Collision-Correlation Attack against some 1^{st}-order Boolean Masking Schemes in the Context of Secure Devices

Thomas Roche and Victor Lomné

ANSSI
51 boulevard de la Tour-Maubourg, 75700 Paris 07 SP, France
firstname.lastname@ssi.gouv.fr

Abstract. In this paper we study the collision-correlation attack published by Clavier *et al.* at CHES 2011 on a 1^{st}-order boolean masking scheme and show its lack of robustness against unknown and high level of measurement noise. In order to improve the attack, we follow the approach that Gérard and Standaert proposed in a recent paper at CHES 2012. Then we address the problem of heterogeneous leakage pointed out by Gérard and Standaert (when the leakage noise is different from one Sbox output to the others due for instance to implementation particularities or resynchronisation reasons), by inserting an efficient termination algorithm in the key-recovery phase of the attack. In a last contribution, we compare (over simulations and real experiments) the enhanced collision-correlation attack and the 2^{nd}-order CPA attack. Similarly to the results of Gérard and Standaert, we show – in the context of masked implementations – the superiority of 2^{nd}-order CPA when its leakage model is not too far from the real leakage function.

Key words: AES, Side-Channel Analysis, Collision Attack, 2^{nd}-order CPA, Masking Scheme

1 Introduction

It is today well-known that cryptographic devices are susceptible to Side-Channel Analysis (SCA). Indeed, computation time [19], power consumption [20] or electromagnetic radiations [13] of an embedded system performing a cryptographic operation leak information about the secret involved in the computation. Various attack methods have been proposed to exploit these side-channel information, the most popular being *Simple Side-Channel Analysis* (SSCA) [20], *Differential Side-Channel Analysis* (DSCA) [20] and *Template Attacks* (TA) [9]. Among these methods, DSCA is particularly devastating: the adversary model is not too restrictive and the attacks are robust to realistic noise levels, an inevitable component in SCA attacks.

Several types of countermeasures have been proposed to thwart DSCA, *e.g.* the use of jittered clock [10], the insertion of random delays [1], the shuffling of operations, the use of dedicated logic styles aiming at *hiding* the side-channel leakages (*e.g.* WDDL [30], MDPL [26]) or masking techniques [8,16].

Masking techniques have become popular as their soundness can be formally proven (see [8]). A *masking scheme* transforms a cryptographic algorithm: each intermediate variable (referred to as *sensitive variables* in the sequel) is shared — by means of random *masks* — such that each share alone is independent of the secret. Nevertheless it has been observed that such countermeasures succumb to higher-order DSCA, where the attacker combines the leakage of several internal variables (typically, the two shares of a shared sensitive variable in a 2nd-order DSCA). Moreover, a sound 1st-order masking scheme induces a non-negligible overhead on the computational cost and developers usually design *light* versions of masking schemes.

We are interested here in a very common 1st-order masking scheme [22] and its security against SCA. To study such a scheme, we use as an example the AES cipher [12], hence `SubBytes` will denote the non-linear layer composed of 16 8-bit Sboxes. The basic idea of the masking scheme is to pre-compute a unique masked Sbox, $\widehat{\text{Sbox}}$, for each cipher execution, such that $\forall x \in \mathbb{F}_{2^8}, \widehat{\text{Sbox}}(x) = \text{Sbox}(x \oplus m) \oplus m'$ with m and m' two random bytes. Then, during the cryptographic operation, each byte of the `SubBytes` transformation input is masked by the same mask value m, allowing to use $\widehat{\texttt{Sbox}}$ for each Sbox look-up operation (see [21] for a complete description of the masking scheme). When correctly implemented, such a masking scheme is perfectly masked at the 1st-order (*i.e.* no univariate side-channel leakage depends on the secret), and is considered the most efficient software implementation of 1st-order AES masking scheme on 8-bit CPUs (see for instance [14]). In the following we will often refer to this masking scheme as the *Mask Reuse Scheme*.

Collision-based SCA denotes a type of attacks that do not rely on an *a priori* knowledge on the device leakage function (whereas it is the case in (HO-)DSCA-like attacks). The general idea is to use side-channel information to detect a collision between two cipher sensitive variables (see [2–6, 24, 28, 29]). Moreover they can naturally be applied against several masking techniques where masks are reused (*e.g.* [11]). Even though the collision-based SCA attacks are leakage function oblivious, most of the attacks found in the literature rely on other device dependent parameters. In a recent article, Gérard and Standaert [15], tackle this issue on the linear collision attack of Bogdanov [3,4] targeting unprotected AES implementations. They show that, when reducing the knowledge on the device leakage, the collision attack becomes much less interesting that expected (in their practical setup, the CPA is always better than collision-based SCA attacks).

This is also what we observed on a protected implementation with an enhanced version of the *collision-correlation* attack proposed by Clavier *et al.* [11]. The main idea of the attack (first due to Moradi *et al.* [24]) is to detect, through side-channel, the collision of two Sbox outputs in the first round of AES. To this purpose, the attacker computes the Pearson correlation between the two corresponding leakages $((L_a)_{i \leq N}, (L_b)_{i \leq N})$ acquired from N successive cipher executions with chosen plaintexts. If the key guess is correct, the chosen plaintexts will always lead to the same Sbox output value $(z_a = z_b)_{i \leq N}$ and then the correlation coefficient will be high. The main drawback of Clavier *et al.* attack is that it uses a fixed threshold value to distinguish high correlation coefficient from low ones. In a very similar context, Bogdanov uses an heuristic algorithm for collision detection [4]. Figure 1 illustrates this issue when the noise grows. Indeed, it shows the difference between two correlation coefficients of two leakage sources (simulated with the classical Hamming Weight model and an additive Gaussian noise on two Sbox outputs) with respect to the noise standard deviation. The best correlation value (upper curve) corresponds to a correct key guess (the two intermediate variables are equal), whereas the second correlation (lower curve) corresponds to a case where the key guess is not correct, hence there is not always collision. In Clavier *et al.*, the threshold value must be set between the two curves. It is obvious that, when the noise becomes high (which is often true when attacking secure devices), fixing such threshold *a priori* is equivalent to precisely know the noise level (hence implies a profiled step in the attack).

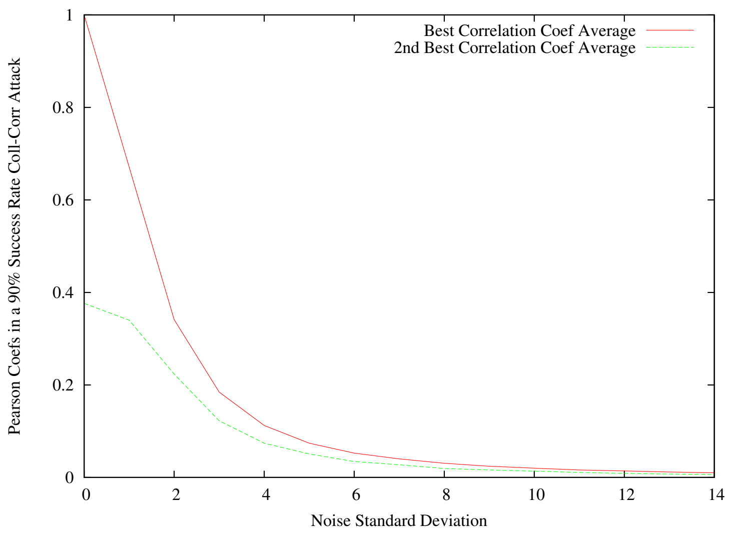


Fig. 1. Set of threshold values T with respect to σ: $\rho\Big(HW(z_k) + \mathcal{N}(0,\sigma), HW(z_k) + \mathcal{N}(0,\sigma)\Big) > T > \rho\Big(HW(z_k) + \mathcal{N}(0,\sigma), HW(z_{\tilde{k}}) + \mathcal{N}(0,\sigma)\Big)$

As mentioned before, Gérard and Standaert [15] propose to solve this issue – in the un-masked context – using Bayesian extensions and Low Density Parity Check codes soft-decoding.

2 Our Contributions

Our first contribution is to propose a collision-correlation attack on the considered mask reuse scheme when the noise is unknown and high. To that purpose we will follow the approach of Gérard and Standaert [15].

In [15], the authors observe that their approach does not take into account the problem of heterogeneous leakage: the leakage noise is different from one Sbox output to the others due to implementation particularities or resynchronisation reasons.

In a second contribution, we address this issue by inserting an efficient termination algorithm in the key-recovery step of the attack.

In our last contribution, we compare (over simulations and real experiments) collision-correlation attack and 2^{nd}-order CPA. The 2^{nd}-order CPA is shown to be more efficient than collision-correlation attack when no LDPC soft-decoding is used and when the leakage function is close to the CPA leakage model (Hamming Weight here). However, since to our knowledge, no Bayesian extension has been devised for 2^{nd}-order CPA, collision-correlation takes the lead (in our experiments) when improved with such techniques.

The rest of the paper is organised as follows. Section 3 describes the collision-correlation attack and the improvements we propose. Section 4 compares the efficiency of the collision-correlation attack with the 2^{nd}-order CPA attack. Finally section 5 concludes this work.

3 Collision-Correlation Attack

We propose to describe the collision-correlation attack in two steps, first the collision detection mechanism, then the key recovery phase.

Notation. In the following, we will denote respectively by p, c, k, z and m the plaintext, the ciphertext, the secret key, the target sensitive variable and the mask. All these values are over 16 bytes (since we consider the AES cipher). We will frequently use their vector representation over $(\mathrm{GF}(2^8))^{16}$ (*e.g.* $p = (p_0, \cdots, p_{15})$). Moreover, if N encryptions are considered, we will denote the i^{th} plaintext by p^i.

3.1 DetectCollision

The detection of collision is a critical component of any collision-based SCA. In our context, one needs to detect a collision between the manipulation of two bytes during the AES first round. As mentioned in Section 1, we will focus on two Sbox outputs $(z_a \oplus m, z_b \oplus m)$ of the masked AES implementation. Over a set of N plaintexts $(p^i)_{i \leq N}$, the `DetectCollision` function must detect, for some index i, the value $\alpha = p_a^i \oplus p_b^i$ such that the two sensitive variables collide ($z_a^i \oplus m^i = z_b^i \oplus m^i$) from the N side-channel leakage pairs $(L_a^i, L_b^i)_{\{i \leq N\}}$ (L_s^i corresponding to the side-channel leakage generated by the manipulation of $z_s^i \oplus m^i$). As a matter of fact, the value of α is of interest: if $z_a^i = z_b^i$, then $Sbox(p_a^i \oplus k_a) = Sbox(p_b^i \oplus k_b)$, implying $p_a^i \oplus p_b^i = \alpha = k_a \oplus k_b$.

The basic idea of collision-correlation attack proposed by Clavier *et al.* in [11] is recalled in Section 1. In order to adapt the collision-correlation attack in a more robust context, we propose the following scenario: the attacker encrypts 256 sets $\{S_\alpha(a,b)\}_{\alpha<256}$ of N plaintexts such that for every $\alpha < 256$, every $i \leq N$, $p^{i,\alpha} \in S_\alpha(a,b)$ is such that $p_a^{i,\alpha} \oplus p_b^{i,\alpha} = \alpha$. That way, one may compute, for each $\alpha < 256$, the correlation coefficient:

$$\rho_\alpha = \rho((L_a^{i,\alpha})_{i\leq N}, (L_b^{i,\alpha})_{i\leq N}) \ ,$$

and then compare them to each other in order to take the decision: $k_a \oplus k_b = \operatorname{argmax}(\rho_\alpha)$. This strategy is more expensive than the attack described in [11], however it is not based on a hypothetical knowledge of the attacker (*i.e.* a target specific threshold), which is a necessary condition for a fair comparison with non-profiled attacks such as 2nd-order DSCA. The detection algorithm was presented in a chosen plaintext scenario, however it is easy to see that, when the number of plaintexts is large enough, a known plaintext setting will be equivalent (when the plaintexts follow an uniform distribution each of the 256 sets would have the same number N of plaintexts, in first approximation). A formal study of the efficiency of this DetectCollision with respect to different attack scenarios (known plaintext vs. chosen plaintext scenarios) is proposed in Annex A. Algorithm 1 describes the different steps of the `DetectCollision` mechanism.

Algorithm 1 `DetectCollision` in a known plaintexts setting

INPUT: $(p_a^i)_{i\leq 256\cdot N}, (p_b^i)_{i\leq 256\cdot N}, (L_a^i)_{i\leq 256\cdot N}, (L_b^i)_{i\leq 256\cdot N}$

OUTPUT: 2 key bytes difference: $\alpha = k_a \oplus k_b$

```
1. for $i = 1$ to $256 \cdot N$
2.     do send the index $i$ in the set $S_{p_a^i \oplus p_b^i}$
           [If the $(p^i)_{i\leq 256\cdot N}$ are uniformly distributed, about $N$ indexes will be sent to
         each $S_j$]
3. for $\hat{\alpha} = 0$ to 255
4.     do $\rho_{\hat{\alpha}} \leftarrow \rho((L_a^i)_{i\in S_{\hat{\alpha}}}, (L_b^i)_{i\in S_{\hat{\alpha}}})$
           [Evaluate the Pearson Correlation Coefficient for the hypothesis $\hat{\alpha} = k_a \oplus k_b$]
5. return $\alpha = \operatorname{argmax}(\rho_{\hat{\alpha}})$
```

3.2 Key-Recovery

Bayesian Extension and LDPC Soft-Decoding The above proposed collision detection mechanism based on correlation is actually very similar to the treatment of Gérard and Standaert in [15] when adapted to the context of attacking a masking scheme (the main difference being that no inter-traces correlations or trace averaging can be used in our context). Hence, the Bayesian extension approach in order to use a LDPC soft-decoding algorithm can be directly applied here and will replace the step 5 of Algorithm 1. For reasons of completeness we recall briefly the results of [15]; details and approach argumentation can be found in the original paper of Gérard and Standaert.

The Bayesian extension from the correlation evaluations $\rho_{\hat{\alpha}}$ can be written as follows:

$$\Pr[k_a \oplus k_b = \hat{\alpha} | \operatorname{arctanh}(\operatorname{Norm}(\rho_{\hat{\alpha}})) = s] = \operatorname{Norm}(e^{2s}) \ . \tag{1}$$

Equation 1 comes from many approximations and has no rigorous justification in [15] otherwise that being shown to work well in simulations and real experiments. We also observed this success and let for further studies the formal justification of the formula[1].

From the Bayesian extension of the correlation evaluation, the use of *Low Density Parity Check* codes (denoted as LDPC in the following) as soft-decoding technique is shown to fit the problem of retrieving a correct set of 120 equations. The decoding algorithm is recalled below.

Algorithm 2 `LDPCSoftDecoding` procedure (Alg. 2 in [15])

Input: The distributions $\Pr[k_a \oplus k_b = \hat{\alpha} | \rho_{\hat{\alpha}}]$
Output: The likeliest consistent system S

```
1. for $0 \le a < b \le 15, \alpha \in \mathrm{GF}(256)$
2.     do $P_{a,b}(\alpha) \mapsto \Pr[k_a \oplus k_b = \alpha]$
3. while $(\mathrm{argmax}_\alpha P_{0,1}(\alpha), \cdots, \mathrm{argmax}_\alpha P_{14,15}(\alpha))$ is not a codeword
4.     for $0 \le a < b \le 15, \alpha \in \mathrm{GF}(256)$
5.         do $P_{a,b}(\alpha) \mapsto P_{a,b}(\alpha) \cdot \prod_{c \notin \{a,b\}} \sum_{\beta \in \mathrm{GF}(256)} P_{a,c}(\beta) \times P_{b,c}(\beta \oplus \alpha)$
6. return $(\mathrm{argmax}_\alpha P_{0,1}(\alpha), \cdots, \mathrm{argmax}_\alpha P_{14,15}(\alpha))$
```

As remarked by the authors of [15], the LDPC soft-decoding does not perfectly match the cases where the target bytes do not leak homogeneously. They observe the phenomenon on an optimized implementation (the "Furious AES", see [15] for more details), we also observe the same problem with our setup. As a matter of fact, one of the hypothesis that lead to equation 1 is an independent Gaussian noise with same mean and standard deviation on each target Sbox output leakage. When this hypothesis is not verified, we observe that Algorithm 2 may finish with a worst system than the initial one. In [15], the issue is patched with an ad-hoc list-decoding algorithm. We propose here a generic post-treatment algorithm that solve the issue efficiently (if not optimally).

Termination Algorithm In step 3 of Algorithm 2, instead of waiting for a complete codeword, a hard-decoding algorithm is applied to the vector

$$(\mathrm{argmax}_\alpha P_{0,1}(\alpha), \cdots, \mathrm{argmax}_\alpha P_{14,15}(\alpha))$$

[1] As described in Annex C.2, the direct application of this Bayesian extension to the 2nd-order CPA attack does not work properly

and the best candidate keys are tested when enough key bits are retrieved. This process is detailed below.

Solving this equation system is done in two steps, in a first step the erroneous equations (i.e. failed collision detection) are identified and corrected if possible. In the second step, when only correct equations are left, the number of missing bits of information about the secret key is evaluated. If the number of missing bits is not too large (we choose to set the limit to 40 bits) the secret key is then retrieved by brute force on these bits. We insist here that the algorithm points out the missing bits; their location being known, the brute force is then realistic.

Detection/Correction of Erroneous Equations It can be easily checked that the equation system of 120 equations possesses 16 unknowns and a rank of 15 (when all the equations are correct, *cf.* [3]). Hence, 120-15 of these equations may be considered as redundant and then used to detect, and eventually correct, some potential erroneous equations.

Algorithm 3 describes the so-called decoding algorithm, the main idea of the algorithm is to look for triangular relations of the following form, considering three key indexes a, b, c:

$$(k_a \oplus k_b) = (k_a \oplus k_c) \oplus (k_c \oplus k_b) \enspace .$$

Algorithm 3 `SolveSystem_Coll-Corr`

Input: A noisy equation system S_{in}
Output: A clean equation system S_{out}

```
1. Initialize an Empty Equation System $S_{out}$
2. for $a = 0$ to 15
3.     for $b = a + 1$ to 15
4.           do Find the value $\alpha_{max} = (k_a \oplus k_c) \oplus (k_c \oplus k_b)$
                 that appears the most often over all indexes of $c$.
5.          if $\alpha_{max}$ appeared more than 3 times
                 then do set the equation $k_a \oplus k_b = \alpha_{max}$ into $S_{out}$.
6. return $S_{out}$
```

Remark 1. The threshold value 3 appearing in the algorithm is not arbitrary fixed, it is the minimum threshold that will distinguish some erroneous equations from the others: α_{max} appears at least two times since for $k_c = k_a$ and $k_c = k_b$, α is the same.

Solving the System When the equation system is purged of the erroneous equations, the secret key cannot be directly recovered, there is some unknown bits left (at least 8 since the rank of the system is 15 for 16 unknown bytes).

In fact, the same equation system was studied in length in [3,5]. The chance for the attacker to be able to solve it (*i.e.* the chance that the number of unknown bits left is below 40), was computed as a function of the number of equations in the system.

We give the simulation results of the algorithm in Section 4.1 and compare them to a similar system solving success rate in the case of the 2nd-order CPA attack (introduced in Annex C). Using Algorithm 3, the equation system is correctly solved with a success probability above 90% when only 40% of the equations are correct. In Section 4.2, we also show that the simulation results match our experimental results with respect to the key-recovery, *i.e.* the repartition of erroneous equations in the system does not have a special structure that helps / handicaps the detection algorithm.

3.3 Full Attack Algorithm in a Known Plaintext Setting

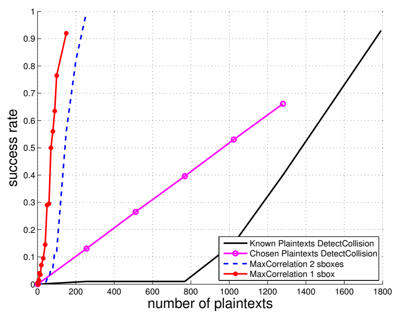
We have seen different improvements of the basic collision-correlation attack (that would be mainly composed of the `DetectCollision` algorithm), indeed, the attack could skip the LDPC soft-decoding and the Bayesian extension approach altogether and feed directly the `SolveSystem_Coll-Corr` algorithm from the outputs of the `DetectCollision` algorithm applied at each pair of Sbox outputs. We will see in our real setup that the use of the LDPC soft-decoding will greatly improve the data complexity of the attack. The data complexity evaluation of the attacks is conducted in Section 4 and is compared to the 2nd-order CPA (recalled in Annex C).

4 Success Rate Comparison

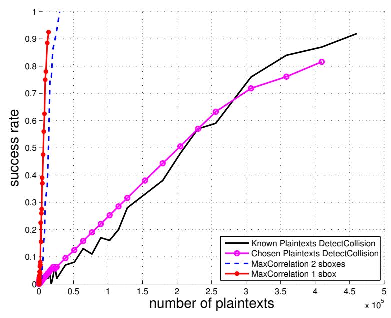
4.1 Attack Simulation in the Hamming Weight Model

In the previous section, a collision-correlation method has been described, allowing to defeat some 1st-order masking schemes of AES (exposed in Section 1). Moreover, in Annex C, we added a complete description of 2nd-order CPA attack. For each of the attacks, a first phase consists in attacking all the possible key byte pairs independently and outputs an overdetermined equation system. The second phase uses simple and efficient algorithms to solve/correct the obtained equation systems. This way, comparisons between the two attack methods can be directed at two different levels of the process. Firstly, we want to know how the first phase of each method deals with the noise to guess a pair of key bytes. Secondly, taking into account the probability of correctly guessing a key byte pair, we want to know how each system solving algorithm will behave with respect to the partially erroneous equation system and retrieve the correct full key.

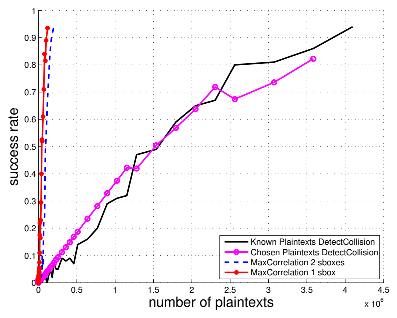
2-Byte Attack Success-Rates In order to evaluate how, for each attack method, the first phase deals with the noise to guess a pair of key bytes, several experiments have been performed by simulation. Assuming an additive Gaussian noise of mean 0 and standard deviation σ, we have modeled the leakage of two Sbox outputs following a Hamming Weight leakage model. Then we have applied, on the one hand, the `MaxCorrelation` (Algorithm 4, Annex C) function on the two leakages, and on the other hand, the `DetectCollision` function (Algorithm 1, Section 3). For comparison reasons we also added the success rate of the classical 2nd-order CPA that targets one byte at a time (see Annex C). In each case, 100 simulations have been performed, with random plaintexts and random subkeys. Figure 2 shows the evolution of the success rate with respect to the number of plaintexts for both algorithms, assuming different levels of noise (*i.e.* different values of σ). Figure 2(a) considers a perfect Hamming Weight leakage model without noise, while 2(b) and 2(c) relate to a more realistic noise range, corresponding on what has been observed in the real experiment described in Section 4.2.



(a) Noise standard deviation: 0



(b) Noise standard deviation: 4



(c) Noise standard deviation: 7

Fig. 2. Evolution of the success rate *w.r.t.* the number of plaintexts for the `MaxCorrelation` and the `DetectCollision` algorithms applied on two Sbox outputs in simulation

Superiority of the 2nd-order CPA. The results in simulations are clear: the collision-correlation attack is much less efficient than the 2nd-order CPA. The collision-correlation attack pays the price that each plaintext is dedicated to a unique subkey candidate and thus its leakage trace cannot be used in the process of comparing two different key candidates (by opposition to CPA). Even though the 1-Byte 2nd-order CPA [2] has a better success rate than the 2-Byte 2nd-order CPA [3], we must recall that the latter one retrieve 2 bytes of key at a time when the first one can only retrieve 1 byte of key at a time.

Nevertheless, one can note that the Hamming Weight model used for the simulations is certainly helping 2nd-order CPA attacks. As a matter of fact, in our simulation, the CPA predictions perfectly match the leakage function. In fact this is the ideal situation to use CPA, and the real experiments permit to diminish a bit the superiority of 2nd-order CPA over collision-correlation attacks (see Section 4.2).

Known vs. Chosen Plaintexts in Collision-Correlation Attacks. We displayed in Figure 2 the success rate for both known and chosen plaintexts attacks. At this level (when focusing on two bytes only), there is no difference between the two mentioned chosen plaintexts scenarios (Annex A). We choose to expose only the best possible choice for the chosen plaintext strategies: always the best tradeoff for the number of sets S_i (*i.e.* the value M in Equation 4, Annex A). Apart from the noiseless case, the results are very close for both strategies. This actually implies that the `DetectCollision` success rate is a linear function of the number of plaintexts (from Equation 4, Annex A) as long as it does not come too close to its extremal values (0 and 1). As for the noiseless case, it is easy to see on Figure 2(a) that the linear property is not verified for the known plaintexts strategy, we actually observe on this figure an artefact that occurs when the number of plaintexts is too small for the correlation to be effective. These are the only cases where a chosen plaintexts approach is interesting, however they do not have much application in the real world.

Interestingly enough, considering the collision-correlation variants, in one hand the known plaintext strategy is as efficient as the chosen plaintext strategy when the optimal chain of plaintexts is used (see Annex A and Annex B). In another hand, in the chosen plaintext scenario with random plaintexts (scenario 1 in Annex A), the attack on the full key is always less efficient due to the appearance of *colliding collisions* (see Equation 3, Annex A).

Key-Recovery [4]

On Figure 3 are displayed the simulation results of the key-recovery algorithm for both the 2-byte 2nd-order CPA attack and the collision-correlation attack.

The results show that the equation system of the 2-byte 2nd-order CPA is easier to correct from erroneous equations: 30% of correct equations are enough

[2] 1-Byte 2nd-order CPA consists in combining the leakage of the handling of the mask with the leakage of the handling of a masked Sbox output

[3] 2-Byte 2nd-order CPA consists in combining the leakage of the handling of two masked Sbox outputs

[4] We only focus on the hard-decoding algorithms here. The Bayesian extension and LDPC soft-decoding will be used for the real experiments (and only in the case of collision-correlation attack since we still have no such treatment in the case of 2nd-order CPA attack)

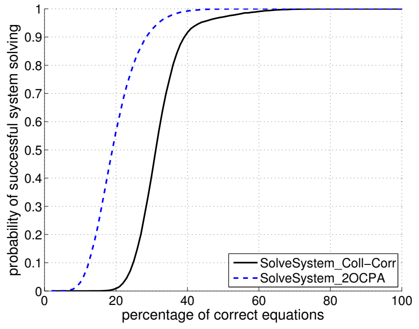


Fig. 3. success rate of System Solving as a function of the fraction of correct equations in the system computed over 100000 random simulations

to solve the system with very high probability whereas 40% of correct equations are needed in the collision-correlation attack. This result is not very surprising since the equation system of the 2-byte 2nd-order CPA conceals more information about the secret key: each equation gives 16-bit of information instead of 8-bit in the case of collision-correlation.

Success Rate of the Full AES Key Putting together the results on both attack phases gives straightforwardly the success rate to recover the full AES secret key. The 2nd-order CPA is clearly ahead of the collision-correlation attack. Notice that here we do not consider the Bayesian extension approach and just the termination algorithm composed with either the `DetectCollision` or the `MaxCorrelation` algorithms.

Comparison between 1*-Byte* 2^{nd}*-order CPA and* 2*-Byte* 2^{nd}*-order CPA.* In the latter case, using the termination algorithm that makes use of the natural redundancy in the equation system, achieving 90% success rate corresponds to a 30% success rate in the `MaxCorrelation` algorithm. However, when considering the 1-Byte 2nd-order CPA, no redundancy is available to the attacker, hence in order to have a 90% success rate in retrieving the full AES key, he should retrieve each of the 12 bytes (assuming the rest of the bytes are brute forced) independently with probability $e^{\frac{ln(90/100)}{12}} \simeq 0.99$ (assuming the leakage is homogeneous among each leaking intermediate variable). Surprisingly enough, under this hypothesis, the 2-Byte 2nd-order CPA becomes more efficient to retrieve the full AES key with high probability. For instance, when the noise standard deviation is set to 4, only 9000 plaintexts are needed for the 2-Byte 2nd-order CPA to recover the key with 90% success rate when 20000 plaintexts would be necessary for the 1-Byte 2nd-order CPA to achieve such a success rate.

4.2 Attack Results in a Real Setup

In order to validate our study in a real setup, we applied the attacks on a 1st-order masked implementation of AES using the mask reuse scheme described in [21]. To this end, we used an 8-bit microcontroller (MCU) based on a 8051 architecture. Thus the mask reuse AES implementation handles one byte at a time, and is a perfect target for the attacks considered here.

We have performed the measurements using an Electro-Magnetic (EM) setup composed of a commercial tiny EM sensor connected to a low-noise amplifier. To sample the EM side-channel measurements, we used a digital oscilloscope, with a sampling rate of 10 GSamples per second, whereas the MCU was running at about 30 MHz. Let us note that the MCU clock was not stable, we had to resynchronize the measurements. This process is out of the scope of this work, but we emphasize that such resynchronization step is always needed in a real setup. This is one of the reasons why heterogeneous leakages appeared in our measurements.

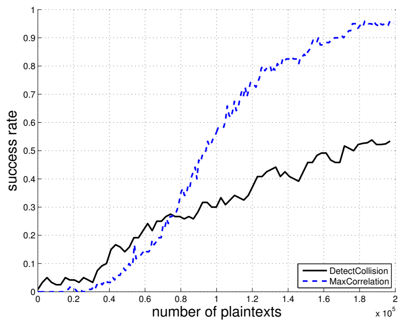
To validate this acquisition setup, we first performed a classical 1^{st}-order CPA attack on the AES encryption measurements by feeding the masks with 0. The attack succeeded with about 1000 traces (to retrieve the 16 bytes of the AES master key, we emphasize that some of the key bytes were retrieved in about 200 traces), thus validating both the quality of our acquisition setup and the efficiency of our resynchronization algorithm.

Secondly, we acquired 200000 measurements in a known plaintext setting, this time by feeding the masks with random values. We checked that the 1st-order CPA failed before testing the attacks considered in this work.

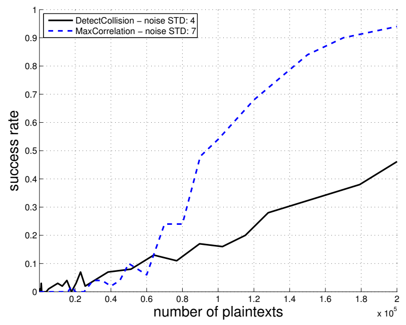
We also emphasize that the knowledge of good leakage locations in the consumption traces is crucial in 2nd-order attacks. In the case of 2nd-order CPA attack, this is mainly due to its computational cost that becomes impractical when too many samples in each trace have to be processed. In the case of the collision-correlation attack, this is even more true since if non data-dependent samples are involved, they will most likely lead to false-positive collisions detections. The research of the data-dependent samples in a set of side-channel traces has been studied in the side-channel literature (see [5] for instance in the case of collision-based SCA), in our case we computed the variance on the set of traces to select the data-dependent samples.

Remark 2. We also tried to mount a 1-Byte 2nd-order CPA. However, for unknown reasons, we were not able to catch the leakage of the mask manipulation during the Sbox re-computation phase.

We selected 16 sets of 10 samples each, each set corresponding to the handling of one Sbox output of the first round. We then applied, for each possible pair of subkey bytes, the `MaxCorrelation` and the `DetectCollision` algorithms, using the corresponding sets of samples. Figure 4(a) shows the success rate (over the 120 key byte pairs) of these two algorithms. The `MaxCorrelation` algorithm outperforms the `DetectCollision` algorithm, but the difference is less obvious than in the simulation results (confirming that the simulation leakage model is not very realistic, and gives a great advantage to 2^{nd}-order CPA). Figure 4(b) proposes the two success rate simulations that match best the real experiment, interestingly enough the real experiment corresponds to a simulated noise standard deviation of 7 for the `MaxCorrelation` algorithm whereas it matches a simulated noise standard deviation of 4 for the `DetectCollision`, this difference corresponds to the *stochastic* noise (i.e. that comes from the use of a leakage model).



(a) Evolution of the success rate *w.r.t.* the number of plaintexts for the `MaxCorrelation` and the `DetectCollision` algorithms applied on the **real measurements**



(b) Evolution of the success rate *w.r.t.* the number of plaintexts for the `MaxCorrelation` and the `DetectCollision` algorithms in **simulations** with different noise components

In a second phase we applied, for each of the two attack methods, the two `SolveSystem` algorithms described in Algorithm 5 of Annex C.2 and Algorithm 3 of Section 3.2 respectively. In the case of the 2-byte 2^{nd}-order CPA, the full key is retrieved with 73000 traces, whereas the collision-correlation attack requires 123000 traces to get 12 bytes of the full key (the 4 last bytes are brute-forced). These results match with the simulated success rate of the `SolveSystem` algorithms. One may note that, without applying the `SolveSystem` algorithms, the classical 2-byte 2^{nd}-order CPA would require at least 160000 traces to retrieve the full key, whereas the collision-correlation attack would not succeed to recover the key even by processing the 200000 traces.

Furthermore, we added the Bayesian extension and LDPC decoding, as depicted in Section 3.2 Algorithm 2 for the collision-correlation attack. Using such improvements, we were able to retrieve the correct key in about 50000 traces for the collision-correlation attack (with a final brute force on 40 bits). Unfortunatly, we could not perform a similar improvement for the 2-byte 2^{nd}-order CPA attack as the Bayesian extension is still a work in progress (see a discussion on this issue in Annex C.2); however, there is no reason that such mechanism cannot be applied in the process of 2-Byte 2^{nd}-order CPA attack and improve the attack as much as it does for the collision-correlation attack.

5 Conclusion

The literature dealing with combinations of collision cryptanalysis and side-channel analysis often took the side of studying the algorithmic problem into an idealized setup (very low noise, profiled attacks). This statement explains in part the low popularity of such techniques in practice. In this article we try to replace the collision-based SCA over masked implementation into a more realistic setup, following the recent work of Gérard and Standaert [15]. To this end we focus on the collision-correlation attack [11] and point out why the construction is intrinsically unappropriated to unknown and high noise context. Moreover, we show how to adapt this attack and provide a comparison with 2^{nd}-order SCA attacks. The results show that 2^{nd}-order attack is more efficient when the leakage function is well approximated (we use the Hamming Weight model in our experiments). However, since the collision-based SCA does not need any knowledge about the leakage function, it will become more efficient relatively to 2^{nd}-order SCA as its prediction function deviates from reality. It must be noted that the Bayesian extension approach (introduced in [15]) greatly improves the collision-correlation attack efficiency, and makes it the best attack in our real setup. We let for further studies the design of an Bayesian extension to the 2-Byte 2^{nd}-order CPA attack, with such improvement we believe the attack would outperforms collision-correlation.

This work opens to other several directions, mainly in the sense of optimisation of collision-based SCA attacks. Firstly, the collision-correlation attack does not make good use of very important advantage of collision-based SCA: when a collision appears, its side-channel signature spreads through potentially many sample points overlapping several clock cycles. This *horizontal* treatment, on which were based the majority of previous collision-based SCA, should be incorporated in some way to optimize the detection of collision. Secondly, a real breakthrough would be to lower its data-complexity cost by making a better use of the traces in the collision detection algorithm: in our setup we divide them in independent sets where correlation is locally computed (*i.e.* the lower bound on the attack complexity beats the actual attack by a factor of 256, maybe this large overhead leaves room for optimizations).

Another direction opened by our study is the research of very efficient masking schemes that are more robust against 2^{nd}-order attacks. Although efficient, the mask reuse scheme clearly helps the attacker compared to a 1^{st}-order masking scheme using more masks. Furthermore, the use of 2^{nd}-order masking schemes should start to be considered in real-world secure devices.

Acknowledgements

We would like to thanks Thomas Fhur and Eliane Jaulmes for many interesting discussions and more particularly for the proposition of an optimal plaintext sequence.

References

1. O. Benoit and M. Tunstall. Efficient Use of Random Delays. Cryptology ePrint Archive, Report 2006/272, 2006.
2. A. Biryukov and D. Khovratovich. Two New Techniques of Side-Channel Cryptanalysis. In Paillier and Verbauwhede [25], pages 195–208.
3. A. Bogdanov. Improved Side-Channel Collision Attacks on AES. In *Proceedings of the 14th international conference on Selected areas in cryptography*, SAC'07, pages 84–95, Berlin, Heidelberg, 2007. Springer-Verlag.
4. A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 30–44, Berlin, Heidelberg, 2008. Springer-Verlag.
5. A. Bogdanov and I. Kizhvatov. Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. *IEEE Transactions on Computers*, 99(PrePrints), 2011.
6. A. Bogdanov, I. Kizhvatov, and A. Pyshkin. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In D. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer Berlin / Heidelberg, 2008.
7. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In Joye and Quisquater [17], pages 16–29.
8. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Wiener [32], pages 398–412.
9. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski Jr., Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2002.
10. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Koç and Paar [18], pages 252–263.
11. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Improved Collision-Correlation Power Analysis on First Order Protected AES. In *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*, CHES'11, pages 49–62, Berlin, Heidelberg, 2011. Springer-Verlag.
12. FIPS PUB 197. *Advanced Encryption Standard.* National Institute of Standards and Technology, Nov. 2001.
13. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
14. L. Genelle, E. Prouff, and M. Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2011.
15. B. Gérard and F.-X. Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting. In E. Prouff and P. Schaumont, editors, *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2012.
16. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

17. M. Joye and J.-J. Quisquater, editors. *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*. Springer, 2004.
18. Ç. Koç and C. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*. Springer, 2000.
19. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
20. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Wiener [32], pages 388–397.
21. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
22. T. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms*. PhD thesis, University of Illinois, 2000.
23. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant Software. In Koç and Paar [18], pages 238–251.
24. A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*, CHES'10, pages 125–139, Berlin, Heidelberg, 2010. Springer-Verlag.
25. P. Paillier and I. Verbauwhede, editors. *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*. Springer, 2007.
26. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In Paillier and Verbauwhede [25], pages 81–94.
27. E. Prouff, M. Rivain, and R. Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.*, 58(6):799–811, 2009.
28. K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES (Combining Side Channel and Differential-Attack). In Joye and Quisquater [17], pages 163–175.
29. K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and its Application to DES. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003.
30. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.
31. J. Waddle and D. Wagner. Toward Efficient Second-order Power Analysis. In Joye and Quisquater [17], pages 1–15.
32. M. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

A A Theoretical Study of the `DetectCollision` Algorithm in Different Attack Scenarios

Notation. In the following we denote by $\Pr_{a,b}^{DetectCol}(N)$ the probability of success of a collision detection for the pair (a, b) of sensitive bytes, we will often refer to $\Pr(N)$ that corresponds to the average value of $\Pr_{a,b}^{DetectCol}(N)$ over all possible pair (a, b).

Known Plaintexts Scenario This approach is described in Algorithm 1 of Section 3.

Data Complexity: for a data complexity of $256 \cdot N$ plaintexts, the average number of correct equations NEqu among the $\binom{16}{2}$ equations is equal to:

$$\mathrm{NEqu}(256 \cdot N) = \Pr(N) \cdot \binom{16}{2} \ , \tag{2}$$

with $\Pr(N)$ the average probability of success of the collision detection.

Chosen Plaintexts Scenario It consists in reducing the number of sets S_j. This will also reduce the probability of correct detection since, for a choice of (a, b), if $S_{k_a \oplus k_b}(a, b)$ is empty, then the `DetectCollision` will obviously set the equation $k_a \oplus k_b = \alpha$ to an erroneous value α.

In such scenario, the chosen plaintext setting is necessary and several directions may be investigated:

scenario 1 The attacker randomly chooses $M < 256$ plaintexts p^i and encrypts each of them N times. Thus, for each pair of bytes (a, b), for all $i \leq N$, the leakage of the N encryptions of p^i goes into the set $S_{p^i_a \oplus p^i_b}(a, b)$. Obviously, for every pair of bytes (a, b), there is at most M non empty sets $S_j(a, b)$. This strategy is the most common choice in the collision-based SCA literature, as a matter of fact it fits very well the attack when the `DetectCollision` function is capable of asserting the presence or the absence of collision (as in [11]). We will see that, in our case, this is a bad choice, even compared to the known plaintext setting.

Data Complexity: the data complexity is equal to $M \cdot N$ encryptions. In order to compute the corresponding success rate, we first need to evaluate the number of collisions that are detectable over all the pairs of byte indexes $\binom{16}{2} = 120$. Hence we want to find the average number of byte collisions NCol when considering M successive plaintexts:

$$\begin{aligned} \mathrm{NCol}(M) &= \mathbb{E}\left(\textstyle\sum_M \sum_{0 \leq a < b < 16} \mathbf{1}_{z_a = z_b}\right) \\ &= M \cdot \tbinom{16}{2} \cdot \tfrac{1}{256} \end{aligned}$$

Since the attacker randomly chooses the plaintexts, it is possible to retrieve several times the collision on a given pair of indices (say (a, b)), *i.e.* there exists two plaintexts p^i and p^j such that $p^i_a \oplus p^i_b = p^j_a \oplus p^j_b = k_a \oplus k_b$. Such redundancy will not help to solve the system (apart from increasing the success probability $\Pr^{DetectCol}_{a,b}(N)$), we need to suppress them from the success rate evaluation. The average number of such *colliding collisions* may be computed as follows:

$$\mathrm{NCol}'(\mathrm{NCol}) = \sum_{i=1}^{\mathrm{NCol}} \left(1 - \left(1 - \frac{1}{\binom{16}{2}}\right)^{\mathrm{NCol}-i}\right)$$

Finally, the average number of collisions that will effectively lead to an equation (keeping in mind the success probability of collision detection $\Pr(N)$) is then given by:

$$\mathrm{NEqu}(M{\cdot}N) = \Pr(N)\cdot\left[\mathrm{NCol}(M) - \sum_{i=1}^{\mathrm{NCol}(M)}\left(1-\left(1-\frac{1}{\binom{16}{2}}\right)^{\mathrm{NCol}(M)-i}\right)\right] \tag{3}$$

Remark 3. The complexity evaluation made here is actually very close to what can be found in [3] (if we omit the success probability of collision detection that was considered to be maximal), however, its theoretical study and empirical results cannot be applied in our context since the authors of [3] targeted an unmasked implementation that made possible to consider collision between two bytes from two different encryptions.

scenario 2 The attacker *carefully* chooses $M < 256$ plaintexts p^i and encrypts each of them N times. The difference with the previous scenario is that the attacker follows an optimal choice of the plaintexts $\{p_i\}_{i<M}$ such that for every pair of bytes (a, b), for all (i, j) *s.t.* $i \neq j$, we have $p^i_a \oplus p^i_b \neq p^j_a \oplus p^j_b$. We propose in Annex B such an optimal strategy.

Data Complexity: here also the data complexity is $M \cdot N$ plaintexts. For the success rate, the context is pretty similar to *scenario 1*, the main difference being that there is no difference anymore between NEqu and NCol since by construction of the plaintexts there cannot exist a *colliding collision*. In this scenario the average number of byte collisions when considering $M \leq 256$ successive plaintexts (with an average probability of success of $\Pr(N)$ for collision detection):

$$\begin{aligned}\mathrm{NEqu}(M\cdot N) &= \Pr(N)\cdot\mathbb{E}\left(\textstyle\sum_M\sum_{0\leq a<b<16}\mathbf{1}_{z_a=z_b}\right)\\ &= \Pr(N)\cdot M\left(\binom{16}{2}\cdot\frac{1}{256}\right)\end{aligned} \tag{4}$$

Hence, when $M = 256$, this strategy is equivalent to the known plaintext setting. Furthermore, we will observe by simulations (Section 4) that $\Pr(N)$ is a linear function of N (over some fixed range $\Pr(N) \in [0+\epsilon, 1-\epsilon]$), making this chosen plaintexts scenario completely equivalent to the known plaintexts approach (it can be easily checked that Equ. 2 and Equ. 4 are then similar).

B Finding an Optimal Sequence of Plaintexts for the Collision-Correlation Attack

Proposition 1. *Let* $v_0, v_1, \cdots, v_{n-1}$ *be* $n \leq 256$ *distinct elements in* $GF(256)$*. Let consider the* $(255 \times n)$ *matrix* A *defined as follows: the* k^{th} *element of row* i*, denoted* $a_{i,k}$*, is such that*

$$a_{i,k} = v_k \times \alpha^i \ ,$$

where α is a primitive element of $GF(256)$.
The addition in $(GF(256))^{255}$ of any pair of columns of A is a vector that contains all the non-zero values of $GF(256)$.

Proof. For any value $i < n$ and any value $j < n$ such that $i \neq j$, let us consider the i^{th} and j^{th} column of A, denoted respectfully A_i and A_j, and the vector W resulting from the addition of A_i and A_j. The elements of W, denoted $(w_k)_{k<256}$, are such that

$$w_k = (v_i \oplus v_j) \times \alpha^k$$

Hence, assuming that there exists two elements of W that are equal means that there exists $k < l < 256$ such that

$$w_k \oplus w_l = (v_i \oplus v_j) \times (\alpha^k \oplus \alpha^l) = 0 \ .$$

Since $v_i \neq v_j$ by definition, we have $\alpha^k \oplus \alpha^l = 0$, which is absurd from the fact that α is a primitive element of $GF(256)$.
For the same reasons, $w_k = 0$ is also absurd. $\diamond$

From Proposition 1, we can straightforwardly build a sequence of 255 plaintexts such that the exclusive-or between two fixed bytes is distinct among all the plaintexts and also different from 0. Furthermore, a plaintext composed with the repetition of the same byte (and then for which any difference between two bytes is null) completes the sequence to form an optimal chain of 256 plaintexts.

C 2nd-order CPA

In this Annex we propose a 2nd-order *Correlation Power Analysis* (CPA) algorithm, that we call 2-byte 2nd-order CPA attack, dedicated to the target implementation. As a matter of fact, the mask reuse scheme allows to combine pairs of masked sensitive variables, which leads to an interesting termination algorithm that focuses on retrieving the whole 16-byte secret key.

Let us assume that the attacker encrypts a set of N plaintexts $\{p^i\}$, the attack will focus on the handling of the Sbox outputs of the first AES round. Hence, for each encryption $i \leq N$, there are 16 sensitive variables of interest: $\forall a \in \{0 \cdots 15\}, z_a^i = \texttt{SubBytes}(p_a^i \oplus k_a)$. Moreover, for each sensitive variable we will assume that the consumption leakage L_a^i retrieved by the attacker is of the form: $L_a^i = HW(z_a^i \oplus m^i) + \mathcal{N}$, where m^i is the random byte used during the first round of the i^{th} encryption to mask every `SubBytes` output and $\mathcal{N}$ is a noise component, independent of the sensitive variable and of the mask (usually modeled as a Gaussian noise with mean μ and standard deviation σ).

The idea of 2nd-order DSCA (due to Messerges [23]) is to combine the two leakages corresponding to the handling of the two shares of a masked intermediate variable and then apply a 1st-order DSCA attack.

Different leakage combining functions have been proposed in the literature (see for instance [8, 23, 31]). Eventually, in [27], Prouff *et al.* proved that, if one assumes that the device leaks in the Hamming Weight model (HW), and if one considers the Pearson correlation as distinguisher (*i.e.* CPA [7]), the best known combining function is the normalized product of the two leakages. We will then consider the following 2nd-order attack described below (denoted by 1-Byte 2nd-order CPA in the sequel).

Let the two shares of the sensitive variable z be $z \oplus m$ and m, with m the mask, and their respective leakages being L_1 and L_2. Then, the normalized product combining function consists in computing:

$$C_{prod}(L_1, L_2) = (L_1 - \mathbb{E}(L_1)) * (L_2 - \mathbb{E}(L_2)) \ ,$$

with $\mathbb{E}$ being the Expectation function. In addition, authors of [27] show that the optimal prediction function (to be correlated with $C_{prod}(L_1, L_2)$ over, say, N messages), denoted f_{opt}, is:

$$f_{opt}(\hat{z}) = \frac{1}{256} \cdot \sum_{m=0}^{255} (HW(\hat{z} \oplus m) - \mathbb{E}(HW(\hat{z} \oplus m))) \cdot (HW(m) - \mathbb{E}(HW(m))) \ ,$$

where $\hat{z} = \texttt{SubBytes}(p \oplus \hat{k})$ with p a plaintext byte and $\hat{k}$ a guess on the secret key byte. Thus, the adversary computes, for each key byte candidate, the correlation between $f_{opt}(\hat{z})$ and $C_{prod}(L_1, L_2)$. As the number N of plaintexts grows, the correct key candidate leads to the highest correlation value.

Application to AES Mask Reuse Scheme In the implementation described in Section 1, a unique masked Sbox is recomputed before the beginning of each encryption. If it can be easy to identify the position of the AES rounds in the side-channel trace, it is not always the case for the leakage of the Sbox recomputation. Thus, applying a 2nd-order CPA as explained before will imply to select large strips of samples to be sure to include the handling of the mask during the Sbox recomputation, and the handling of the masked Sbox output.

Another method would be to combine the handling of two masked Sbox outputs during the first round computation, and guess two key bytes at a time (this attack will then be denoted by 2-Bytes 2nd-order CPA in all the following). In this case one combines the handling of the leakage of two masked sensitive variables, $z_a \oplus m$ and $z_b \oplus m$. Since the same mask is used for both sensitive variables, the normalized product combining function C_{prod} will reveal a dependence on z_a and z_b. Hence, following [27], the optimal prediction function becomes:

$$\begin{aligned} f_{opt}(\hat{z_a}, \hat{z_b}) = \tfrac{1}{256} \cdot \textstyle\sum_{m=0}^{255} \ & (HW(\hat{z_a} \oplus m) - \mathbb{E}(HW(\hat{z_a} \oplus m))) \\ & \cdot (HW(\hat{z_b} \oplus m) - \mathbb{E}(HW(\hat{z_b} \oplus m))) \ . \end{aligned} \qquad (5)$$

Such a method implies to make an hypothesis on two key bytes at a time (k_a and k_b), but will eventually give information about both key bytes. Furthermore, it is possible to attack all pairs of key bytes independently, i.e. the 120 pairs of key bytes (k_a, k_b), with $a \neq b$, and $a, b \in \{0 \cdots 15\}$ (similarly to the collision-correlation attack presented in Section 3). This last remark leads to a key-recovery algorithm that is specific to our target implementation and attack choice. In the following are described the two main components of the 2-byte 2nd-order CPA attack, namely the `MaxCorrelation` function that provides the best hypothesis of pair of key bytes from a set of side-channel leakages and the key-recovery algorithm which, from 120 best key byte pair candidates, provide the full 16-byte secret key. The complete 2-byte 2nd-order CPA algorithm is given in Algorithm 6. It can be noted that the 1-Byte 2nd-order CPA does not lead to such key-recovery mechanism, since each key byte is retrieved independently from the others. Eventually, this will make this attack less efficient than the 2-Bytes 2nd-order CPA (see Section 4.1).

C.1 MaxCorrelation

We propose in Algorithm 4 to expose the `MaxCorrelation` function for purpose of completeness, this is a basic component of 2nd-order CPA, we do not provide improvement to the state of the art (the 1-Byte version of the attack is well known and then not recalled here).

Algorithm 4's inputs are 4 vectors of size N, namely $(p_a^i)_{i\leq N}$ and $(p_b^i)_{i\leq N}$ that correspond respectively to the a^{th} and the b^{th} bytes of N plaintexts and $(L_a^i)_{i\leq N}$ and $(L_b^i)_{i\leq N}$ that are the information leakages corresponding to the handling of $z_a \oplus m$ and $z_b \oplus m$ retrieved during the encryption of the N plaintexts. The function outputs the best guess for the pair of key bytes (k_a, k_b).

Algorithm 4 `MaxCorrelation` for 2-Bytes 2nd-order CPA

```
INPUT: $(p_a^i)_{i\leq N}, (p_b^i)_{i\leq N}, (L_a^i)_{i\leq N}, (L_b^i)_{i\leq N}$
OUTPUT: 2 key bytes: $(k_a, k_b)$

1. for $\hat{k_a} = 0$ to 255
2.     for $\hat{k_b} = 0$ to 255
3.         do $\rho_{\hat{k_a},\hat{k_b}} \leftarrow \rho((f_{opt}(\hat{z_a^i}, \hat{z_b^i}))_{i\leq N}, (C_{prod}(L_a^i, L_b^i))_{i\leq N})$
               [Evaluate the Pearson Correlation Coefficient for the hypothesis $(\hat{k_a}, \hat{k_b})$]
4. return $(k_a, k_b) = \text{argmax}(\rho_{\hat{k_a},\hat{k_b}})$
```

C.2 Key-Recovery

Bayesian Extension and LDPC Soft-Decoding In a very similar context, the use of Bayesian extension and LDPC soft-decoding have been shown to be very efficient to reduce the data complexity of collision-correlation attack (see Section 3.2 and results in Section 4.2). However, to our knowledge, no such treatment has been conducted on 2nd-order CPA and we observed that a direct application of the above mentioned mechanism in our context does not provide relevant results. One reason may be that the Bayesian extension is done under the hypothesis that a Fisher transformation of the correlation coefficient follows a Gaussian law (see [15]). This is a good approximation when the two correlated random variables follow a Gaussian law (as in 1st-order CPA or collision-correlation attack) but not anymore in 2nd-order CPA (where the normalized product appears).

Hence we let for future work the Bayesian approach in the case of 2nd-order CPA.

Termination Algorithm We call *key-recovery* the part of the attack algorithm which, from the output of the `MaxCorrelation` function applied independently to the 120 pairs $(z_a, z_b)_{0 \leq a < b < 16}$ of sensitive bytes, delivers the most probable full secret key candidate. Thus the attacker receives a list of equations involving the key bytes: $\forall 0 \leq a < b < 16, (k_a, k_b) = (\alpha_a, \alpha_b)$. Solving the equation system is trivial when there is no erroneous equation, all the work is then dedicated to identifying and correcting (if possible) the erroneous equations[5].

Detection/Correction of Erroneous Equations It can be easily checked that the equation system possesses 120 equations and 16 unknowns. Hence, 120-16 of these equations may be considered as redundant and then used to detect, and eventually correct, some potential erroneous equations. Although generic decoding algorithms may exist, we have not investigated this direction as the equation system possesses a very specific structure that led us to an efficient algorithm, we let open the question of a better, if not optimal, decoding algorithm for the equation system.

Algorithm 5 describes the so-called decoding algorithm, the main idea of the algorithm is to consider that, in a given equation $(k_a, k_b) = (\alpha_a, \alpha_b)$, if one of the two values α_i is erroneous it would be merely by chance that the other one is correct. The other way around, if one half of the equation can be asserted as correct, there is a good chance than the other half is also correct.

Algorithm 5 `SolveSystem_2OCPA`

INPUT: A noisy equation system S_{in}

OUTPUT: A clean equation system S_{out}

```
1. Initialize an Empty Equation System $S_{out}$
2. Initialize 3 arrays to zero: $max\_val[16], max\_count[16]$ and $B[16][256]$
3. for $a = 0$ to 15
4.     do Set in $max\_val[a]$ the value that is proposed the most often for the byte $k_a$
5.     do Set in $max\_count[a]$ the number of times $max\_val[a]$ is proposed for the byte $k_a$
6. for $a = 0$ to 15
7.     for $b = 0$ to 15, $b \neq a$
8.         do Consider the equation $(k_a, k_b) = (\alpha_a, \alpha_b)$ from $S_{in}$
```

[5] this study would be useless if all the sensitive variables were leaking exactly the same way. We believe that this is not a realistic assumption specifically when dealing with secure devices (for instance because of random clock). Indeed, our real experiment invalidated the idealist assumption

```
9.          if $max\_val[a] = \alpha_a$ then do $B[b][\alpha_b] \leftarrow B[b][\alpha_b] + max\_count[a]$
10. for $a = 0$ to 15
11.     do Find the index $\alpha$ such that $B[a][\alpha]$ is maximum, set $k_a = \alpha$ into $S_{out}$
12. return $S_{out}$
```

We give the simulation results of the algorithm in Section 4.1 and compare them to the system solving success rate in the case of the collision-correlation attack. Using the Algorithm 5, the equation system will be correctly solved with a success probability above 90% when only 30% of the equations are correct[6]. In Section 4.2, we also show that the simulation results match our experimental results with respect to the key-recovery, *i.e.* the repartition of erroneous equations in the system does not have a special structure that helps/handicaps the detection algorithm.

C.3 Full 2nd-order CPA Algorithm

In this section is described the full attack algorithm based on 2nd-order CPA.

Algorithm 6 2-Bytes 2nd-order CPA

INPUT: A 1st-order mask reuse AES implementation, a number of plaintexts N
OUTPUT: The 16-byte secret key

```
1. Initialize an Empty Equation System $S_{in}$
2. do Encrypt $N$ new plaintexts $\{p^i\}_{1\leq i\leq N}$, retrieve $N\times 16$ leakages $\{L_a^i\}_{1\leq i\leq N, 0\leq a\leq 15}$
        [$L_a^i$ being the leakage of the $a^{\text{th}}$ Sbox output during the encryption of the $i^{\text{th}}$
   plaintext]
3. for $a = 0$ to 15
4.     for $b = a + 1$ to 15
5.         do $(\alpha_a, \alpha_b) \leftarrow$ MaxCorrelation($\{p_a^i, p_b^i, L_a^i, L_b^i\}_{i\leq N}$)
                           [Find the best choice $(\alpha_a, \alpha_b)$ for the pair of key bytes $(k_a, k_b)$]
6.         do Add $(\mathbf{k_a}, \mathbf{k_b}) = (\alpha_\mathbf{a}, \alpha_\mathbf{b})$ to $S_{in}$
7. do $S_{out} \leftarrow$SolveSystem_2OCPA($S_{in}$)          [The secret key $k$ is retrieved from the
   equation system.]
8. return $(k_a)_{a<16}$                                           [contained in $S_{out}$]
```

A study of the data complexity (N) of the attack is proposed in Section 4 and compared with the collision-correlation attack.

[6] when the erroneous equations are uniformly distributed