$\begin{aligned}\mathscr{C}(E_1, F_1, G_1) &= \mathscr{C}(E_1, E_0, F_0) \\ &= \underbrace{(E_1 \wedge E_0)}_{\text{Expected Leakage}} \oplus (\neg E_1 \wedge F_0)\end{aligned}$$

4. Target $\neg E_1 \wedge F_0$ leakage, use the variability of $E_1$ to recover the value of $\chi_4 = F_0$ constant:

$$\begin{aligned}\mathscr{C}(E_1, F_1, G_1) &= \mathscr{C}(E_1, E_0, F_0)\\ &= (E_1 \wedge E_0) \oplus \underbrace{(\neg E_1 \wedge F_0)}_{\text{Expected Leakage}}\end{aligned}$$

5. Target $A_1 \wedge A_0$ leakage, use the variability of $A_1$ to recover the value of $\chi_5 = A_0$ constant:

$$\begin{aligned}\mathscr{M}(A_1, B_1, C_1) &= \mathscr{M}(A_1, A_0, B_0)\\ &= \underbrace{(A_1 \wedge A_0)}_{\text{Expected Leakage}} \oplus (A_1 \wedge B_0) \oplus (A_0 \wedge B_0)\end{aligned}$$

6. Target $A_1 \wedge B_0$ leakage, use the variability of $A_1$ to recover the value of $\chi_6 = B_0$ constant:

$$\begin{aligned}\mathscr{M}(A_1, B_1, C_1) &= \mathscr{M}(A_1, A_0, B_0)\\ &= (A_1 \wedge A_0) \oplus \underbrace{(A_1 \wedge B_0)}_{\text{Expected Leakage}} \oplus (A_0 \wedge B_0)\end{aligned}$$

7. Target $E_2$ leakage, use the variability of $W_1$ and $E_1$ and the knowledge of $E_0$ and $F_0$ to recover the value of $\chi_7 = (C_0 + G_0)$ constant:

$$\begin{aligned}E_2 &= D_1 + H_1 + \Sigma_1(E_1) + \mathscr{C}(E_1, F_1, G_1) + K_1 + W_1\\ &= \underbrace{C_0 + G_0}_{\text{Unknown}} + \underbrace{\Sigma_1(E_1) + \mathscr{C}(E_1, E_0, F_0) + K_1 + W_1}_{\text{Known Variable}}\end{aligned}$$

8. Target $A_2$ leakage, use the variability of $W_1$, $E_1$ and $A_1$ and the knowledge of $E_0$, $F_0$, $A_0$ and $B_0$ to recover the value of $\chi_8 = G_0$ constant:

$$\begin{aligned}A_2 &= H_1 + \Sigma_1(E_1) + \mathscr{C}(E_1, F_1, G_1) + \Sigma_0(A_1) + \mathscr{M}(A_1, B_1, C_1) + K_1 + W_1\\ &= G_0 + \underbrace{\Sigma_1(E_1) + \mathscr{C}(E_1, E_0, F_0) + \Sigma_0(A_1) + \mathscr{M}(A_1, A_0, B_0) + K_1 + W_1}_{\text{Known Variable}}\end{aligned}$$

Therefore, the resulting system of equation can be solved:

$$\chi_1 = D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) \tag{11}$$
$$\chi_2 = H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + \Sigma_0(A_0) + \mathscr{M}(A_0, B_0, C_0) \tag{12}$$
$$\chi_3 = E_0 \tag{13}$$
$$\chi_4 = F_0 \tag{14}$$
$$\chi_5 = A_0 \tag{15}$$
$$\chi_6 = B_0 \tag{16}$$
$$\chi_7 = C_0 + G_0 \tag{17}$$
$$\chi_8 = G_0 \tag{18}$$

$A_0$, $B_0$, $E_0$, $F_0$ and $G_0$ are directly known (Equations 15, 16, 13, 14, 18). Knowledge of $G_0$ allows to get $C_0$ (Equation 17). $H_0$ can now be recovered as the only unknown value in equation 12. Then, the same applies to $D_0$ being now the only unknown value in equation 11.

**Summary of early attack:** Early attacks have several constraints:

– Known message.
– Varying message with sufficient entropy.
– Leakage on two end-of-turn stored variables ($E_i$, $A_i$) over two first rounds.
– Leakage on four intermediate bitwise-and operations during second round ($E_1 \wedge F_1$, $\neg E_1 \wedge G_1$, $A_1 \wedge B_1$ and $A_1 \wedge C_1$)

Later works pointed a drawback of such an attack in that it requires leakages for two different processed values. One from round-end memory ($A_i$ and $E_i$) that will be stored, read and manipulated several times and one from temporary sub-operations (bitwise-and) that can be harder to target for an attacker depending on the attack conditions.

## 3.2 Partial Attack with Fewer Constraints

**Short description:** Methodology was furnished in slides from [17], briefly showing an update of early attacks. Authors implied that leakage of bitwise-and operations used in Section 3.1 are not necessary anymore in counterpart of a chosen message constraint. This short description was later detailed in [19].

**Step-by-step description:** Description is given here as some part were still omitted in [19]. The new attack process needs four different sets of acquisitions:

– Set $\mathscr{S}_0$ with $W_0$ varying.
– Set $\mathscr{S}_1$ with $W_0$ fixed to any known value and $W_1$ varying.
– Set $\mathscr{S}_2$ with $W_0$ and $W_1$ fixed to any known value and $W_2$ varying.
– Set $\mathscr{S}_3$ with $W_0$, $W_1$ and $W_2$ fixed to any known value and $W_3$ varying.

Then the attack becomes:

1. Focusing on set $\mathscr{S}_0$ they use the same steps (1) and (2) of early attack method described in Section 3.1 to recover $\delta E_1$ and $\delta A_1$ as conditions are identical ($W_0$ varying):

$$\begin{aligned} E_1 &= \underbrace{\delta E_1}_{\text{Unknown Constant}} + \underbrace{K_0 + W_0}_{\text{Known Variable}} \\ A_1 &= \underbrace{\delta A_1}_{\text{Unknown Constant}} + \underbrace{K_0 + W_0}_{\text{Known Variable}} \end{aligned}$$

From now on, $\delta E_1$ and $\delta A_1$ become known constants.

2. Focusing on set $\mathscr{S}_1$, as $W_0$ is fixed to any constant value $W_0^*$, then $E_1$ becomes fixed to a known constant value $E_1^* = \delta E_1 + K_0 + W_0^*$. The same stands for $A_1$ that is fixed to a constant known value $A_1^* = \delta A_1 + K_0 + W_0^*$. Then, $\delta E_2$ and $\delta A_2$ become also constant as the only variable inside them are $E_1$ and $A_1$. Then, the exact same attack than step (1) can be applied to recover $\delta E_2^*$ and $\delta A_2^*$ :

$$\begin{aligned} E_2 &= \underbrace{\delta E_2^*}_{\text{Unknown Constant}} + \underbrace{K_1 + W_1}_{\text{Known Variable}} \\ A_2 &= \underbrace{\delta A_2^*}_{\text{Unknown Constant}} + \underbrace{K_1 + W_1}_{\text{Known Variable}} \end{aligned}$$

From now on, $E_1^*$, $A_1^*$, $\delta E_2^*$ and $\delta A_2^*$ become known constants.

3. The same principle propagates for set $\mathscr{S}_2$ when $(W_0, W_1)$ are fixed to constant values $(W_0^*, W_1^*)$, then $(E_2, A_2)$ becomes known fixed too $(E_2^*, A_2^*)$ and finally $(\delta E_3, \delta A_3)$ become fixed to constant values and the same attack becomes applicable to recover them:

$$\begin{aligned} E_3 &= \underbrace{\delta E_3^*}_{\text{Unknown Constant}} + \underbrace{K_2 + W_2}_{\text{Known Variable}} \\ A_3 &= \underbrace{\delta A_3^*}_{\text{Unknown Constant}} + \underbrace{K_2 + W_2}_{\text{Known Variable}} \end{aligned}$$

From now on, $E_2^*$, $A_2^*$, $\delta E_3^*$ and $\delta A_3^*$ become known constants.

4. Again, the same principle propagates for set $\mathscr{S}_3$ when $(W_0, W_1, W_2)$ are fixed to constant values $(W_0^*, W_1^*, W_2^*)$, then $(E_3, A_3)$ become known fixed too $(E_3^*, A_3^*)$ and finally $(\delta E_3, \delta A_3)$ become fixed to constant values and the same attack becomes applicable to recover them:

$$\begin{aligned} E_4 &= \underbrace{\delta E_4^*}_{\text{Unknown Constant}} + \underbrace{K_3 + W_3}_{\text{Known Variable}} \\ A_4 &= \underbrace{\delta A_4^*}_{\text{Unknown Constant}} + \underbrace{K_3 + W_3}_{\text{Known Variable}} \end{aligned}$$

From now on, $E_3^*$, $A_3^*$, $\delta E_4^*$ and $\delta A_4^*$ become known constants.

Therefore, the resulting system of equation can be solved:

$$
\begin{aligned}
\delta E_1 &= D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) && (19)\\
\delta A_1 &= H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + \Sigma_0(A_0) + \mathscr{M}(A_0, B_0, C_0) && (20)\\
\delta E_2^* &= D_1 + H_1 + \Sigma_1(E_1^*) + \mathscr{C}(E_1^*, F_1, G_1)\\
&= C_0 + G_0 + \Sigma_1(E_1^*) + \mathscr{C}(E_1^*, E_0, F_0) && (21)\\
\delta A_2^* &= H_1 + \Sigma_1(E_1^*) + \mathscr{C}(E_1^*, F_1, G_1) + \Sigma_0(A_1^*) + \mathscr{M}(A_1^*, B_1, C_1)\\
&= G_0 + \Sigma_1(E_1^*) + \mathscr{C}(E_1^*, E_0, F_0) + \Sigma_0(A_1^*) + \mathscr{M}(A_1^*, A_0, B_0) && (22)\\
\delta E_3^* &= D_2 + H_2 + \Sigma_1(E_2^*) + \mathscr{C}(E_2^*, F_2, G_2)\\
&= B_0 + F_0 + \Sigma_1(E_2^*) + \mathscr{C}(E_2^*, E_1^*, E_0) && (23)\\
\delta A_3^* &= H_2 + \Sigma_1(E_2^*) + \mathscr{C}(E_2^*, F_2, G_2) + \Sigma_0(A_2^*) + \mathscr{M}(A_2^*, B_2, C_2)\\
&= F_0 + \Sigma_1(E_2^*) + \mathscr{C}(E_2^*, E_1^*, E_0) + \Sigma_0(A_2^*) + \mathscr{M}(A_2^*, A_1^*, A_0) && (24)\\
\delta E_4^* &= D_3 + H_3 + \Sigma_1(E_3^*) + \mathscr{C}(E_3^*, F_3, G_3)\\
&= A_0 + E_0 + \Sigma_1(E_3^*) + \mathscr{C}(E_3^*, E_2^*, E_1^*) && (25)\\
\delta A_4^* &= H_3 + \Sigma_1(E_3^*) + \mathscr{C}(E_3^*, F_3, G_3) + \Sigma_0(A_3^*) + \mathscr{M}(A_3^*, B_3, C_3)\\
&= E_0 + \Sigma_1(E_3^*) + \mathscr{C}(E_3^*, E_2^*, E_1^*) + \Sigma_0(A_3^*) + \mathscr{M}(A_3^*, A_2^*, A_1^*) && (26)
\end{aligned}
$$

$E_0$ can be recovered in Equation 26 as it is the only unknown value. Then, $A_0$ becomes the only unknown value in Equation 25. This phenomenon is cascading for $F_0$ in Equation 24, $B_0$ in Equation 23, $G_0$ in Equation 22, $C_0$ in Equation 21, $H_0$ in Equation 20 and finally $D_0$ in Equation 19. Eventually, all targeted values $(A_0, \ldots, H_0)$ are recovered.

**Application on HMAC:** Attack on HMAC is incomplete as stated in [17]. The chosen message requirement allows to attack inner hash but this property is lost for outer hash where message is only known. It has to be noted that this partial attack still reduces the security of HMAC.

**Summary of partial attack:** the given methodology releases the constraint from early attacks where several types of leakages were needed. However, in counterpart, the message now requires to be chosen. The constraints of this attack version are:

- Chosen message.
- Leakage on two end-of-turn stored variables ($A_i$, $E_i$) over four first rounds.
- Four sets of traces.

This attack can endanger HMAC but remains only a partial threat to its security. Author of [19] have pursued this work to present a full attack path.

## 3.3 Complete Attack with Fewer Constraints

**Short description:** In [19], author starts from the partial attack suggested in [17] and completes it in order to lead to a full secret recovery. The same conditions are used ($A_i$ and $E_i$ leakages and chosen message constraint). However, in exchange to complete the attack with recovery of $S_o$, the HMAC output must now be known.

**Step-by-step description:** The main idea is to attack by the end of the algorithm as an addition is performed between targeted initial values $(A_0, \ldots, H_0)$ and the output of rounds that ingested the message block $(A_{64}, \ldots, H_{64})$. However, attacking the addition itself (as proposed in [2]) requires of another kind of leakage. In order to respect the constraint to use only $A_i$ and $E_i$ leakages during rounds, author of [19] makes use of the following equations:

$$\begin{aligned}
hash_0 &= A_0 + A_{64} && \Rightarrow && A_{64} = hash_0 - A_0\\
hash_1 &= B_0 + B_{64} = B_0 + A_{63} && \Rightarrow && A_{63} = hash_1 - B_0\\
hash_2 &= C_0 + C_{64} = C_0 + A_{62} && \Rightarrow && A_{62} = hash_2 - C_0\\
hash_3 &= D_0 + D_{64} = D_0 + A_{61} && \Rightarrow && A_{61} = hash_3 - D_0\\
hash_4 &= E_0 + E_{64} && \Rightarrow && E_{64} = hash_4 - E_0\\
hash_5 &= F_0 + F_{64} = F_0 + E_{63} && \Rightarrow && E_{63} = hash_5 - F_0\\
hash_6 &= G_0 + G_{64} = G_0 + E_{62} && \Rightarrow && E_{62} = hash_6 - G_0\\
hash_7 &= H_0 + H_{64} = H_0 + E_{61} && \Rightarrow && E_{61} = hash_7 - H_0
\end{aligned}$$

$hash$ values being the variable known HMAC output split in 8 words. $B_{64}$, $C_{64}$ and $D_{64}$ are related to $A_i$ in the 4 last rounds when $F_{64}$, $G_{64}$ and $H_{64}$ are related to $E_i$ in the 4 last rounds. This allows to target, through a DPA-like attacks, initial values $(A_0, \ldots, H_0)$ when leakages from $A_i$ and $E_i$ computed in the four last rounds are leaking.

It has to be noted that author of [19] details methods to palliate if $A_{64}$ and $E_{64}$ are missing which will not be detailed here.

**Summary of complete attack:** This methodology fixes the partial HMAC attack from [17] in counterpart to the HMAC output that requires now to be known. The constraints of this attack version are:

- Chosen message.
- Known HMAC output.
- Leakage on two end-of-turn stored variables ($A_i$, $E_i$) over four first rounds of inner hash.
- Leakage on two end-of-turn stored variables ($A_i$, $E_i$) over four last rounds of outer hash.
- Four sets of traces.

## 4 Our Contribution

Our contribution lies in two parts. On first hand, we will describe how the choice to swap the roles of the message and the key in HMAC usage can open a very low constraint attack path compared to those of the state-of-the-art. On second hand, we will describe some variations of state-of-the-art attacks on HMAC that could reduce the number of traces needed by 25% or allow the attack even in presence of some countermeasures (partial masking).

### 4.1 The Dangerous Message/Key Swap

In this section, we will show how switching the roles of message and key in HMAC algorithm can seriously endanger the secret's confidentiality. This permutation is, for example, performed in protocols such as HKDF and HMAC-DRBG.

Swapping the roles means that the "Key" from Figure 2 becomes known or controlled by the attacker, when the "Message" becomes a constant value, unknown from the attacker. We will denote Key* the known variable key and Message* the unknown varying message. For sake of clarity of explanations, Message* will be considered as entirely secret and constant, and only $E_i$ leakages will be considered. These assumptions will be relaxed later.

As a consequence, $S_i$ becomes known to the attacker as it can be computed from the known Key* through key padding, then combined by exclusive-or with ipad constant and then mixed in $f$ function with known IV ($\mathrm{IV}_0$). In fact, the whole dashed section of Figure 2 becomes known by the attacker, even if only $S_i$ is used here. On the other side, any $W_i$ derived from Message* block values becomes unknown and constant.

Thus, considering the second block absorption of inner hash, IV value is composed of known variables $(A_0, \ldots, H_0)$ and $W_i$ values are unknown constants. As a reminder, the equation for $E_1$ used in previous attacks was:

$$E_1 = \underbrace{D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0)}_{\delta E_1\text{: Unknown Constant}} + \underbrace{K_0 + W_0}_{\text{Known Variable}}$$

When roles are swapped, this equation becomes:

$$E_1 = \underbrace{D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + K_0}_{\text{Known Variable}} + \underbrace{W_0}_{\text{Unknown Constant}}$$

Our attack is much easier compared to previous ones, as the constant to recover per round is directly part of secret information ($W_i$) instead of the aggregation of constants ($\delta E_1$).

**Step by step description of our attack process:** Based on same leakage assumptions done in [17, 19] we can mount a DPA-like attack recovering any unknown constant ($W_i$ values) in intermediate computations by using the variability of known IV:

1. Using the knowledge of varying Key* to compute, for each acquisition, the value of $S_i$, which is the IV of second compression function ($f$) call of inner hash .

2. Targeting $E_1$ leakage, use the variability of known $S_i = (A_0, \ldots, H_0)$ to recover the value of constant $W_0$:

$$E_1 = \underbrace{D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + K_0}_{\text{Known Variable}} + \underbrace{W_0}_{\text{Unknown Constant}}$$

3. The acquired knowledges of $W_0$ and the one of $(A_0, \ldots, H_0)$ allow to compute the values of $(A_1, \ldots, H_1)$ by applying the Equations (1) to (10).
4. Same as step 2 but targeting $E_2$ leakage and using the variability of known $(A_1, \ldots, H_1)$ to recover $W_1$.
5. Same as step 3 but computing $(A_2, \ldots, H_2)$ from knowledge of $W_1$ and $(A_1, \ldots, H_1)$.
6. By inference, step 2 and 3 can be repeated (with increment on indexes) to recover $W_2$ to $W_{15}$ at respective rounds.

The secret $W_0$ to $W_{15}$ can therefore be recovered. There is no need to go beyond $W_{15}$ as the following values are only combinations of the previous ones.

**Discussing using *A* instead of *E*:** The equation detailed above in step-by-step description only made usage of $E_i$ leakages but the same equations stand with only $A_i$ kind of leakage by using:

$$A_1 = \underbrace{H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + \Sigma_0(A_0) + \mathscr{M}(A_0, B_0, C_0) + K_0}_{\text{Known Variable}} + W_0$$

**Discussing using *A* or/and *E*:** As $A_i$ equations and $E_i$ equations both target the same $W_{i-1}$, both can be used if available to potentially reduce the number of traces needed to ensure that the correct value is recovered.
Also, if leakages are inconsistent from one round to another, roles can be mixed (e.g. if implementation or hardware specificity induces best leakage of $A_i$ on some rounds and best leakage of $E_i$ on other ones).

**Discussing the low variability messages:** If a protocol uses a low variability known input (such as a counter), this could thwart state-of-the-art attacks. Indeed, variability of message is used during DPA-like attack to target the constant secret and a low variability could keep part of the secret out of reach.

However, in our attack scenario, the known input is Key* that is passed through hash process $f$ before using it to attack the secret. Therefore, as soon as there are enough different values that can be used as known input to perform the DPA-like attack, our attack benefits from good entropy properties offered by hash sub-function $f$ and is therefore not affected by low variability input.

**Discussing the secret length:** The example given in step-by-step description considers a secret of size up to the length of one block. However, the method can be pursued in case of longer secret Message*. Indeed, once the first Message* block recovered, the output of the second $f$ call of inner hash can be computed and therefore the IV of next call to function $f$ becomes known and is varying. Then the same process can be applied recovering, any number of Message* blocks until fully recovered.

**Discussing the discontinuity of secret:** If portions of Message* are known (varying or not), the attack can still recover the constant secret part. Indeed:

- if a full $W_i$ word is known (varying or not), the equations still apply and the attacker may directly target the next word $W_{i+1}$ or the next Message* block.
- if a portion of a $W_i$ word is known (varying or not) and another is unknown and constant, then it can be split in two parts $W_i = W_{i,\text{Known}} + W_{i,\text{Unknown}}$. Then the equation becomes:

$$E_1 = \underbrace{D_0 + H_0 + \Sigma_1(E_0) + \mathscr{C}(E_0, F_0, G_0) + K_0 + W_{0,\text{Known}}}_{\text{Known Variable}} + \underbrace{W_{0,\text{Unknown}}}_{\text{Unknown Constant}}$$

and therefore the unknown part can still be recovered.

An example of this situation can be found in the HMAC-DRBG process. A simplification of the HMAC-DRBG process is presented here, for more details please refer to specifications in [16]. In HMAC-DRBG, HMAC are performed in a loop ($\|$ being the concatenation operator):

1. $X$, $Y$ and $Z$ are set to known values
2. $HMAC(X, Y\|Z\|\text{Seed})$
3. $X$ is updated
4. $Y$ is updated
5. Random bits are generated
6. Return to step 2

*Seed* is the main target for an attacker as it is the real secret here. Its recovery allows to predict the generated deterministic bit sequence. $X$ and $Y$ initial values are known but their update can remain non-revealed to an attacker. Thus, if another attack allows to recover some ($X$,$Y$) pairs, our attack can then be applied even in presence of varying content inside the Message* (here $Y$).

**Summary of our attack:** Swapping the roles Message/Key relaxes constraints applied on previous attacks. The constraints applied onto our attack are:

- Swap of roles between message and key.
- Leakage of $A_i$ $\underline{\text{or}}$ $E_i$ intermediate values on rounds corresponding to targeted secret round of appearance.

Constraints per attacks are summarized in Table 1.

**Table 1.** Summary of constraints per attack.

| | Early ([14, 2]) | Partial ([17]) | Complete ([19]) | Our Method |
|---|---|---|---|---|
| Input with high entropy | X | | | |
| Leakage of $A$ and $E$ | X | X | X | |
| Other leakages types | X | | | |
| Chosen input | | X | X | |
| Several set of traces | | X | X | |
| Known HMAC output | | | X | |
| Leakage of $A$ <u>or</u> $E$ | | | | X |
| Swap Message/key roles | | | | X |

**Applicability to SHA-1:** Our attack presented here takes advantage of the way the message part ($W_i$) is mixed with IV ($A_0, \ldots, H_0$) in SHA-2. As this algorithm has huge design similarities with its predecessor SHA-1, the applicability of our method on this algorithm can be questioned. Here, briefs extracts from specifications of SHA-1 [15] are used. The notations are kept but equations that transform values are different. The attack is found applicable on SHA-1 by using:

$$A_1 = \underbrace{\text{Rotation}(A_0) + \mathscr{C}(B_0, C_0, D_0) + E_0 + K_0}_{\text{Known Variable}} + \underbrace{W_0}_{\text{Unknown Constant}}$$

## 4.2 Simulations

Previous work in [19] has already described a use case where no leakage can be found, except for $A_i$ and $E_i$ intermediate values. Author has shown that such a leakage can be used to perform an attack.

We still performed simulations to ensure the viability of our attack process. The simulations were done on HMAC-SHA-2 algorithm, with hash size output of 256-bit. First, simulated traces are created:

- 5,000 HMAC executions are performed with varying Key* and fixed Message*. For each execution, 32-bit intermediate values $E_i$ are gathered for each round of each call to $f$ function inside of inner hash of HMAC.
- For each execution, the Hamming weight of gathered 32-bit words are computed and concatenated in order to produce one simulation trace.
- A random normal noise with scale $\sigma$ is added to each trace in order to simulate measurement noise.

Then, we simulate an attacker that have $N$ traces available:

- Value of known Key* are used to compute $S_i = (A_0, \ldots, H_0)$ values for N traces. $\delta E_1$ can then be computed.
- DPA-like attack (here CPA) is performed on $N$ first traces of the set, using the least significant byte of $\delta E_1$ variability to target the least significant

byte of first 32-bit word of Message* ($W_0$). Ranks of candidates were logged during the attack.

- The best ranking candidate (correct or not) is taken to perform carry computation and now another attack of same kind is performed on next least significant byte of $W_0$. The process is repeated for the 4 bytes of $W_0$.
- Eventually a best ranking candidate for $W_0$ (correct or not) is obtained, it can be combined with $(A_0, \dots, H_0)$ to compute $(A_1, \dots, H_1)$ as explained in our attack process. It must be noted that the candidate for $W_0$ can be incorrect if trace number is insufficient to correctly distinguish the good candidate. Therefore, an error will propagate to value of $(A_1, \dots, H_1)$.
- The process can be repeated to target following $W_i$ if necessary.

This attack simulation was performed for various $N$ and $\sigma$ values in order to control the attack feasibility and the number of traces needed for recovery. The simulations were positive to confirm the viability of the method: with $\sigma = 5$, nearly 400 traces were needed to recover the correct value for $W_0$, $W_1$ and $W_2$. The number raised to nearly $1,200$ traces for $\sigma = 10$ and nearly $3,500$ traces for $\sigma = 15$. Figures of attack result per byte are given in Appendix A.

## 4.3 Optimization of State-of-the-Art techniques on HMAC

We also provide two tricks to enhance previous work of state-of-the-art. These tricks do not apply to our new attack methodology presented in Section 4.1.

**25% decrease of number of traces:** An optimization can be realized in attacks on HMAC performed in [17,19]. As described in 3.2, the attack on inner hash requires that 4 sets are acquired. However, we point out that the fourth is not necessary, thus reducing the number of required traces.

The first set is required to start the attack and use variability of $W_0$ to attack $\delta E_1$ that is constant.

The second set is needed in order to fix $W_0$ to have a constant $E_1 = E_1^*$. Looking at Equation 21, the need to have a constant $E_1^*$ comes solely from choice function ($\mathscr{C}$). Indeed, $E_1$ known and variable is not a problem for $\Sigma_1(E_1)$ that is also a known variable and could have been transferred with $W_1$ out of the equation. As the attacker cannot process $\mathscr{C}(E_1, E_0, F_0)$, due to the ignorance of $E_0$ and $F_0$, this part becomes an unknown variable that thwarts DPA-like attacks.

The third set is needed for the exact same reasons. Looking at Equations 23, choice is again the problem here as if $E_2$ is kept variable, $\mathscr{C}(E_2, E_1^*, E_0)$ becomes a unknown variable due to the ignorance of $E_0$.

However for the next round, there is no need of a fourth set, the final attack can be done on the third one. Indeed, in Equation 25, if $E_3$ is kept as a known variable, then $\mathscr{C}(E_3, E_2^*, E_1^*)$ is also a known variable and can then be transferred with $\Sigma_1(E_3)$ to the known variable part of the equation. In summary, the equation:

$$E_4 = \underbrace{A_0 + E_0 + \Sigma_1(E_3^*) + \mathscr{C}(E_3^*, E_2^*, E_1^*)}_{\text{Unknown Constant}} + \underbrace{K_3 + W_3}_{\text{Known Variable}}$$

becomes:

$$E_4 = \underbrace{A_0 + E_0}_{\text{Unknown Constant}} + \underbrace{\Sigma_1(E_3) + \mathscr{C}(E_3, E_2^*, E_1^*) + K_3 + W_3}_{\text{Known Variable}}$$

The exact same process can be applied on equation of $A_4$. Therefore, the attack can be done on 3 sets instead of 4 reducing the number of trace by $\sim 25\%$.

**Shifting start of attack:** As state-of-the-art attacks require leakages of intermediate values on rounds of hash function, a protection against SCA can be applied to mitigate/remove these leakages. As state-of-the-art attacks detailed in Section 3 need leakages only on first rounds, a choice can be made, for cost reduction purposes, to apply the protection only on first rounds.

In such a situation of partial protection and in chosen message context, we suggest to start the attacks on later rounds. As an example, if the $n$ first rounds of hash function $f$ are protected but the following are not, one can fix $W_0$ to $W_{n-1}$ to thwart the protection. Doing so, the input of $(n+1)^{th}$ round $(A_n, \ldots, H_n)$ are fixed and unknown and can be targeted by the same methodology as $(A_0, \ldots, H_0)$ could be. Once $(A_n, \ldots, H_n)$ recovered, and with knowledge of $W_{n-1}$, Equations (1) to (10) can be inverted to recover $(A_{n-1}, \ldots, H_{n-1})$:

$$\begin{aligned}
A_{n-1} &= B_n \\
B_{n-1} &= C_n \\
C_{n-1} &= D_n \\
T2_{n-1} &= \Sigma_0(A_{n-1}) + \mathscr{M}(A_{n-1}, B_{n-1}, C_{n-1}) \\
T1_{n-1} &= A_n - T2_{n-1} \\
D_{n-1} &= E_n - T1_{n-1} \\
E_{n-1} &= F_n \\
F_{n-1} &= G_n \\
G_{n-1} &= H_n \\
H_{n-1} &= T1_{n-1} - \Sigma_1(E_{n-1}) - \mathscr{C}(E_{n-1}, F_{n-1}, G_{n-1}) - K_{n-1} - W_{n-1}
\end{aligned}$$

This process can be done again on previous rounds and thanks to knowledge of $W_{n-1}$ to $W_0$, this allows to recover the targeted $(A_0, \ldots, H_0)$.

The limitation of this method is the control of $W_i$ by attacker as only the first 16 words can be controlled[2]. Therefore, we recommend to mask at least the 16 first rounds to protect from this method.

[2] For SHA-2 algorithm.

## 5 Conclusion

After a detailed summary of state-of-the-art of DPA-like attacks against HMAC, we have shown that some protocols make a dangerous usage of HMAC by swapping the role of the message and the key. We conclude that this choice can drastically reduce the constraints on attacker, which can ease attack on such protocol, even in case of reduced leakages opportunities.

Moreover, we describe two methods that can optimize state-of-the-art attacks. One by reducing by $\sim 25\%$ the number of trace of some attacks. The second allowing to adapt existing attacks even in case a partial masking countermeasure is applied.

This work implies to further study the consequences of alternative usage of algorithms outside the way they are specified. As an open subject, adaptations of attacks to SHA-3 (and corresponding HMAC) should also be investigated as it does not share the same common design of SHA-1 and SHA-2.

## A Simulations Results: Attack Score per Number of Traces

As described in Section 4.2, simulations have been run to verify the feasability of the attack methodology described in this paper. Each sub-plot shows the evolution of the CPA score obtained per candidate for one targeted byte. Bytes are the ones from 32-bit words $W_0$, $W_1$ and $W_2$ values targeted during simulations. The good candidate is in red/plus sign marked line while wrong candidates are blue/straight lines. Figure 3 shows results when noise level is $\sigma = 5$ and Figure 4 shows results when noise level is $\sigma = 15$.

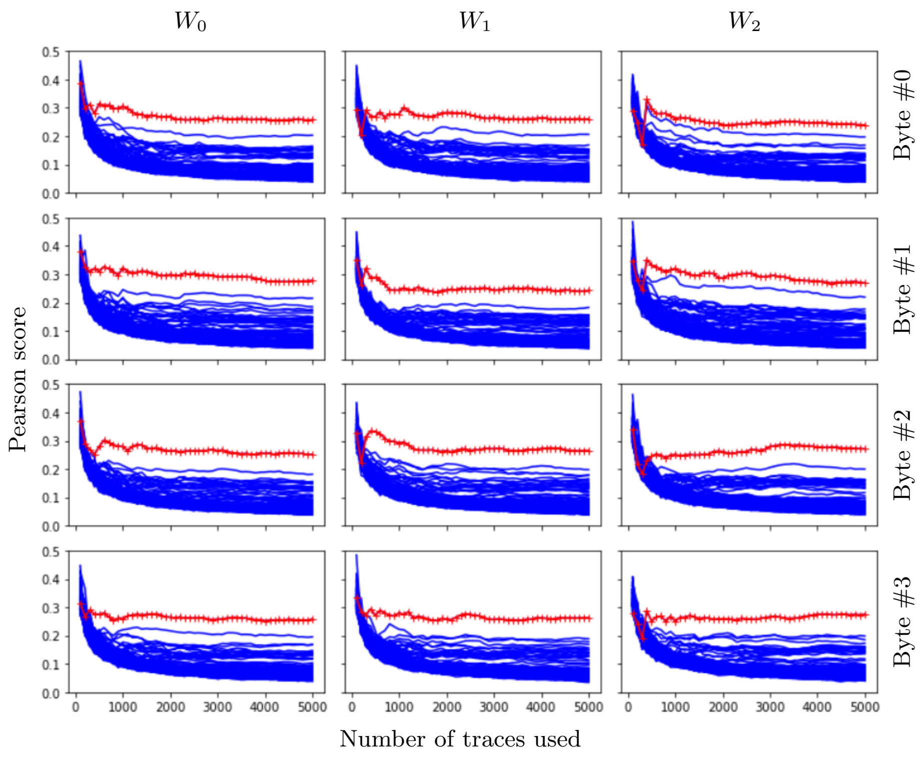

**Fig. 3.** Simulation score evolution per byte, noise $\sigma = 5$

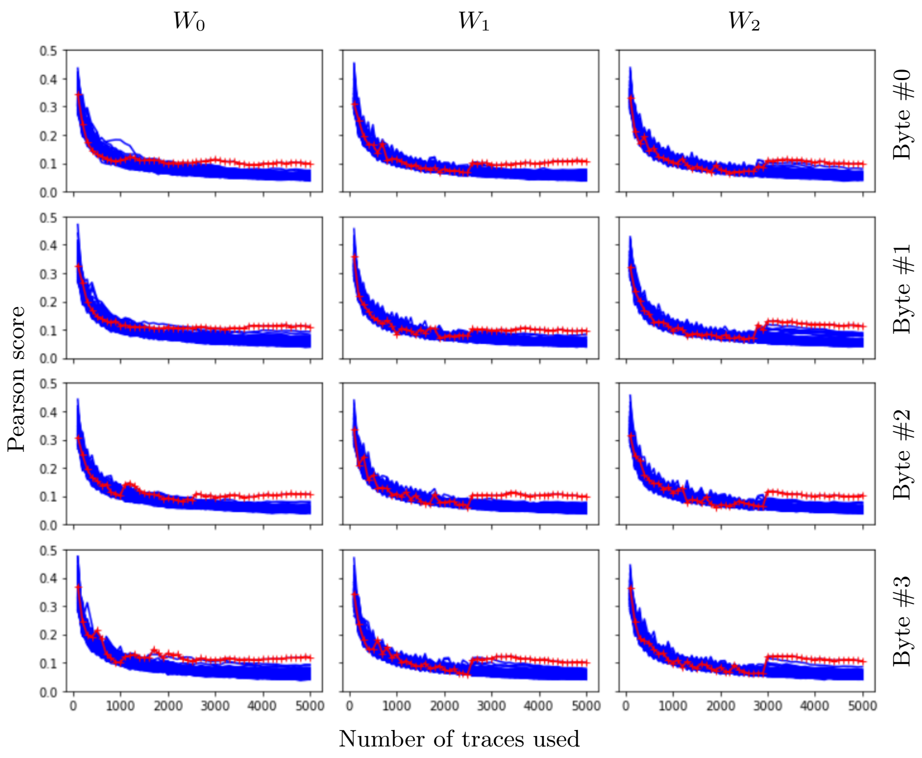

**Fig. 4.** Simulation score evolution per byte, noise $\sigma = 15$

## References

1. Barnes, R.L., Bhargavan, K., Lipp, B., Wood, C.A.: Hybrid public key encryption. RFC **9180**, 1–107 (2022). https://doi.org/10.17487/RFC9180, https://doi.org/10.17487/RFC9180
2. Belaïd, S., Bettale, L., Dottax, E., Genelle, L., Rondepierre, F.: Differential power analysis of HMAC SHA-2 in the hamming weight model. In: Samarati, P. (ed.) SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography, Reykjavík, Iceland, 29-31 July, 2013. pp. 230–241. SciTePress (2013), https://ieeexplore.ieee.org/document/7223170/
3. Belenky, Y., Dushar, I., Teper, V., Chernyshchyk, H., Azriel, L., Kreimer, Y.: First full-fledged side channel attack on HMAC-SHA-2. In: Bhasin, S., Santis, F.D. (eds.) Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12910, pp. 31–52. Springer (2021). https://doi.org/10.1007/978-3-030-89915-8“˙2, https://doi.org/10.1007/978-3-030-89915-8_2
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO ’96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer (1996). https://doi.org/10.1007/3-540-68697-5“˙1, https://doi.org/10.1007/3-540-68697-5_1
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004). https://doi.org/10.1007/978-3-540-28632-5“˙2, https://doi.org/10.1007/978-3-540-28632-5_2
6. Ferrigno, J., Hlavác, M.: When AES blinks: introducing optical side channel. IET Inf. Secur. **2**(3), 94–98 (2008). https://doi.org/10.1049/IET-IFS:20080038, https://doi.org/10.1049/iet-ifs:20080038
7. Fischlin, M., Janson, C., Mazaheri, S.: Backdoored hash functions: Immunizing HMAC and HKDF. In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018. pp. 105–118. IEEE Computer Society (2018). https://doi.org/10.1109/CSF.2018.00015, https://doi.org/10.1109/CSF.2018.00015
8. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer (2001). https://doi.org/10.1007/3-540-44709-1“˙21, https://doi.org/10.1007/3-540-44709-1_21
9. Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low-bandwidth acoustic cryptanalysis. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 444–461. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2“˙25, https://doi.org/10.1007/978-3-662-44371-2_25

10. Housley, R.: Use of the elliptic curve diffie-hellman key agreement algorithm with X25519 and X448 in the cryptographic message syntax (CMS). RFC **8418**, 1–18 (2018). https://doi.org/10.17487/RFC8418, https://doi.org/10.17487/RFC8418
11. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5"̇9, https://doi.org/10.1007/3-540-68697-5_9
12. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1"̇25, https://doi.org/10.1007/3-540-48405-1_25
13. Krawczyk, H., Eronen, P.: Hmac-based extract-and-expand key derivation function (HKDF). RFC **5869**, 1–14 (2010). https://doi.org/10.17487/RFC5869, https://doi.org/10.17487/RFC5869
14. McEvoy, R.P., Tunstall, M., Murphy, C.C., Marnane, W.P.: Differential power analysis of HMAC based on sha-2, and countermeasures. In: Kim, S., Yung, M., Lee, H. (eds.) Information Security Applications, 8th International Workshop, WISA 2007, Jeju Island, Korea, August 27-29, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4867, pp. 317–332. Springer (2007). https://doi.org/10.1007/978-3-540-77535-5"̇23, https://doi.org/10.1007/978-3-540-77535-5_23
15. NIST: Fips pub 180-4 secure hash standard (shs). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (2015), accessed: Dec 10 2024
16. NIST: Recommendation for random number generation using deterministic random bit generators. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf (2015), accessed: Dec 10 2024
17. Pankaj Rohatgi and Mark Marson: Nsa suite b crypto, keys, and side channel attacks. https://www.rambus.com/wp-content/uploads/2015/08/2013-JunMarson-SuiteBAndSideChannel.pdf (2013), accessed: Dec 10 2024
18. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC **8446**, 1–160 (2018). https://doi.org/10.17487/RFC8446, https://doi.org/10.17487/RFC8446
19. Schuhmacher, F.: Canonical DPA attack on HMAC-SHA1/SHA2. In: Balasch, J., O'Flynn, C. (eds.) Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13211, pp. 193–211. Springer (2022). https://doi.org/10.1007/978-3-030-99766-3"̇9, https://doi.org/10.1007/978-3-030-99766-3_9