Improved Blind Side-Channel Analysis by Exploitation of Joint Distributions of Leakages

Christophe Clavier and Léo Reynaud

Université de Limoges, XLIM-CNRS
Limoges, France

christophe.clavier@unilim.fr
leo.reynaud@xlim.fr

Abstract. Classical side-channel analysis include statistical attacks which require the knowledge of either the plaintext or the ciphertext to predict some internal value to be correlated to the observed leakages.
In this paper we revisit a blind (i.e. leakage-only) attack from Linge et al. that exploits joint distributions of leakages. We show – both by simulations and concrete experiments on a real device – that the maximum likelihood (ML) approach is more efficient than Linge's distance-based comparison of distributions, and demonstrate that this method can be easily adapted to deal with implementations protected by first-order Boolean masking. We give example applications of different variants of this approach, and propose countermeasures that could prevent them.
Interestingly, we also observe that, when the inputs are known, the ML criterion is more efficient than correlation power analysis.

Keywords: unknown plaintext, joint distributions, maximum likelihood

1 Introduction

Cryptographic implementations of embedded products like smartcards are known to be vulnerable to statistical side-channel analysis such as Differential Power Analysis [12], Correlation Power Analysis [1] or Mutual Information Analysis [7]. These side-channel analyses are divide-and-conquer attacks where the whole key is recovered by chunks of few bits (e.g. one byte) at a time. This is possible because the device produces a measurable leakage like power consumption or electromagnetic emanation which depends at any instant on the internal value manipulated by the processor. When this value only depends on a public information – like the plaintext or the ciphertext – and a small piece of the key, a so-called subkey, it is possible to validate or invalidate an hypothesis about the subkey by correlating the leakage with a prediction of the internal value.

While these statistical analyses all require the knowledge of the input or the output to be correlated with, there are some use cases or protocols where this information is either not available or not exploitable. This is the case for the derivation of the session key that is used to compute application cryptograms in the EMV payment scheme [4, p. 128] (see also left of Fig. 8). In this case the attacker does not know the output (session key) and the input only varies on its first two bytes, so that he can expect to recover only the two corresponding bytes of the master key.

To deal with situations where neither the plaintext nor the ciphertext are available, Linge et al. introduced the concept of joint distribution analysis [16]. In the case of the AES cipher, the idea is to exploit the fact that the joint distribution of the Hamming weight of a byte m and that of $y = S(m \oplus k)$ depends on k so that this key byte value can be retrieved (at any round) by comparing the distance between the observed experimental distribution of $(\mathrm{HW}(m), \mathrm{HW}(y))$ and all 2^8 theoretical ones. Linge et al. also proposed a so-called *slice* method to convert leakages to Hamming weights. While Le Bouder [14] presented an alternative approach – based on the maximum likelihood (ML) criterion – to Linge's distance-based comparison of distributions, she did not provide any comparison between both methods. In this paper we build upon [16,14] and provide the following contributions: (i) we propose a novel way to estimate Hamming weights based on variance analysis, (ii) we compare the ML and distance-based methods using both the slice and the variance analysis ways of obtaining Hamming weights, (iii) we present new variants that improve the attack by exploiting other and/or more points of interest, (iv) we adapt the blind joint distribution analysis to implementations featuring Boolean masking countermeasure. Our work is supported by experimental results based both on simulations and on real traces.

Another related work by Hanley et al. [11] presents a template-based attack by joint distribution analysis in the blind context. This work differs from our's as it requires a profiling phase on a similar device with known key where the unknown input assumption does not hold. Also, and contrarily to our work, the adaptation of their attack to masked implementations is only applicable on the first round. In the context of blind fault analysis, Korkikian et al. [13] and Li et al. [15] also exploit the joint distribution of $(\mathrm{HW}(m), \mathrm{HW}(y))$ with the distance-based and ML methods respectively.

This paper is organized as follows. In Section 2 we introduce the notations used in the paper and present the original joint distribution analysis from Linge et al. In Section 3 we describe the ML criterion and compare it to Linge's distance-based method. Section 4 considers how the different variants of our attacks can be adapted to implementations protected by first-order Boolean masking. We then depart from the unknown plaintext scenario in Section 5 to further assess the efficiency of the ML criterion and compare it with classical CPA. Concrete experiments on side-channel traces captured from a real device are presented in Section 6 and their results compared to simulation figures. We then provide several application examples of our attacks in Section 7 and discuss which kind of countermeasures could prevent them. Section 8 finally concludes this work.

2 Background and Original Linge's Attack

The attacks presented in this paper assume a software implementation of a block cipher, and without loss of generality we will consider the AES [18]. For these attacks one needs to measure the leakages corresponding to some specific internal byte states. We thus assume that the attacker is able to locate precisely the points of interest related to these variables. This means that an attacker facing an implementation hardened by random delays or other types of time randomization must have been able to preprocess the traces and remove the effect of these desynchronizations. When the traces are aligned, identifying the points of interest may not be an easy task. Since the attacker does not know the plaintext, the statistical T-test or other tests that partition the trace set based on a plaintext dependent value [8,9,17,3] can not be used. Although, it is still possible to identify instants where the device shows a high activity from the peaks on the trace of standard deviations such as depicted in Figure 2. Such traces do not provide any clue by themselves about which kind of internal data leaks, but this information may be guessed based on reasonable assumptions about the implementation.

2.1 Notations

We are mainly concerned with three kinds of internal AES states that we generically call m, x and y, and respectively correspond to:

$\boldsymbol{m}$ **:** the input byte of the XOR operation with the key byte k during the `AddRoundKey` function,

$\boldsymbol{x}$ **:** the result of the XOR operation with the key byte ($x = m \oplus k$), which is the input of the S-Box during the subsequent `SubBytes` function,

$\boldsymbol{y}$ **:** the output of the S-Box ($y = S(x) = S(m \oplus k)$) during the `SubBytes` function.

Note that, except if explicitly stated, we do not assume any particular byte number or any particular round number for m, x and y.

2.2 The Original Attack

The joint distribution analysis proposed by Linge et al. [16] considers the case of two state bytes m and y which are seen as random variables. Assuming uniformly distributed plaintexts, the probability distributions of both m and y – considered separately – are uniform and independent on the key. Though, this is totally different for the joint distribution of the couple (m, y). Indeed this joint distribution actually depends on k. The core idea of Linge's attack is that if the joint distribution of (m, y) depends on the key then it should be also true for the joint distribution of their Hamming weights $(\mathrm{HW}(m), \mathrm{HW}(y))$. We can thus consider 2^8 theoretical distributions of $(\mathrm{HW}(m), \mathrm{HW}(y))$, one per value of k, that we call models and which we denote by $\mathcal{M}_k$. As an illustrative example, Fig. 1 shows models $\mathcal{M}_{39}$ and $\mathcal{M}_{167}$ which clearly appear to be different.

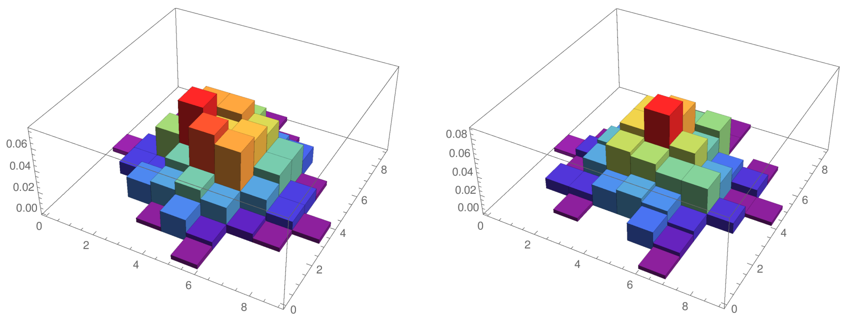


Fig. 1: Joint distributions of $(\mathrm{HW}(m), \mathrm{HW}(y))$ for $k = 39$ (left) and for $k = 167$ (right)

Since we assume the Hamming weight leakage model, an attacker able to infer the Hamming weights of m and y from two corresponding series of leakages ℓ_m and ℓ_y can generate an empirical distribution of $(\mathrm{HW}(m), \mathrm{HW}(y))$ issued from the device. We denote this distribution by $\mathcal{D}$. As this empirical distribution should converge to the model corresponding to the key used in the device, one can compare $\mathcal{D}$ with each model $\mathcal{M}_k$ and select the one that achieves the best match. To sum up, Linge's attack comprises three steps:

Computing the models. One computes the theoretical distribution $\mathcal{M}_k$ for each key candidate. This is simply a matter of considering all possible inputs m, derive the value $y = S(m \oplus k)$, counting how many times each couple $(\mathrm{HW}(m), \mathrm{HW}(y))$ appears, and normalizing in order to obtain the probability distribution. These models are independent from the device and can thus be computed beforehand.

Obtaining the empirical distribution $\mathcal{D}$. Given a large set of traces corresponding to encryptions with random inputs, one measures the leakages ℓ_m and ℓ_y at the two previously identified points of interest. These couples of leakages must be converted to couples of Hamming weights in $\{0, \ldots, 8\} \times \{0, \ldots, 8\}$ in order to comply with the domain of the models. Finally, counting the number of occurrences of each observed Hamming weight couple, and normalizing by the total number of observations, allows to generate the empirical distribution $\mathcal{D}$.

Comparing $\mathcal{D}$ with the models. Linge et al. proposed to compare the empirical distribution to the theoretical ones based on some distance. The closest model $\mathcal{M}_k$ to $\mathcal{D}$ provides the best candidate for the secret. They studied a large panel of 65 distances and selected four of them for giving better results in the presence of errors in estimating the Hamming weights.

A tricky task in this attack is the conversion from leakages to Hamming weights. Linge et al. proposed a simple method that assigns Hamming weights by "slices" of the sorted list of leakages in accordance to their relative probabilities. Given a set of leakages measured at a given point of interest, if we consider them sorted in ascending order, it is reasonable to think that the smallest ones would correspond to a Hamming weight $h = 0$ and the largest ones to $h = 8$. How many leakages should correspond to each Hamming weight slice may be estimated by the theoretical proportion of each of them: given the leakages of n random values, one assigns $h = 0$ to the $\frac{n}{256}\binom{8}{0}$ smallest ones, $h = 1$ to the $\frac{n}{256}\binom{8}{1}$ immediately larger ones, and so on, up to $h = 8$ to the $\frac{n}{256}\binom{8}{8}$ largest leakages.

3 Improved Joint Distribution Analysis

The attack presented in Section 2.2 does not require the knowledge of neither the plaintext nor the ciphertext. This remarkable property results from the fact that the analysis is *local*: the information used to "correlate" with the S-Box output y is self-contained in the trace since it comes from the leakage of m instead of from its value. The important benefit from this is that the attack applies at any arbitrary round, and not solely on the first or last one. On the other side, instead of using the exact value of the input – as in classical attacks – this information is replaced by a noisy estimation of its Hamming weight. This makes this attack less efficient than classical ones (in term of number of traces) and motivates the need to exploit the available information as efficiently as possible.

In this section we recall an improved method to exploit the joint distribution of leakages at points m-y which is based on the maximum likelihood criterion [14]. The idea is to compute for each key hypothesis the probability of this key given the observed leakages. The attacker then selects the most probable one. In this approach the noisy leakage must be converted to a noisy Hamming weight which does not require to be an integer value since the noise is modeled as being distributed according to a centered Gaussian law with variance σ^2. Section 3.2 discusses several ways to convert the original leakages to real-valued Hamming weights.

3.1 Maximum Likelihood Criterion

We consider a noisy measurement (h_m, h_y) of a couple of Hamming weights (h_m^*, h_y^*) corresponding to the values manipulated at points of interest related to m and y. That means $h_m^* = \mathrm{HW}(m)$ and $h_y^* = \mathrm{HW}(y) = \mathrm{HW}(S(m \oplus k))$. We have $h_m = h_m^* + \omega_m$ and $h_y = h_y^* + \omega_y$ where ω_m and ω_y are two independent and centered Gaussian noises with standard deviations σ_m and σ_y respectively[1]. The probability of the key given a single observation of Hamming weights can be derived from Bayes formula as:

$$\Pr(k|(h_m, h_y)) = \frac{\Pr((h_m, h_y)|k) \cdot \Pr(k)}{\Pr((h_m, h_y))}$$

Note that in this equation the denominator $\Pr((h_m, h_y))$ is a normalization term that does not depend on the key. We can simply ignore it since we are just interested in comparing the probabilities to each other instead of actually computing their values[2]. We so have

$$\Pr(k|(h_m, h_y)) \propto \Pr((h_m, h_y)|k) \cdot \Pr(k)$$

where the term $\Pr(k)$ corresponds to the uniform distribution in the case of a first observation of (h_m, h_y) and more generally to the posterior distribution computed based upon the already exploited Hamming weights couples. The probability of the key given a set of observations $((h_m, h_y)_i)_{i=1\ldots n}$ can then be derived in the following iterative way:

$$\Pr(k|((h_m, h_y)_i)_{i=1\ldots n} \leftarrow \Pr((h_m, h_y)_n|k) \cdot \Pr(k|((h_m, h_y)_i)_{i=1\ldots n-1} \quad (1)$$

[1] The assumption that the distribution of the noise is Gaussian is not restrictive. If it is not, one uses the same equations given in this section, except that Eq. (3) must be consequently adapted to the actual (or supposed) distribution of the noise.

[2] For sake of simplicity, we continue to use the notation $\Pr(\cdot)$ in the next equations while this actually denotes a term which is proportional to the actual probability.

Considering that the observed Hamming weights can be issued from any possible actual ones, the multiplicative term $\Pr((h_m, h_y)|k)$ can be computed thanks to the law of total probabilities as:

$$\Pr((h_m, h_y)|k) = \sum_{h^*_m, h^*_y} \Pr((h_m, h_y)|(h^*_m, h^*_y)) \cdot \Pr((h^*_m, h^*_y)|k) \qquad (2)$$

The second term of the product comes from the same precomputed models as exploited in the original method, while the first term is simply the probability of the noise that accounts for the observation:

$$\begin{aligned}\Pr((h_m, h_y)|(h^*_m, h^*_y)) &= \Pr(\omega_m = h_m - h^*_m) \cdot \Pr(\omega_y = h_y - h^*_y) \\ &= \left(\frac{1}{\sigma_m \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{h_m - h^*_m}{\sigma_m}\right)^2} \right) \cdot \left(\frac{1}{\sigma_y \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{h_y - h^*_y}{\sigma_y}\right)^2} \right) \qquad (3)\end{aligned}$$

Equations (1) to (3) allow to compute the probability distribution of the key given the observed Hamming weights. This exploits the full information that can be derived from the measurements. Based on this distribution, the attacker simply selects the key with highest probability.

3.2 Estimating the Hamming Weights

Section 2.2 describes Linge's "slice" method for converting leakages to Hamming weights. While clever and quite simple to apply, its main drawback is that it estimates Hamming weights as integers, so that the process may arbitrarily assign two different Hamming weights to two quite near (possibly even equal) leakages. While such integer values can be used in Equations (1) to (3), we see this threshold effect as undesirable since the maximum likelihood method can take advantage of a more smooth estimation without such rounding inaccuracies. We now propose two methods for converting real-valued leakages to real-valued Hamming weights.

Linear Regression According to the linear model $\ell = \alpha\,\mathrm{HW}(v) + \beta$, it is possible to estimate the Hamming weight from the leakage ℓ as soon as we know – or have estimated – the values of the constant coefficients α and β. Linear regression infers from two series $(\ell_i)_i$ and $(\mathrm{HW}(v_i))_i$ of leakages and corresponding Hamming weights the coefficients α and β of the linear relationship that best fits the set of points. Unfortunately this requires the knowledge of the byte values v_i corresponding to each leakage l_i. This means that this method preferably applies during a characterization phase on a known-key device, the inferred coefficient values being subsequently used for the attack on a similar target device with an unknown key.

Variance Analysis As for linear regression, our second method for converting leakages to Hamming weights also estimates α and β. However, as far as we know this is the first proposed method that can estimate these coefficients without the knowledge of the key or the plaintexts/ciphertexts. It does not need them because it is not required to know which v_i corresponds to which ℓ_i.

From a large set of execution traces with varying inputs it is possible to compute the variance (or the standard deviation) of the leakage at each instant. Usually, this variance trace clearly shows two kinds of time samples. Those for which the variance is low, which correspond to a low activity of the device, or at least to a constant activity independent from the algorithm input. At these instants we consider that the variance level reflects the variance of the measurement noise on the leakage. On the other hand when the activity is related to a data that depends on the algorithm input, then the variance is quite larger as it also includes that of the manipulated data. This is illustrated in Figure 2 where three groups of 16 peaks correspond to the standard deviation when m, x and y bytes are manipulated, while the initial portion up to time sample 30 000 corresponds to a low activity process.

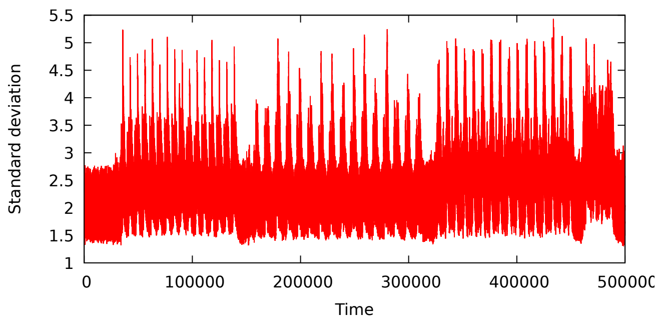


Fig. 2: Standard deviation trace computed on 1000 executions

From the measurement of the variance levels both on a quiet part and at the point of interest for the attack, one can derive the coefficient α of the leakage model. The noisy leakage expresses as: $\ell = \alpha\,\mathrm{HW}(v) + \beta + \omega$. Due to the independence of the noise from the data, we have:

$$\mathrm{Var}(\ell) = \mathrm{Var}(\alpha\,\mathrm{HW}(v) + \beta) + \mathrm{Var}(\omega) = \alpha^2\,\mathrm{Var}(\mathrm{HW}(v)) + \mathrm{Var}(\omega) \quad (4)$$

As v is a random byte value the variance of its Hamming weight is equal to 8 times the variance of a uniformly distributed bit, that is $\mathrm{Var}(\mathrm{HW}(v)) = 8 \times \frac{1}{4} = 2$. We can now derive α from Eq. (4):

$$\alpha = \pm\sqrt{\big((\mathrm{Var}(\ell) - \mathrm{Var}(\omega)\big)/2} \tag{5}$$

Once α is known, the value of β can be inferred from the model as $\beta = \mathrm{E}(\ell) - \alpha\,\mathrm{E}(\mathrm{HW}(v)) = \mathrm{E}(\ell) - 4\alpha$, where $\mathrm{E}(\ell)$ is estimated by the average leakage at the considered point of interest. Finally, from α and β, a leakage ℓ can be converted to the estimated Hamming weight $h = \frac{(\ell-\beta)}{\alpha}$.

3.3 Experimental Results

In this section we provide experimental results that compare the original distance-based method with that based on the ML criterion. We performed simulations where m is generated at random uniformly and $y = S(m \oplus k)$ is derived from m and from the key byte to be recovered (k is drawn at random for each run). We generated our observations by adding two independent Gaussian noises with same variance to $\mathrm{HW}(m)$ and $\mathrm{HW}(y)$.

Based on the sets of real-valued Hamming weights $(h_m)_i$ and $(h_y)_i$, we computed integer versions of them suitable for applying the distance method. To that end we applied Linge's slice method to the real-valued Hamming weights in a same manner as if they were original leakages. Note that applying the slice method to the real-valued Hamming weights is strictly equivalent to applying it to the leakages from which they are supposed to be linearly derived.

The left part of Fig. 3 presents the results in terms of the average rank of the correct key based on 10 000 runs with a medium noise level of $\sigma = 1.0$. Drawings in plain line style refer to the "slice" way to derive integer Hamming weights from real-valued ones. Blue, green and gray lines refer to the Inner Product, to the Pearson χ^2 and to the Euclidean distances respectively. Red lines refer to the ML criterion for which we also show in dotted line style the results when using directly the real-valued Hamming weights. In the case of the ML we used the same noise level $\sigma = 1.0$ for the attack phase as we used to generate the observations.

We can clearly see that IP and Euclidean distances do not give good results whereas the Pearson χ^2 based distance gives better ones. Also, ML strongly outperforms all distance-based methods, particularly when used with original real values. For the maximum likelihood the average rank is about 5 with 1000 observations, and below 2 with only 2000 observations.

These simulation results demonstrate that the maximum likelihood approach is superior to the distance based one in two respects: (i) it is intrinsically better when compared with the same observations (integer Hamming weights generated by the slice process) and further, (ii) it can take great advantage of real-valued Hamming weight estimations that can be directly inferred from the measured leakage.

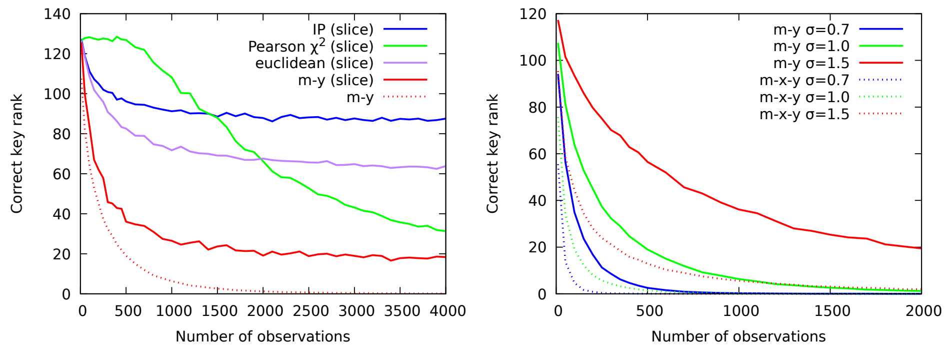


Fig. 3: Left: comparison of original distance method and *m*-*y* maximum likelihood ($\sigma = 1.0$). Right: comparison of *m*-*y* and *m*-*x*-*y* variants for different levels of noise.

Variants with more Points of Interest While the joint Hamming weights distribution analysis has been presented in Section 3.1 with two observed leakages (namely m and y), it is possible to use more of them if available. For example we can use the joint leakage from the three points of interest of m, $x = m \oplus k$ and $y = S(m \oplus k)$. Such so-called *m*-*x*-*y* attack is a straightforward generalization of the *m*-*y* attack where the theoretical models contain values of $\Pr((h^*_m, h^*_x, h^*_y)|k)$ instead of $\Pr((h^*_m, h^*_y)|k)$, where the summation of Eq. (2) is over all triplets (h^*_m, h^*_x, h^*_y)[3], and where the conditional probability of the observation in Eq. (3) includes an extra term corresponding to $\Pr(\omega_x = h_x - h^*_x)$.

The right part of Fig. 3 compares both *m*-*y* (plain lines) and *m*-*x*-*y* (dotted lines) variants of the maximum likelihood attack with three noise levels σ=0.7 (blue), σ=1.0 (green) and σ=1.5 (red). Notice that for a same noise level the *m*-*x*-*y* attack is quite more efficient than the *m*-*y* one. This

[3] While this can be viewed as a multiplication by 9 of the terms in the summation, it is worth to note that $\Pr((h^*_m, h^*_x, h^*_y)|k)$ is non null for at most 256 triplets.

is because the observation of h_x brings extra information that helps to further discriminate candidate keys. We also observe that the effect of the noise is important as it requires about five times more observations to get the same reliability on the key for σ=1.5 than for σ=1.0.

It is also possible to use other points of interest. For example, the y value is subsequently used in the `MixColumns` operation. Thus, depending on the implementation, there may exist instants where $2y$ and $3y$ are also manipulated. We have studied variants of the attack where these variables are included in the analysis. This results in attacks of types m-y-$2y$, m-y-$3y$, m-y-$2y$-$3y$ and the same ones with x also. The simulation results show that adding more variables to the analysis always gives better results, but this gain is smaller for $3y$ than for x, and even smaller for $2y$ [4].

Variant *m-x* We now present a particular variant of the joint distribution analysis which involves only the leakages of m and x. This variant is special in the sense that if one computes the theoretical models for all possible k then one observes that they form classes of indistinguishable models, with each class being specific to the Hamming weight of k. That means that the distribution of $(\mathrm{HW}(m), \mathrm{HW}(m \oplus k))$ only depends on $\mathrm{HW}(k)$. This property is not so surprising, and comes from the fact that the XOR operation acts on bits independently from each other and that the Hamming weight function is invariant by any permutation of the bits.

There are two practical consequences of this property. First, the amount of information that can be retrieved from a m-x joint distribution analysis is less than for the m-y variant (about 2.5 bits instead of 8 bits on average). The second consequence is that the m-x attack retrieves $h_k = \mathrm{HW}(k)$ more efficiently than the m-y attack retrieves k. This is due not only to the fact that there are only 9 models to distinguish from, but also to the fact the those models are more different from each other.

As an illustrative example, the top of Fig. 4 shows the models $\mathcal{M}_{h_k}$ for values 0, 1 and 2 of h_k. One can observe that these distributions show a characteristic pattern made of respectively 1, 2 and 3 parallel and linear structures like "walls".

Simulation results for the m-x variant are presented on the bottom of Fig. 4. The correct Hamming weight of the key is "first ranked" (arbitrarily, say a mean rank less than 0.2) with less than 100, 200 and 500 traces for noise levels σ equal to 0.7, 1.0 and 1.5 respectively.

[4] This last observation can be explained by the fact that information brought by y and $2y$ are somewhat redundant. Indeed their Hamming weights are quite correlated since they are equal for all $y < 128$ values for which $2y$ is equal to $y \ll 1$.

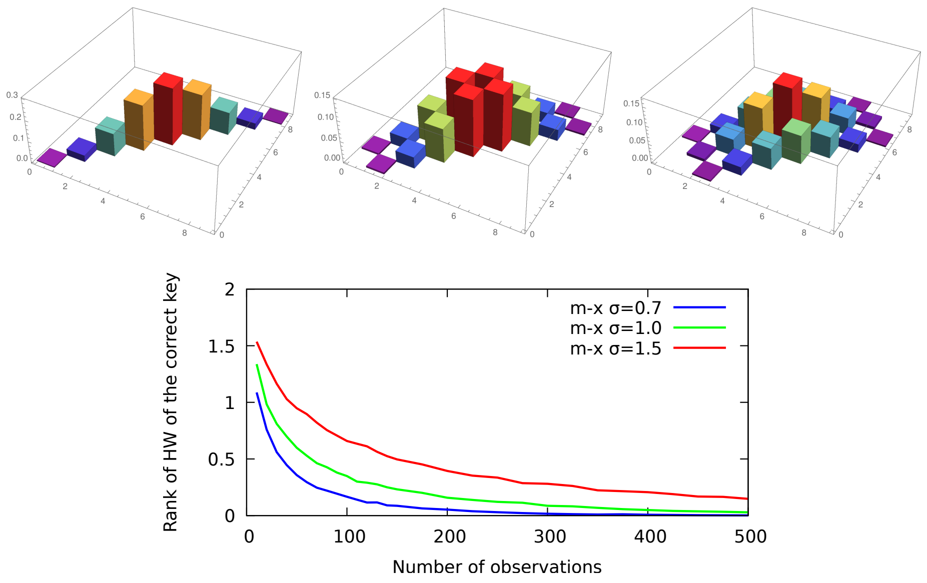


Fig. 4: Top: joint distributions of $(\mathrm{HW}(m), \mathrm{HW}(x))$ for $\mathrm{HW}(k)$ equal to 0, 1 and 2. Bottom: simulation results of the m-x variant for different noise levels.

4 Implementations Protected by Boolean Masking

Both Linge's and the maximum likelihood methods presented in Section 3 require a non protected implementation. Notably, they can not recover the key in the presence of the Boolean masking countermeasure [10,19]. This defense technique prevents from classical statistical attacks by XOR-masking all intermediate state bytes of the ciphering path with a random mask byte which is refreshed at every execution. To do so, it is necessary to generate a modified S-Box S' designed to produce a masked version $y' = y \oplus r_{out}$ of the normal output $y = S(x)$ when it receives a masked input $x' = x \oplus r_{in}$. The modified S-Box is thus defined as $y' = S'(x') = S(x' \oplus r_{in}) \oplus r_{out}$. From the measured leakages $\ell_{m'}$ and $\ell_{y'}$ the attacker infers a masked couple $(\mathrm{HW}(m'), \mathrm{HW}(y'))$ which is differently distributed than the couple $(\mathrm{HW}(m), \mathrm{HW}(y))$ based on which the models are defined.

4.1 Variants m-y and m-x-y

Figure 5 presents different options for implementing the Boolean masking countermeasure. We focus here on the area involving the XOR with the key and the S-Box. These schemes differ according to whether the key itself is masked or not, and whether the input and output masks of the S-Box are the same or not.

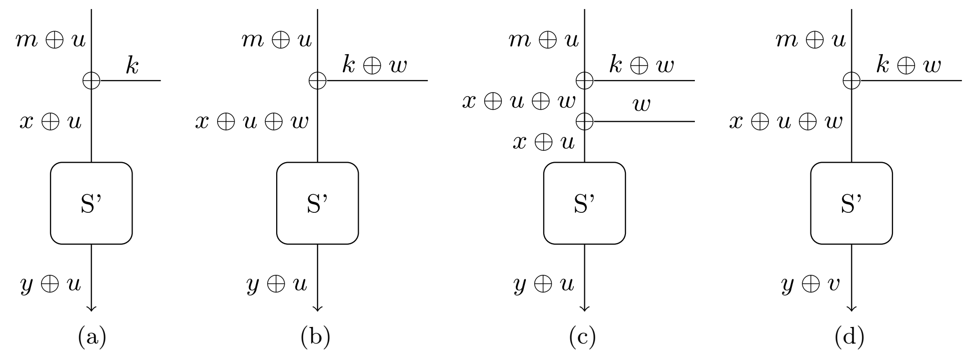


Fig. 5: Examples of Boolean masking schemes

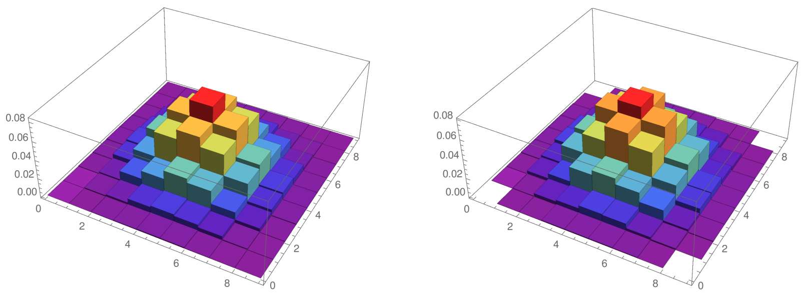
When an attacker performs the first-order joint distribution analysis on an implementation protected by first-order Boolean masking, he generates an empirical distribution of masked couples $(\mathrm{HW}(m'), \mathrm{HW}(y'))$ and "compares"[5] it to a distribution of non masked couples $(\mathrm{HW}(m), \mathrm{HW}(y))$. The consequence is that the empirical distribution will not match the models even for the correct key.

In the case where m and y are masked by the same value (schemes a, b and c of Fig. 5), it is possible to recover the consistency between both empirical and theoretical distributions if we define the models as being distributed according to the distribution of masked couples with an uniformly distributed m and an uniformly distributed mask u.

Thus, it is possible to adapt the joint distribution attack to such masked implementations and the only modification consists in creating the models in a way that fits with the distribution of the couples of masked Hamming weights. Precisely, there still exists 256 models $\mathcal{M}_k$, one per key byte, and each model contains the conditional probabilities $\Pr((\mathrm{HW}(m'), \mathrm{HW}(y'))|k)$. But in this case, these probabilities are computed by counting the number of occurrences of each couple of Hamming weights when both m and the mask u range over all byte values. These models still mutually differ but they are much more similar to each other

[5] This comparison is either explicit (Linge's distances) or implicit (ML).

than for the non-masked case. This is illustrated on the top of Fig. 6 which presents the models for the same example keys than those presented in Fig. 1. We verified that all 256 models are different from each other.



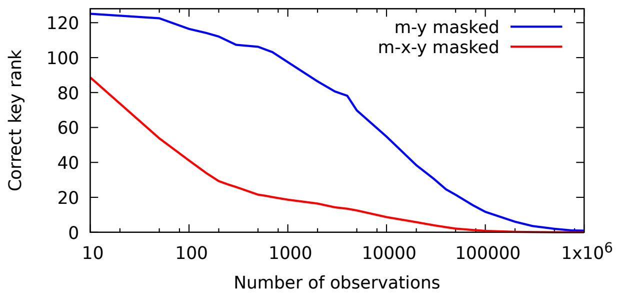


Fig. 6: Top: joint distributions of $(\mathrm{HW}(m'), \mathrm{HW}(y'))$ for $k = 39$ (left) and $k = 167$ (right). Bottom: comparison of second-order m-x and m-x-y variants ($\sigma = 1.0$)

The greater similarity between theoretical distributions in the masked case induces a much larger number of observations that are needed to distinguish between them. This is due to the fact that one must wait longer before the empirical distribution converges toward its model.

Notice that a m-x-y variant of such second-order joint distribution analysis is also possible provided that all three intermediate state bytes are masked by the same value. This is notably the case for schemes a and c of Fig. 5. We present experimental results for both m-y and m-x-y variants of the second-order joint distribution analysis (with ML criterion) on the bottom of Fig. 6. These simulation results were obtained by averaging the rank of the correct key over 1000 runs with a noise level equal to σ=1.0.

Here also the m-x-y variant is more efficient than the m-y one. We also observe that the number of traces needed to recover the key is much more than for the first-order attack. Nevertheless, this demonstrates that joint distribution analysis also works on masked implementations provided that relevant variables are masked by the same value.

4.2 Variant m-x

We have seen in Section 3.3 that the m-x variant is particular in the sense that it allows to recover the Hamming weight of k instead of k itself. Another remarkable and important property of this variant is that it is exactly as efficient when applied to masked values $m' = m \oplus u$ and $x' = x \oplus u$ as it is when applied directly to m and x. This means that masking is totally useless with respect to this attack. The reason is that both joint distributions of $(\mathrm{HW}(m), \mathrm{HW}(x))$ and $(\mathrm{HW}(m'), \mathrm{HW}(x'))$ are the same. This is because both series of (m, x) and (m', x') are the same in a permuted order, so are equal the series of their Hamming weights.

We stress on the importance of this special behavior: even an implementation protected by Boolean masking is vulnerable to the m-x variant which can recover the Hamming weight of the key byte with about only few hundreds traces. Again, this is only true if m and x are masked by the same value, which is the case of schemes a and c of Fig. 5.

5 Joint Distribution Analysis with Knowledge of the Plaintext

As stated by Eq. (2), the joint distribution analysis with ML criterion uses the conditional distributions $\Pr((h_m^*, h_y^*)|k)$ of the Hamming weights given the key. These models are built in a precomputation phase by counting, for the given key, the number of occurrences of each Hamming weight couple (or triplet for a m-x-y attack) for all possible values of m, and possibly all values of the mask u in the case of a masked implementation.

In this section we study how to adapt this attack to the classical case where the plaintext is known from the attacker. Of course, contrarily to the blind context, the attack is now feasible only on the first round.

5.1 First-order Attack

When m is known it is no more useful to include h_m in the observation, and we work with the probability distribution of h_y^* (or of (h_x^*, h_y^*) for a m-x-y attack) for given values of k and m. Note that in this case the distribution is degenerated as a unique h_y^* value (or a unique (h_x^*, h_y^*) couple) resulting from k and m. Though, the computation of the probability distribution of k given the observations remains feasible in a similar way by summing over only h_y^* (or (h_x^*, h_y^*)) in Eq. (2).

We have simulated the m-y attack on 1000 runs and compared it with the classical CPA. Results are presented on the left part of Fig. 7 for σ noise levels of 1.0, 3.0 and 5.0 respectively. We note that retrieving the key by the ML method is slightly more efficient than by CPA. On the other hand, CPA does not require the determination of the point of interest.

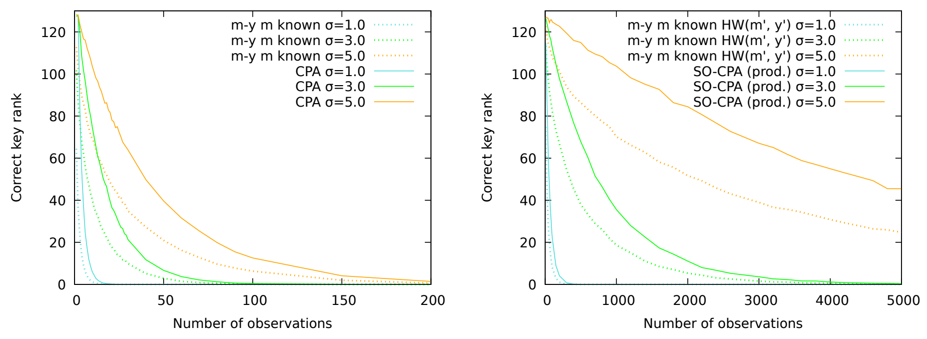


Fig. 7: Left: comparison of m-y attack (with m known) to CPA. Right: comparison of 2^{nd}-order m-y attack (with m known) to 2^{nd}-order CPA. Variant with points of interest $m' = m \oplus u$ and $y' = y \oplus u$.

5.2 Second-order Attack

In the case of a masked implementation, a first option is also to straightforwardly adapt the attack to the knowledge of m. The attack takes m' and y' as points of interest. For each couple (k, m) we precompute a corresponding model that gives the distribution of $(h_{m'}^*, h_{y'}^*)$ when only the mask u varies.

As an alternative method, one can substitute the observation of m' by that of the random value that masks y. The two points of interest are then u and $y' = y \oplus u$, and the models correspond to the distribution of $(h_u^*, h_{y'}^*)$. A great advantage of this variant is that it applies even when two independent masks are added to m and y. On the other hand, it requires to identify the point of interest of the mask, which may be difficult.

We have simulated both variants on 1000 runs. In each case we compare the ML method with the 2^{nd}-order CPA where the combination of the leakages (centered product[6]) is correlated with $\mathrm{HW}(m \oplus y)$ in the first case, and with $\mathrm{HW}(y)$ in the second case. Both variants give almost the same results, which is not surprising as when m is known, the information brought by $m' = m \oplus u$ and by u are essentially the same. The right part of Fig. 7 presents the results for the variant which exploits m' and y'. Note that the ML method finds the key somewhat earlier than 2^{nd}-order CPA.

6 Concrete Experiments

In this section we present concrete experiments on side-channel traces captured from a real device. We have implemented two versions of a software AES on an Arduino Uno 8-bit microcontroller. The first version does not feature any countermeasure while the other one implements Boolean masking with the same mask on m, x and y.

We present two attacks: a m-y attack with unknown plaintext on the naive implementation, and a m-y attack with known plaintext on the masked implementation (variant with points of interest on m' and y').

Traces were acquired on a Lecroy WaveRunner oscilloscope with a sampling rate of 5 GS/s. The 1000 traces for the first attack and the 200 traces for the second one were perfectly aligned and the points of interest were blindly determined based on the highest peaks of the standard deviation trace. Figure 2 shows the computed trace that was used for the first attack. It clearly shows three groups of 16 peaks. The points of interest corresponding to m and y bytes were easily identified by assuming that the three groups correspond to manipulations of m, x and y respectively. For the second attack the points of interest for m' and y' where identified similarly based on a standard deviation trace that exhibits four groups of peaks corresponding to successive manipulations of m, m', x' and y'.

We used the variance analysis method of Section 3.2 to derive α and β coefficients at each point of interest. Based on the first part of the standard deviation trace of Figure 2, we have estimated the standard deviation (in leakage unit[7]) of the noise by visual inspection, and we choose the value 2.0 which approximately lies in the middle of the vertical range. This procedure resulted in the same estimated value for the second attack.

[6] Given two leakages ℓ_1 and ℓ_2 the *centered product* combining function computes $f(\ell_1, \ell_2) = (\ell_1 - \mathrm{E}(\ell_1)) \times (\ell_2 - \mathrm{E}(\ell_2))$. The *absolute value of centered difference* combining function defined by $g(\ell_1, \ell_2) = |(\ell_1 - \mathrm{E}(\ell_1)) - (\ell_2 - \mathrm{E}(\ell_2))|$ has also been considered but shows to be less efficient than the centered product.

[7] σ of the noise on the leakage and that on the Hamming weight are equal up to the factor $|\alpha|$. It is thus expressed in leakage unit or in bit unit according to the context.

Notice that when α is derived from Eq. (5), the attacker must decide its sign. If he does not know which sign is correct for α_m nor for α_y, he must perform the attack four times, and the four sorted lists of key candidates must be interleaved when trying to find to correct whole key by key enumeration. In our case, a prior characterization of the device revealed that α is negative for all points of interest as explained in Appendix A.

Table 1 gives the ranks of correct key bytes for the m-y attack with unknown plaintexts on the unprotected implementation. For comparison purpose the attack has also been performed with the distance-based method using Inner Product and Euclidean distances [8]. Except for six bytes, the maximum likelihood always finds the correct key in the first 10 positions, whereas the distance based attacks are quite less efficient. Note that the standard deviation of the noise (in bit unit) was more or less equal to $\sigma = 0.7$ for each byte. Simulations show that for this noise level the average rank is about 58 for the Inner Product, about 29 for the Euclidean distance, and close to 0.25 for the maximum likelihood.

Table 1: Rank of the correct key byte for a m-y attack with unknown plaintexts on an unprotected implementation (1000 traces)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
In. Prod.	**0**	**0**	167	29	45	187	192	45	77	108	36	124	5	104	64	147
Eucl. Dist.	1	80	210	106	3	62	186	17	38	68	194	48	27	120	21	116
ML (slice)	1	1	29	**0**	**1**	46	**1**	**0**	**1**	32	36	19	26	67	66	28
ML (var.)	**0**	2	**6**	1	**1**	**17**	**1**	**0**	**1**	**19**	**5**	**15**	**4**	**40**	**19**	**13**

Similarly, Table 2 gives the ranks of correct key bytes for the m-y attack with known plaintexts on the masked implementation. ML and centered product 2nd-order CPA give similar almost perfect results. For comparison, we also provide results for the absolute value of centered difference combining function which show to be globally less efficient.

7 Applications and Possible Countermeasures

7.1 Applications

We now present three applications of the attacks presented in this paper.

[8] The Pearson χ^2 distances were impossible to compute due to an insufficient number of traces.

Table 2: Rank of the correct key byte for a m-y attack with known plaintexts on a masked implementation (200 traces)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SO-CPA (abs. diff.)	**0**	5	1	213	**0**	109	**0**	75	**0**	**0**	**0**	58	**0**	1	3	**0**
SO-CPA (product)	**0**	**0**	6	64	**0**	16	**0**	**0**	1	**0**	**0**	**2**	**0**	**0**	**0**	**0**
ML (variance)	1	**0**	**0**	**12**	**0**	**5**	**0**	**0**	**0**	**0**	**0**	9	**0**	**0**	**0**	**0**

1. Our first application relates to the AES-based[9] EMV session key derivation function depicted on the left of Fig. 8. As Linge et al. already noted this scheme resists to classical side-channel analyses like DPA or CPA. An attacker who wants to recover the master key would target the first AES. Unfortunately, its output is not known since this is the session key. It is thus impossible to perform an attack at the last round. It is also impossible to attack the first round except on the first two key bytes since the 14 remaining input bytes are constant. Linge et al. also observed that, contrarily to DPA and CPA, joint distribution analysis can be used to recover the master key. Indeed after two rounds all state bytes can be considered to vary uniformly. It is thus possible to apply their attack e.g. at the third round to retrieve the value of K_3. While Linge's attack is restricted to naive implementations, our m-y and m-x-y variants presented in Section 4.1 can do the same on implementations protected by first-order Boolean masking.

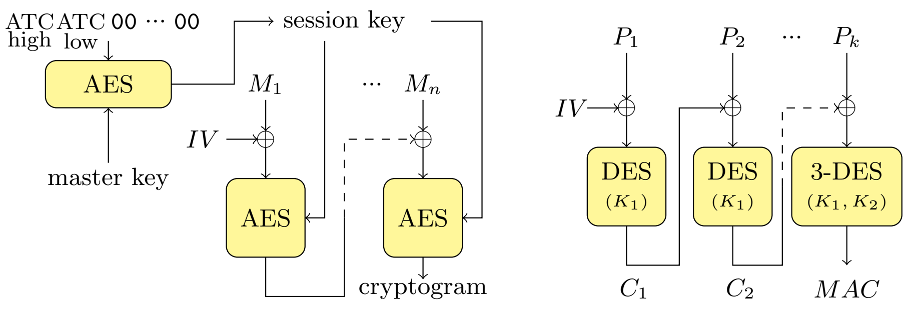


Fig. 8: Left: EMV session key derivation scheme. Right: ISO/IEC 9797-1 MAC scheme using 3-DES algorithm.

[9] EMV scheme also allows to use the Triple DES function.

2. EMV session key derivation can also be attacked by the *m-x* variant. Since this variant only recovers the Hamming weights of the key bytes, applying it on the 16 bytes of a round key is not sufficient as this brings an average of only about 20 bits of information. Instead we can perform the attack at all rounds, which recovers the Hamming weights of all 176 bytes of the expanded key. While this is much more information about the key, one would wonder whether this information can be efficiently exploited to retrieve the ciphering key K. It has been shown [2] that a branch-and-bound like algorithm can recover K quite efficiently from part of these Hamming weights. This algorithm can also deal with some errors in the estimation of the Hamming weights. It is thus possible to recover an AES key on a Boolean masked implementation with no extra cost compared to a naive one.

3. In [6] Feix et al. show that a fixed key used to compute a cryptogram with the standard scheme ISO/IEC 9797-1 MAC algorithm using 3-DES algorithm can be compromised even if the DES itself is implemented in a secure way. The right part of Fig. 8 shows how the MAC value is computed from a k-bloc plaintext $P = (P_1, \ldots, P_k)$ and a 112-bit secret key $K = (K_1, K_2)$. Their attack obtains side-channel information outside the DES function, at the protocol level. Precisely, if an attacker fixes the first n plaintext blocks and lets P_{n+1} vary, then the fixed value of the intermediate ciphertext block C_n can be recovered by correlating, byte per byte, the known value P_{n+1} with the leakage of $C_n \oplus P_{n+1}$. Once C_n is known, K_1 can be retrieved by a 56-bit exhaustive search against a known plaintext/ciphertext pair. Once K_1 is known, K_2 is also recovered by a 56-bit exhaustive search.

   A proposed fix to this attack, which consists in applying a Boolean masking on all plaintext blocks, has later been proven vulnerable to 2^{nd}-order analysis [5] if the masks do not have maximal entropy. The authors show that one can jeopardize a masked implementation in the two following cases: (i) a *same* 8-byte mask block $M = (R_0, R_1, \ldots, R_7)$ is used to mask all P_i blocks, or (ii) all P_i are masked with different mask blocks $M_i = (R_i, R_i, \ldots, R_i)$ made of a *same* repeated random byte. They notice that the attack does not work when all mask blocks $M_i = (R_{i,0}, \ldots, R_{i,7})$ are different and made of different bytes, and consequently recommend this full entropy masking.

   In the case of careful Boolean masking with full entropy, we observe that both P_{n+1} and $P_{n+1} \oplus C_n$ are still masked by the same value M_{n+1}. It is then possible to mount an *m-x* type joint distribution analysis[10] which reveals the Hamming weights of each C_n byte. One can obtain similar information for several plaintexts $(P_1, \ldots, P_n)$ and use them all in the exhaustive search phase. More precisely, any key candidate that complies with the Hamming weights of the first pair will be checked against those of the second one, and so on.

7.2 Possible Countermeasures

As stated in Sections 4.1 and 4.2, it is possible to apply the joint distribution analysis to implementations protected by first-order masking. Yet, a requirement for all m-y, m-x-y and m-x variants is that the targeted variables are all masked with a same value. As a result, the masking scheme d of Fig. 5 is not vulnerable to our attacks since m, x and y are all masked by independent random values. We thus recommend this masking scheme or any other one which would share the same property.

We also recommend any countermeasure that introduces time randomization – like random delays or shuffling of independent operations – and spoils the notion of point of interest or make them difficult to identify.

8 Conclusion

We demonstrated that the maximum likelihood method better exploits couples of Hamming weights in Linge's joint distribution analysis. Given a set of observed couples of Hamming weights it computes the posterior probability of each key candidate and selects the most probable of them. We have studied the non trivial problem of inferring Hamming weights from leakages and described a new method based on variance analysis that does not require the knowledge of the key and the plaintexts/ciphertexts (contrarily to linear regression).

We compared the ML approach to Linge's technique based on distances between distributions and showed, by simulations and concrete experiments, that it recovers the key value more efficiently. We derived several variants – m-x-y, m-x and others – of the original m-y attack and adapted the generation of theoretical models to make this attack work in the presence of Boolean masking. We noticed a remarkable property of the m-x attack that applies equally well on naive and masked implementations.

[10] A classical second-order CPA on the pair of leakages of $(P_{n+1} \oplus M_{n+1}, P_{n+1} \oplus M_{n+1} \oplus C_n)$ is not possible in this case as it would imply to correlate the combination of these leakages with the Hamming weight of C_n which does not vary.

We proposed new applications of our attacks that can threaten the EMV session key derivation even on protected implementations, and we proposed implementation guidelines in order to thwart our attacks or at least make them quite difficult.

As future works, it could be interesting to study how the ML criterion can deal with non Gaussian noises and with non linear leakage functions.

Acknowledgements

The authors would like to thank Yanis Linge, Antoine Wurcker and Benoit Feix for fruitful discussions about the attacks presented in this paper.

References

1. Éric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES '04*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, 2004.
2. Christophe Clavier, Damien Marion, and Antoine Wurcker. Simple Power Analysis on AES Key Expansion Revisited. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES '14*, volume 8731 of *Lecture Notes in Computer Science*, pages 279–297. Springer, 2014.
3. François Durvaux and François-Xavier Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT '16*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.
4. EMV Co. EMV Integrated Circuit Card Specifications for Payment Systems, Book 2, Security and Key Management, November 2011. Version 4.3.
5. Benoit Feix, Andjy Ricart, Benjamin Timon, and Lucille Tordella. Defeating Embedded Cryptographic Protocols by Combining Second-Order with Brute Force. In *Cardis 2016*.
6. Benoit Feix and Hugues Thiebeauld. Defeating ISO9797-1 MAC Algo 3 by Combining Side-Channel and Brute Force Techniques. *IACR Cryptology ePrint Archive*, Report 2014/702, 2014.
7. Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES '08*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
8. Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. Stochastic Methods. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES '06*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29. Springer-Verlag, 2006.
9. Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A Testing Methodology for Side-Channel Resitance Validation. In *NIST Non-invasing attack testing workshop*, 2011.

10. Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer-Verlag, 1999.
11. Neil Hanley, Michael Tunstall, and William P. Marnane. Unknown Plaintext Template Attacks. In Heung Youl Youm and Moti Yung, editors, *Workshop on Information Security Applications – WISA '09*, volume 5932 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2009.
12. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
13. Roman Korkikian, Sylvain Pelissier, and David Naccache. Blind Fault Attack against SPN Ciphers. In Assia Tria and Dooho Choi, editors, *Fault Diagnosis and Tolerance in Cryptography – FDTC '14*, pages 94–103. IEEE Computer Society Press, 2014.
14. Hélène le Bouder. *Un formalisme unifiant les attaques physiques sur circuits cryptographiques et son exploitation afin de comparer et rechercher de nouvelles attaques.* PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne, 2014.
15. Yang Li, Mengting Chen, Zhe Liu, and Jian Wang. Reduction in the Number of Fault Injections for Blind Fault Attack on SPN Block Ciphers. *ACM Trans. Embedded Comput. Syst.*, 16(2):55:1–55:20, 2017.
16. Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the Joint Distributions of a Cryptographic Function in Side Channel Analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design – COSADE '14*, volume 8622 of *Lecture Notes in Computer Science*, pages 199–213. Springer, 2014.
17. Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT '13*, Lecture Notes in Computer Science, pages 486–505. Springer-Verlag, 2013.
18. National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standard #197, 2001.
19. Kai Schramm and Christof Paar. Higher Order Masking of the AES. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA '06*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer-Verlag, 2006.

A Determination of the sign of α

The sign of α indicates whether the leakage function linearly increases ($\alpha > 0$) or decreases ($\alpha < 0$) with the Hamming weight of the data. Experiments on our device with known plaintexts and a known key demonstrated that positive CPA peaks always occur on the descending part of the leakage during the clock cycle, while negative peaks always occur on its ascending part. This is clearly visible in Fig. 9 where the power consumption and the CPA traces are depicted in red and green respectively.

On the same figure one can notice that the standard deviation peaks (in blue) may occur either on positive or negative CPA peaks. Deciding whether a standard deviation peak corresponds to a positive or negative α value simply consists in observing whether it matches with a falling or a raising edge of the leakage respectively. In the experiments described in Section 6 we observed that the selected points of interest – defined by the highest standard deviation peaks – always correspond to the ascending part of the clock cycle leakage, which means a negative α value.

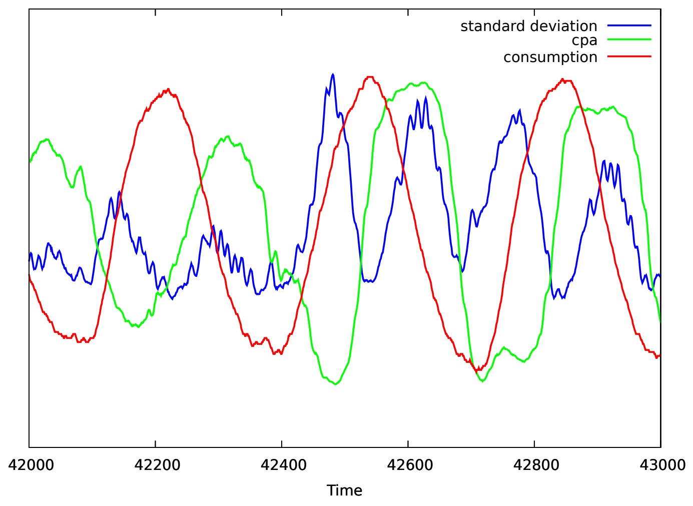


Fig. 9: Relation between the correlation sign and the raising/falling part of the leakage