Behind the Scene of Side Channel Attacks — Extended Version —

Victor Lomné, Emmanuel Prouff, and Thomas Roche

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

Abstract. Since the introduction of side channel attacks in the nineties, a large amount of work has been devoted to their effectiveness and efficiency improvements. On the one side, general results and conclusions are drawn in theoretical frameworks, but the latter ones are often set in a too ideal context to capture the full complexity of an attack performed in real conditions. On the other side, practical improvements are proposed for specific contexts but the big picture is often put aside, which makes them difficult to adapt to different contexts. This paper tries to bridge the gap between both worlds. We specifically investigate which kind of issues is faced by a security evaluator when performing a state of the art attack. This analysis leads us to focus on the very common situation where the exact time of the sensitive processing is drown in a large number of leakage points. In this context we propose new ideas to improve the effectiveness and/or efficiency of the three considered attacks. In the particular case of stochastic attacks, we show that the existing literature, essentially developed under the assumption that the exact sensitive time is known, cannot be directly applied when the latter assumption is relaxed. To deal with this issue, we propose an improvement which makes stochastic attack a real alternative to the classical correlation power analysis. Our study is illustrated by various attack experiments performed on several copies of three micro-controllers with different CMOS technologies (respectively 350, 130 and 90 nanometers).

1 Introduction

Since the seminal Differential Power Analysis of Kocher *et al.* [18], various Side Channel Attacks (SCA) have been proposed and improved (*e.g.* [8, 9, 11, 12, 36]). In order to to compare and classify them, **theoretical** frameworks have then been introduced [11, 23, 40, 44]. Their main purpose is to identify the attacks similarities and differences, and to exhibit contexts where one is better than another. They have laid the foundation stones for a general comparison and evaluation framework. In parallel, several **practical** works have addressed issues arising when applying an SCA in the real world (*e.g.* in an industrial context) [2, 5, 17, 25, 42]. Those works essentially attempt to fill the gap between the theoretical analysis of the attacks and their application in non-idealized contexts. However, whereas the published theoretical analyses usually tend towards generic

and formal statements (sometimes at the cost of too simple models), many of the practical analyses only focus on a particular attack specificity and often put the big picture aside. The latter analyses are indeed usually dedicated to one specific attack running against a specific target device, which makes them hard to generalize. This paper tries to be at the intersection of both worlds: we study practice-driven issues while keeping a generic approach w.r.t. attacks mechanisms and targeted platforms. This approach and our final purpose are close to those in the works of Standaert *et al.* [38] and Renauld *et al.* [33].

The starting observation of our study is that side channel traces are never reduced to one point in practice, even when they rely on the manipulation of a single variable. In contrary, those traces are often composed of a large number of points (typically several thousands). In spite of the evidence of this observation, it is rarely taken into account when analysing the effectiveness of a side channel attack. Such an analysis is indeed frequently done under the assumption, sometimes implicit, that a small number of points of interest (POI) has already been extracted from the traces either by pattern matching [22], or by dimension reduction [1, 6, 7, 37] or thanks to a previous successful attack [10, 33]. However, the two first categories of techniques are not yet perfect and, after reduction, the traces are often still composed of several points in practice. And, what is more important, the risk of information loss during the reduction process leads most of attack practitioners to not apply them. The third technique (performing a first attack to identify the POI) allows for interesting analyses, but it does not correspond to a *real attack* context. Moreover, the best POI for one attack type may not be so good for another one. Eventually, we come to a situation where attacks are analysed in a (uni-dimensional) context which does not fit with the (multi-dimensional) reality faced by the attack practitioners.

We argue in this paper that the state-of-the-art uni-dimensional analyses cannot be straightforwardly adapted to multi-dimensional contexts, which raises new interesting issues. The selection of the most likely candidate among the results of several instantaneous attacks is one of them. Indeed when the leakage traces are composed of several points, a side channel attack against the targeted sensitive variable must be performed for each point (*a.k.a* time index) in the traces. Then, the adversary must apply a strategy to select the most likely candidate among the different *instantaneous attacks* results. A classical method is to select the one with the highest score (*e.g.* the highest correlation coefficient in a Correlation Power Analysis – `CPA`– [8]). Nevertheless we argue that this strategy can be ineffective for some attack categories, including the case of the Linear Regression Analysis (`LRA`) [11,35,36]. For the latter one, we propose a new strategy to select the most likely candidate and we demonstrate its effectiveness in practice.

Another interesting issue when dealing with a large number of high dimensional traces is the reduction of the computational complexity. Here again, some works have investigated the use of parallel computing to decrease the data processing time [4,20] but their goal was not to diminish the algorithmic complexity of the attacks. This work studies the `CPA`, the `LRA` and the Template Attacks (`TA`)

with this goal in mind. A common structure in their algorithmic description is exhibited and then used to propose a new general *modus operandi* which enables to significantly reduce the computation time when the number of traces is non-negligible.

Finally, to make sure that our analysis is consistent with the reality, we completed our investigations by several experiments performed on three microcontrollers based on different CMOS technologies (`350`, `130` and `90` nanometers process). We report here on these experiments results. We moreover use them to confirm and complete the interesting behaviours observed in [33]: (1) the leakage seems to diverge from the classical Hamming weight model as the CMOS technology tends to the nanometer scale, which makes `LRA` a promising tool for side channel evaluations of nano-scale devices and (2) `TA` is effective in practice, even when the *templates* are built on one copy of the device and the attack is done on another copy.

The paper is organized as follows. In Section 2, we introduce the theoretical background for our study and we present the outlines of our proposal. Then, three sections are dedicated to the application of our ideas to the `LRA`, `CPA` and `TA` respectively.

2 SCA: Practical Issues

In this section, we introduce some basics and we get into the specifics of the problematic focused in this paper.

2.1 Notations

Throughout this paper, random variables are denoted by large letters. A realization of a random variable, said X, is denoted by the corresponding lowercase letter, said x. A *sample* of several observations of X is denoted by $(x_i)_i$. It will sometimes be viewed as a vector defined over the definition set of X. The notation $(x_i)_i \hookleftarrow X$ denotes the instantiation of the set of observations $(x_i)_i$ from X. The *mean* of X is denoted $\mathbb{E}[X]$, its standard deviation by $\sigma[X]$ and its *variance* by $\mathrm{var}[X]$. The latter equals $\mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$. The *covariance* of two random variables X and Y is denoted by $\mathrm{cov}(X, Y)$ and satisfies $\mathrm{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. When we will need to specify the variable on which statistics are computed, we will write the variable in subscript (*e.g.* $\mathbb{E}_X[Y]$ instead of $\mathbb{E}[Y]$).

The notation $\overrightarrow{X}$ will be used to denote column vectors and $\overrightarrow{X}[u]$ will denote its u^{th} coordinate. Calligraphic letters will be used to denote a matrix. The elements of a matrix $\mathcal{M}$ will be denoted by $\mathcal{M}[i][j]$. Classical additions and multiplications (over real values, vectors or matrices) are denoted by $+$ and $\times$ respectively. Scalar-vector operations are denoted by $\cdot$ and $/$ (all the coordinates of the vector are multiplied, respectively divided, by the scalar). When applied to vectors or matrices, the symbols $\cdot^2$ and $\sqrt{\cdot}$ denote the operation consisting in computing the square (resp. the square root) of all the vector/matrix coordinates. Eventually, a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ will be called a (n, m)-function.

2.2 General Attacks Framework

In this paper, the attacks framework is described by considering that the adversary targets the manipulation of a single sensitive variable Z, but the study and results directly extend to contexts where several variables are targeted in parallel. The variable Z is supposed to functionally depend on a public variable X and a secret sub-part k such that $Z = F(X, k)$ where F is a $(n+n, m)$-function (which implies $X, k \in \mathbb{F}_2^n$ and $Z \in \mathbb{F}_2^m$). The bit-lengths n and m depend on the cryptographic algorithm and the device architecture[1].

The attacks are described under the assumption that the adversary owns N side channel traces $\overrightarrow{\ell}_0$, ..., $\overrightarrow{\ell}_{N-1}$, each of them containing information about Z. Namely, the i^{th} leakage trace $\overrightarrow{\ell}_i \hookleftarrow \overrightarrow{\mathbf{L}}$ corresponds to the processing of a public value $x_i \hookleftarrow X$ and contains information on the value $z_i \hookleftarrow Z$ such that $z_i = F(x_i, k)$. The dimension of the traces (*i.e.* the number of different instantaneous leakage points) is denoted by d. By definition, we have $d \doteq \dim \overrightarrow{\mathbf{L}}$.

When little information is known about the implementation and the device (which is usually the case in practice), the exact manipulation time of z_i cannot be precisely determined *a priori*. Also, precision in the observation often comes at the cost of a high sampling rate[2]. As a consequence, the dimension of the traces is usually high (from several thousand of points up to millions) and the attack must be repeated on all of their coordinates independently (as *e.g.* in `CPA` and `LRA`) or must consider huge traces chunks globally (as *e.g.* for `TA`). Although bearing differences, the three latter attacks follow a common process flow (see Algorithm 1 below). Starting from this generic description, this paper studies, in Sections 3, 4 and 5 respectively, the *effectiveness* and *efficiency* of the `CPA`, `LRA` and `TA` attacks. The core ideas of those analyses are presented in the two next sub-sections.

1. Pre-processing on the observations $\overrightarrow{\ell}_i$.
   $\hookrightarrow$ *extract statistics from the traces.*
2. Pre-processing on the predictions $F(x_i, \hat{k})$ for all $\hat{k}$.
   $\hookrightarrow$ *extract statistics from the predictions based on a model and/or an estimation of the leakage statistics conditioned by the values of the manipulated data.*
3. Processing on the predictions and observations.
   $\hookrightarrow$ *compute a* distinguishing value *for each key hypothesis and each instantaneous leakage time of each observation.*
4. Return a key candidate.
   $\hookrightarrow$ *select the key candidate with the maximum* distinguishing value *over all the instantaneous leakages times.*

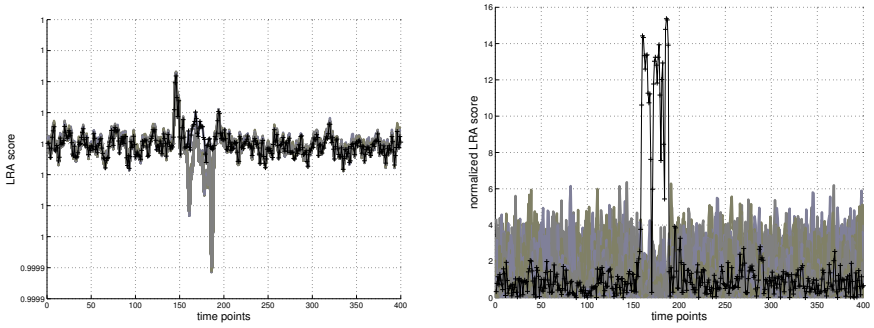
Algorithm 1: Generic Attack Flow

[1] An example of function F is the function that applies a so-called *sbox* transformation to the bitwise addition between k and X.

[2] especially in the case of Electro-Magnetic side channel measurements

2.3 Effectiveness Discussions

A part of our study is dedicated to the *distinguisher value* definition and, more precisely its relevance when considering side channel traces with a large number of points. This study was motivated by the observation that the classical `LRA` distinguisher value for one leakage time is not comparable as such to that computed for another leakage time. Figure 1(a) illustrates this claim for an `LRA` targeting the device B described in Section 2.5: when directly applying the protocol given in [11, 36], the correct key candidate does not maximize the distinguisher value globally but only in a local area, which makes the attack unsuccessful unless this area is known by the adversary (which is not assumed here). This observation led us to study the handling of distinguishing values in `SCA` attacks. We for instance show that by normalizing the `LRA` distinguishing values, the correct key candidate becomes clearly distinguishable even when considering the full vector of instantaneous attack results (as depicted on Figure 1(b)).



(a) without normalization (b) with normalization

Fig. 1. Instantaneous `LRA` scores computed over 10000 traces (scores for the correct key in black)

More generally, our study relies on a well studied problem which is the comparison of the results of two different instantaneous attacks [11,21,38,39,41,43]. For the `LRA`, it will lead to a modification of the candidate selection rule.

2.4 Algorithmic Complexity Improvements Proposals

The other important issue an evaluator faces when performing `SCA`, is the computational complexity of the attack when the number of traces N and their dimension grows to millions. Indeed, the execution time of naive attack imple-

mentations can easily reach several days of processing and this is not compatible with standard evaluation processes[3].

We show in Sections 3, 4 and 5 that the three considered attacks may be rewritten in a *partitioning* fashion that can be exploited to significantly decrease the algorithmic complexity. Roughly speaking, the basic idea is to lower the impact of the heavy computations (*i.e.* Algorithm 1, Step 3) so that its complexity does no longer depend on the traces number N but on the dimension n of the targeted data. To that purpose, we propose to modify the attack first step so that it processes separately the traces with respect to their input value x_i. As a result, the algorithmic complexity of the attacks Step 3 is divided by $\frac{N}{2^n}$ making the algorithmic improvement interesting when $N \gg 2^n$ (which is often the case in practice). The improved generic attack flow is summed up in Algorithm 2.

1. Partitioning of the N observations $\overrightarrow{\ell}_i$ according to the values x_i.
   $\hookrightarrow$ *extract statistics for each of the* 2^n *partitions.*
2. Pre-processing on the predictions $F(x_i, \hat{k})$ for all $\hat{k}$.
   $\hookrightarrow$ *extract statistical data from the key hypothesis and plaintext based on the function* F *and a model and/or an estimation of the leakage statistics.*
3. Processing on the predictions and traces statistics.
   $\hookrightarrow$ *compute a* distinguishing value *for each key hypothesis and instantaneous leakage based on the pre-processed statistics.*
4. Return a key candidate.
   $\hookrightarrow$ *select the key candidate with the best* distinguishing value *over all the instantaneous leakages*

Algorithm 2: Improved Generic Attack Flow

Almost every SCA may be rewritten as a combination of tests on statistics estimated on leakage partitions. Some of them (*e.g.* the `DPA` [18] or the multi-bit `DPA` [24]) were actually originally written as such, whereas the other ones were developed in a partitioning way after their introduction (see *e.g.* [19] for the `CPA`, [10] for the `LRA` and [38] for the `MIA`). To the best of our knowledge, this property has however never been exploited to improve the attacks efficiency.

2.5 Experimental Setup

For each studied SCA, practical experiments were performed on three Micro-Controller Units (MCUs for short) with different CMOS technologies (350, 130 and 90 nanometers processes). The observed processing was that of an AES128 encryption handling one byte at a time. Each attack was performed against 4

[3] In Common Criteria evaluations applied on hardware security devices, all penetration tests (including invasive and non-invasive attacks) have usually to be performed in 3 months, leaving only few weeks for the whole side channel evaluation.

sbox outputs of the first round. Furthermore, to measure the variability of our experiments, we used three different copies for each MCU (called copy 1, 2 and 3 in the sequel). This choice enabled us to perform the `TA` profiling step on one copy and to use the results to attack other ones. Also, it gives more credit to our experimental results as the templates consistency was checked on three different versions of the same MCU.

The side channel observations were obtained by measuring the electromagnetic (EM) radiations emitted by the device. To this aim, several sensors were used, all made of several coils of copper (the diameters of the coils were respectively of 1mm, 500μm and 250μm for the 350, 130 and 90nm MCUs), and were plugged into a low-noise amplifier. To sample measurements, a digital oscilloscope was used with a sampling rate of 1G samples per second for the 350nm MCU and 10G samples per second for the others, whereas the MCUs were running at few dozen of MHz.

We insist on the fact that the temporal acquisition window was set to record the first round of the AES only. This synchronization has been done thanks to *simple electromagnetic analysis* [30]. As the MCU clocks were not stable, we had to resynchronize the measurements. This process is out of the scope of this work, but we emphasize that it is always needed in a practical context and it impacts the measurements noise.

We sum-up the specificities of the three experimental campaigns hereafter:

- **Device A** (3 copies): 90nm CMOS technology with MCU based on a 8-bit 8051 architecture. EM traces composed of 12800 points each after resynchronization. Highest Signal to Noise Ratio (SNR) over the full traces equals to 0.09.
- **Device B** (3 copies): 130nm CMOS technology with MCU based on a 8-bit 8051 architecture. EM traces composed of 16800 points each after resynchronization. Highest SNR equals to 0.6.
- **Device C** (3 copies): 350nm CMOS technology with MCU based on a 8-bit AVR architecture. EM traces composed of 51600 points each after resynchronization. Highest SNR equals to 0.3.

3 Practical Evaluation of Linear Regression Attacks

Linear regression attacks (a.k.a. stochastic attacks) have been introduced by Schindler *et al.* in 2005 [36]. Initially, they were presented with a profiling step and were viewed as an alternative to the template attacks [13]. In [11], the authors have shown how to express the linear regression attacks such that the profiling stage is no longer required. They also argued that the `LRA` can be applied in the same context as the `CPA`, but with weaker assumption on the device behavior. Subsequently, these results of Doget *et al.* have been extended in [10] to apply against masked implementations. In parallel, linear regression attacks have been used to analyse/model the deterministic part of the information leakage for complex circuits [15, 16]. As a matter of fact, all those analyses assume that the side-channel traces are composed of a single leakage point: the issue raised in

Section 2.3 is thus put aside. Moreover, the question of the efficient processing of the attack, when applied against high dimensional leakage traces, is not tackled. The rest of this section aims at dealing with two issues.

3.1 Attack Description

In LRA, the adversary chooses a so-called *basis* of functions[4] $(\mathtt{m}_p)_{1 \leqslant p \leqslant s}$ with the only condition that $\mathtt{m}_1$ is a constant function (usually $\mathtt{m}_1 = 1$). Then, for each x_i and each sub-key hypothesis $\hat{k}$, the prediction $\hat{z}_i = F(x_i, \hat{k})$ is calculated. The basis functions $\mathtt{m}_p$ are then applied to the $\hat{z}_i$ independently, leading to the construction of a $(N \times s)$-matrix $\mathcal{M}_{\hat{k}} \doteq (\mathtt{m}_p(F(x_i, \hat{k}))_{i,p}$. The comparison of this matrix with the set of d-dimensional leakages $(\overrightarrow{\ell}_i)_{i \leqslant N} \hookleftarrow \overrightarrow{\mathbf{L}}$ is done by processing a linear regression of each coordinate of $\overrightarrow{\ell}_i$ in the basis formed by the row elements of $\mathcal{M}_{\hat{k}}$. Namely, a real-valued $(s \times d)$-matrix $\mathcal{B}_{\hat{k}}$ with column vectors $\overrightarrow{\beta}_1, \cdots, \overrightarrow{\beta}_d$ is estimated in order to minimize the error when approximating $\overrightarrow{\ell}_i^{\top}$ by $(\mathtt{m}_1(F(x_i, \hat{k})), \cdots, \mathtt{m}_s(F(x_i, \hat{k}))) \times \mathcal{B}_{\hat{k}}$. The matrix $\mathcal{B}_{\hat{k}}$ is defined such that:

$$\mathcal{B}_{\hat{k}} = \underbrace{\left(\mathcal{M}_{\hat{k}}^{\top} \times \mathcal{M}_{\hat{k}}\right)^{-1} \times \mathcal{M}_{\hat{k}}^{\top}}_{\mathcal{P}_{\hat{k}}} \times \mathcal{L} \ , \tag{1}$$

where $\mathcal{L}$ denotes the $(N \times d)$-matrix with the $\overrightarrow{\ell}_i^{\top}$ as row vectors. In the following, the u^{th} column vector of $\mathcal{L}$ (composed of the u^{th} coordinate of all the $\overrightarrow{\ell}_i$) is denoted by $\overrightarrow{\mathcal{L}}[u]$. Moreover, the $(s \times N)$-matrix $\left(\mathcal{M}_{\hat{k}}^{\top} \times \mathcal{M}_{\hat{k}}\right)^{-1} \times \mathcal{M}_{\hat{k}}^{\top}$, which does not depend on the leakage values, is denoted by $\mathcal{P}_{\hat{k}}$.

To quantify the estimation error, the *goodness of fit model* is used and the *correlation coefficient of determination* $\mathcal{R}^2$ is computed for each u. The latter is defined by $\mathcal{R}^2 = 1 - \text{SSR/SST}$, where SSR and SST respectively denote the *residual sum of squares* (deduced from $\mathcal{B}_{\hat{k}}$) and the *total sum of squares*[5] (deduced from $\mathcal{L}$). We give in Algorithm 3 the pseudo-code corresponding to a classical LRA attack processing.

3.2 On the LRA Effectiveness

Let us focus on the best candidate selection step in a classical LRA. Each sub-key hypothesis $\hat{k}$ is first associated with a score which is the greatest *instantaneous coefficient of determination* when testing it for all temporal coordinates u. It is denoted by $\max_u \mathcal{R}[\hat{k}][u]$ in Alg. 3. The second phase of the selection consists in the processing of the maximum $\text{argmax}_{\hat{k}}(\max_u \mathcal{R}[\hat{k}][u])$. The purpose of the latter step is to identify the candidate that maximises the greatest instantaneous coefficient. Implicitly, such a classical approach by *total maximisation of the distinguisher value* assumes that the most likely candidate corresponds to the

[4] The basis choice and its impact are not a trivial matter, see [10] for a detailed study.
[5] for their exact definitions, see their construction in Alg. 3

Algorithm 3: `LRA` - Linear Regression Analysis

Input : a set of d-dimensional leakages $(\overrightarrow{\ell}_i)_{i \leqslant N}$ and the corresponding plaintexts $(x_i)_{i \leqslant N}$, a set of model functions $(\mathtt{m}_p)_{p \leqslant s}$

Output: A candidate sub-key $\hat{k}$

/* Processing of the leakage Total Sum of Squares ($\overrightarrow{\mathrm{SST}}$) */

for $i = 0$ **to** $N - 1$ **do**

  $\mu_{\overrightarrow{\mathbf{L}}} = \mu_{\overrightarrow{\mathbf{L}}} + \overrightarrow{\ell}_i$

  $\sigma_{\overrightarrow{\mathbf{L}}} = \sigma_{\overrightarrow{\mathbf{L}}} + \overrightarrow{\ell}_i^2$

$\overrightarrow{\mathrm{SST}} = \sigma_{\overrightarrow{\mathbf{L}}} - 1/N \cdot \mu^2_{\overrightarrow{\mathbf{L}}}$

/* Processing of the 2^n predictions matrices $\mathcal{M}_{\hat{k}}$ and $\mathcal{P}_{\hat{k}}$ */

for $\hat{k} = 0$ **to** $2^n - 1$ **do**

  /* Construct the matrix $\mathcal{M}_{\hat{k}}$ and $\mathcal{P}_{\hat{k}}$ */

  for $p = 1$ **to** s **do**

    for $i = 0$ **to** $N - 1$ **do**

      $\mathcal{M}_{\hat{k}}[i][p] \leftarrow \mathtt{m}_p[F(x_i, \hat{k})]$

  $\mathcal{P}_{\hat{k}} = (\mathcal{M}_{\hat{k}}^\top \times \mathcal{M}_{\hat{k}})^{-1} \times \mathcal{M}_{\hat{k}}^\top$

for $\hat{k} = 0$ **to** $2^n - 1$ **do**

  /* Test hyp. $\hat{k}$ for all leakage coordinates */

  for $u = 0$ **to** $d - 1$ **do**

    /* *Instantaneous attack (at time u)* */

    $\overrightarrow{\beta} = \mathcal{P}_{\hat{k}} \times \overrightarrow{\mathcal{L}}[u]$

    /* Compute an estimator $\overrightarrow{E}$ of $\overrightarrow{\mathcal{L}}[u] = (\overrightarrow{\ell}_0[u], \cdots, \overrightarrow{\ell}_{N-1}[u])^\top$ */

    $\overrightarrow{E} = \mathcal{M}_{\hat{k}} \times \overrightarrow{\beta}$

    /* *Compute the estimation error (*i.e. *the SSR)* */

    $\mathrm{SSR} = 0$

    for $i = 0$ **to** $N - 1$ **do**

      $\mathrm{SSR} = \mathrm{SSR} + (\overrightarrow{E}[u] - \overrightarrow{\ell}_i[u])^2$

    /* Compute the coefficient of determination */

    $\mathcal{R}[\hat{k}][u] = 1 - \mathrm{SSR}/\overrightarrow{\mathrm{SST}}[u]$

/* Most likely candidate selections */

candidate $= \mathrm{argmax}_{\hat{k}}(\max_u \mathcal{R}[\hat{k}][u])$

return candidate

greatest value taken by the distinguisher not only over all sub-key hypotheses but also over all the leakage times. This assumption relies on another one, often done in the embedded security community, which states that the value of a distinguisher computed between wrong hypotheses (*i.e.* computed for a wrong sub-key value or a wrong time) and the leakage values tends toward its minimum value (often 0) when the sample size N increases (see *e.g.* [21]). However, as already noticed in several papers (*e.g.* by Messerges in [24], Brier *et al.* in [8] or by Whitnal *et al.* in [45]), both assumptions are often not verified in practice, where the adversary must for instance deal with the *ghost peaks* phenomenon. The situation is even worst for the `LRA` attacks since the vector of coefficients $\overrightarrow{\beta}$ (and thus the set of predictions) depends not only on $\hat{k}$ but also on the attack time [6] u. The strength of the `LRA`, namely its ability to adapt to the instantaneous leakage, is also its weakness as it makes it difficult to compare the different instantaneous attacks results.

To illustrate the issue raised in the previous paragraph, we experimented a `LRA` against an AES sbox processing running on Device B (see Section 2.5). The full leakage traces were composed of 16800 points. We performed the attack on the full trace length and, for each time coordinate, we recorded the scores of all the 256 key-candidates after $N = 1000$ observations. For clarity reasons, we present in Figure 2 the results only for a temporal window of size 250 points where the targeted variable was known to occur. In the top of the figure, the rank of the correct key is plotted and it can yet be observed that it is 0 for few times. In the second trace of Figure 2, the instantaneous maximum scores comprised in $[0.9982, 0.999]$ are plotted[7]: it may be checked that the maximum among those scores corresponds to a time ($t = 238$) when the correct key is not ranked first. This explains why the total maximisation approach fails in returning the correct key candidate in this case.

To build a better rule than the total maximisation test, we respectively plotted in the third and fourth traces of Figure 2 the mean (plain green trace) and the variance (plain red trace) of the instantaneous scores (*i.e.* the values $\mu(u) = 2^{-8} \sum_{\hat{k}} \mathcal{R}[\hat{k}][u]$ and $\sigma(u) = 2^{-8} \sum_{\hat{k}} (\mathcal{R}[\hat{k}][u] - \mu(u))^2$ with u denoting the time coordinate in abscissa). For each time, we also plotted in black dashed line, the maximum score $\max(u) = \max_{\hat{k}}(\mathcal{R}[\hat{k}][u])$. It may be observed that the correct key is ranked first at the time u when the distance $\max(u) - \mu(u)$ is large and $\sigma(u)$ is small. The third (red) trace and the fourth (gray) trace aim at supporting this claim. Eventually, they suggest us the following pre-processing before comparing the instantaneous attack results: for each leakage coordinate, center the maximum of the coefficients of determination and divide it by their standard deviation. The resulting scoring is plotted in the fifth (magenta) trace, where it can been checked that the maximum is indeed achieved for the correct key.

[6] As recalled in Section 4, it is not the case for the `CPA` as the predictions $\mathtt{m}(F(x, \hat{k}))$ are the same for each instantaneous attack

[7] For visibility purpose, we chose to not plot the scores lower than 0.9982

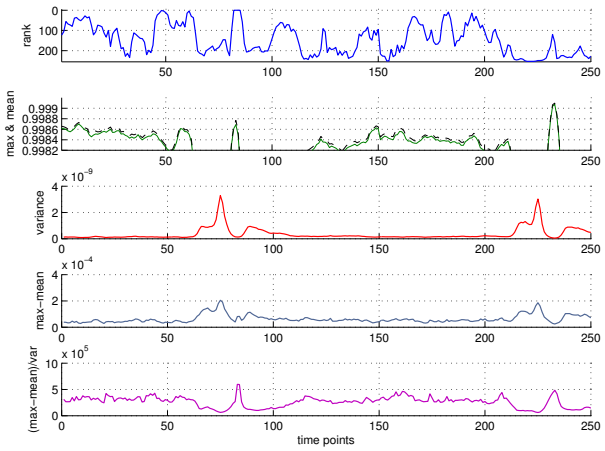


Fig. 2. LRA on Device B (over 1000 traces): Scores Statistics

As a conclusion, and in the light of our analysis, we propose to replace the candidate selection step of the LRA by the following ones[8]:

for $u = 0$ **to** $d - 1$ **do**

$\quad \text{attackRes}[u] = \{\text{argmax}_{\hat{k}}(\mathcal{R}[\hat{k}][u]), \frac{\max_{\hat{k}}\left(\mathcal{R}[\hat{k}][u]\right) - \mathbb{E}_{\hat{k}}\left[\mathcal{R}[\hat{k}][u]\right]}{\sigma_{\hat{k}}\left[\mathcal{R}[\hat{k}][u]\right]}\}$

$\text{candidate} = \text{arg1max2}(\text{attackRes})$

In Section 3.4, our scores pre-processing technique is applied to attack samples of Device A and Device C in order to test whether our observations, about (1) the ineffectiveness of the classical LRA and (2) the soundness of the new pre-processing, stay valid for other devices than Device B.

3.3 On the LRA Efficiency

The construction of the prediction matrices in Alg. 3 implies, for each $\hat{k}$, the processing of 3 products of matrices with one dimension equal to s (number of basis functions) and the second dimension equal to N (number of leakage traces). The processing of the instantaneous attacks also requires two such matrix products for each pair $(\hat{k}, u)$ with $u \leqslant d$. This makes the application of a linear regression attack as depicted in Alg. 3 difficult to perform (and even impossible)

[8] where arg1max2 is a function returning the first coordinate of the maximum of an array of 2-dimensional elements, the maximisation being computed with respect to the second coordinate of the array elements.

when the number N of leakage traces and/or the number d of attack times are large. Fortunately this complexity can be significantly reduced. It can indeed be easily shown (see [10]) that the processing of the vectors $\overrightarrow{\beta}$ is unchanged if performed for the set of *averaged leakages* $(\frac{1}{\#\{i:x_i=x\}} \sum_{i,x_i=x} \overrightarrow{\ell}_i)_{x\in\mathbb{F}_2^n}$ instead of $(\overrightarrow{\ell}_i)_i$. Actually, this amounts to change the definition of the matrices $\mathcal{M}_{\hat{k}}$ and $\mathcal{L}$ in (1) such that $\mathcal{M}_{\hat{k}} \doteq (\mathtt{m}_p(F(x,\hat{k}))_{x\in\mathbb{F}_2^n, p\leqslant s}$ and $\mathcal{L}$ is a $(2^n \times d)$-matrix whose x^{th} row vector $\overrightarrow{\mathcal{L}^\top}[x]$ equals $\frac{1}{\#\{x_i=x\}} \sum_{i,x_i=x} \overrightarrow{\ell}_i$. This improvement essentially lets the first 9 steps of Alg. 3 unchanged except the loop 7-8 which is now computed over $x \in \mathbb{F}_2^n$ instead of over $i \in [0; N-1]$. Then, before Step 10, the following processing is done to compute the elements of the matrix $\mathcal{L}$:

```
for i = 0 to N − 1 do
    L⃗^⊤[x_i] = L⃗^⊤[x_i] + ℓ⃗_i
    count[x_i] = count[x_i] + 1
for x = 0 to 2^n − 1 do
    L⃗^⊤[x] = L⃗^⊤[x]/count[x]
```

Eventually, Steps 10-17 are replaced by the following ones where we recall that $\overrightarrow{\mathcal{L}}[u]$ denotes the u^{th} column vector of $\mathcal{L}$.

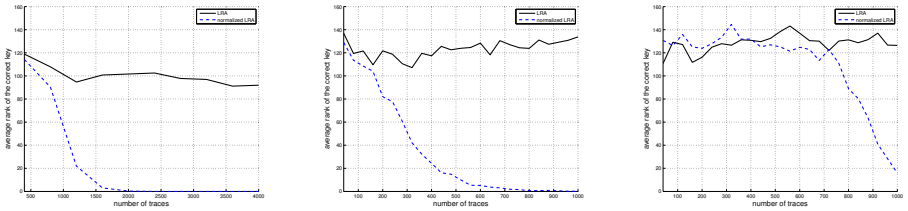
```
for k̂ = 0 to 2^n − 1 do
    /* Test hypothesis k̂ for all leakage coordinates                    */
    for u = 0 to d − 1 do
        /* Instantaneous attack (at time u)                             */
        β⃗ = P_k̂ × L⃗[u]
        /* Compute an estimator Ê of L⃗[u] = (L[0][u], ··· , L[2^n − 1][u])^⊤ */
        Ê = M_k̂ × β⃗
        /* Compute the estimation error (i.e. the residual sum of squares) */
        SSR = 0
        for x = 0 to 2^n − 1 do
            SSR = SSR + (Ê[x] − L[x][u])^2
        /* Compute the coefficient of determination                     */
        R[k̂][u] = 1 − SSR/SST⃗[u]
```

The efficiency improvements proposed here for the `LRA` attack allows for a significant time/memory gain. First, it replaces the $(N \times s)$-matrix products at Step 13 by $(2^n \times s)$-matrix products. More globally, the complexity is reduced from $O(s \times d \times N)$ to $O(s \times d \times 2^n)$. A detailed counting of the elementary operations for each attack is given in Appendix A. If the $\overrightarrow{\beta}$ values are not needed (*i.e.* the weights of the linear regression is of no interest to the attacker),

the matrix products $\mathcal{M}_{\hat{k}} \times \mathcal{P}_{\hat{k}}$ can also be pre-processed. This enables to save one matrices product per loop iteration (over $\hat{k}$ and u).

3.4 Experiments

We experimented the classical and improved `LRA` against against three copies of Devices A, B and C (see Section 2.5). The attacks target four bytes of the AES state after the first `SubBytes` operation and they are applied on the full side channel traces. Each attack has been performed 10 times against each of the three copies. The average rank over the four correct sub-keys is plotted in Figure 3 for each device. We recall that the rank of a sub-key k is here defined as the position of $\max_u \mathcal{R}[k][u]$ in the vector $(\max_u \mathcal{R}[\hat{k}][u])_{\hat{k}}$ after sorting (see Section 2.2 for a discussion about this choice). The experiments reported in Figure 3(a)-(c) are done with a *linear basis* (*i.e.* the functions $\mathtt{m}_i$ were chosen such that $\mathtt{m}_0$ is constant equal to 1 and $\mathtt{m}_i$, with $i \leqslant 8$, returns the i^{th} bit of its inputs). It may be observed that the classical attack always failed whereas the improved one succeeded with less than 2500 observations (and even less than 800 for Device B). After noticing that the attack was less efficient on Device A (90nm) than that on Device B (130nm), we chose to perform a new attack campaign against the former, but this time with a quadratic basis (*i.e.* the functions corresponding to the product of two input bits were added to the linear basis). Results are given in Appendix C (Fig. 9) and it can be noticed that we did not witness any effectiveness improvement.

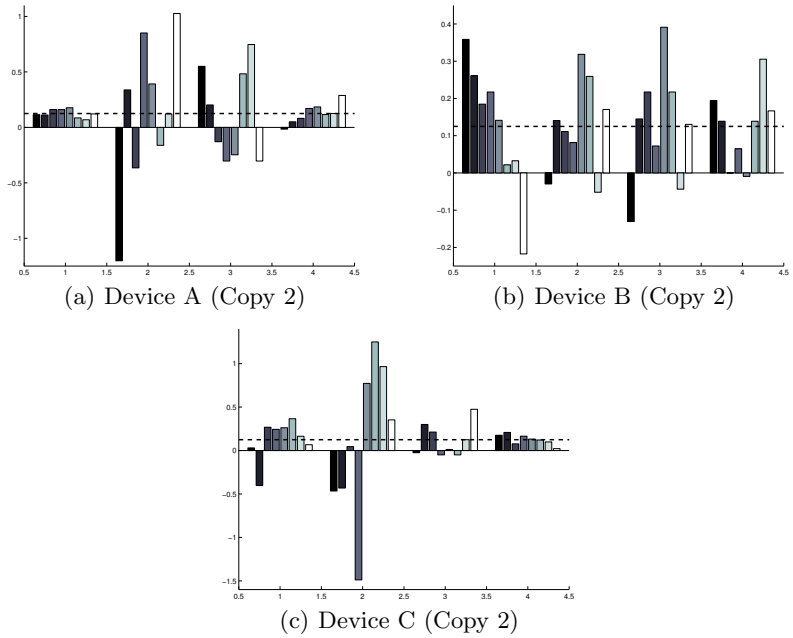


(a) `LRA` on Device A (b) `LRA` on Device B (c) `LRA` on Device C

Fig. 3. `LRA` campaign – Rank evolution *versus* number of observations

A `LRA` as described in Alg. 3 does not only return a candidate but also, for the time when the attack succeeds, a regression of the leakage as a linear combination of the basis functions. We chose to exploit this particular feature and to extract, for the linear basis, the weights of each bit returned by the `LRA` for each of the four targeted sbox outputs (*i.e.* these weights are the coordinates of the four vectors $\vec{\beta}$ related to the correct targeted sub-keys and the leakage time for which the attack succeeds more efficiently). The weights are plotted (after normalization) in Figure 4 for the second copy of Device A, B and C. For

comparison, we also plotted in blue dashed line the value corresponding to the normalized weight (*i.e.* $\frac{1}{8}$) of a bit in a Hamming Weight model. For Devices A and C, it may be noticed that the first and fourth sbox output manipulations leak closely to the Hamming weight model (normalized weights are more or less balanced). This is not the case of the weights of the second and third sbox outputs, which are very unbalanced. For Device B, one can observe that at least two bits leak significantly less than the others.



(a) Device A (Copy 2)

(b) Device B (Copy 2)

(c) Device C (Copy 2)

Fig. 4. `LRA` weights for each of the 3 devices

4 Practical Evaluation of Correlation Power Analysis

Correlation Power Analysis was proposed by Brier *et al.* [8] in 2004. The idea is to use Pearson's correlation coefficient as a distinguisher instead of the difference of means test originally proposed by Kocher *et al.* [18]. `CPA` is certainly nowadays the most popular side channel technique as it proved itself (combined with classical leakage models like the Hamming weight and Hamming distance) very efficient on an overwhelming majority of hardware architectures[9]. The naturally good effectiveness of `CPA` was already mentioned in Brier *et al.* paper as the use

[9] which amount to say that, for classical embedded technologies, the HW and HD models are close to real leakage functions up to a linear factor.

of CPA allows to reduce the presence of *ghost peaks* compared to classical DPA. Subsequent works on CPA effectiveness essentially consisted in applying signal filtering techniques or pre-processing on the leakage traces (see *e.g.* [6, 26] for recent works on this subject). We do not propose amelioration here on this topic. About the attack efficiency, very few works dealt with this problem at the algorithmic level. In fact, published works on this subject have essentially addressed the issue of processing the CPA on multicore CPU or graphic processor unit (see *e.g.* [4]). We show in this section that re-writing the CPA as in a partitioning way allows for a significant efficiency gain. This re-writing has already been proposed in [19] but, to the best of our knowledge, it has never been used for efficiency improvement purpose[10].

4.1 Attack Description

In a correlation power analysis attack, the adversary computes, for each plaintext x_i and each sub-key hypothesis $\hat{k}$, an hypotheses tuple $\hat{z}_i$ such that $\hat{z}_i = F(x_i, \hat{k})$. Then, he chooses a so-called *model function* $\mathtt{m}(\cdot)$ from $\mathbb{F}_2^m$ into $\mathbb{R}$ and applies it to each hypothesis which leads to the construction of a new set of so-called *predictions* $(h_i)_{i \leqslant N} \hookleftarrow \overrightarrow{H} = \mathtt{m}(F(X, \hat{k}))$. This latter set of hypotheses is then compared to the set of leakages $(\overrightarrow{\ell}_i)_{i \leqslant N} \hookleftarrow \overrightarrow{\mathbf{L}}$ to assess on the likelihood of $\hat{k}$ as a candidate for one of the sub-keys k. The core principle of a CPA is to make the comparison by estimating the linear correlation between $\overrightarrow{H}$ and each coordinate of $\overrightarrow{\mathbf{L}}$ independently. We give in Alg. 4 the pseudo-code corresponding to the CPA attack discussed previously. A brief description of it may also be found in Appendix B.

4.2 On the CPA Effectiveness

Based on the discussion conducted in Section 3.2 and the success of the experimental validation of the new selection strategy, this may seem natural to try it in the CPA context. However, the fact that the same constant model $\mathtt{m}(\cdot)$ is used for each instantaneous attack (which was not the case for the LRA) implies that two correlation coefficients corresponding to two different coordinates of the vectors $\overrightarrow{\ell}_i$ are directly comparable (since they are quantifying the linear correlation between the leakages and a common model). Therefore the most likely hypothesis corresponds to the greatest value of the correlation coefficient/score when the set of hypotheses $(\hat{z}_i)_i$ is tested with respect to each coordinate u of the vectors $\overrightarrow{\ell}_i$. We actually checked that the normalizing strategy applied in Section 3.2 indeed does not enable to improve the CPA efficiency.

[10] One may note that, in [38], CPA is listed as a *comparison* distinguisher by opposition to the *partition* distinguishers like the MIA.

Algorithm 4: `CPA` - Correlation Power Analysis

Input : a set of d-dimensional leakages $(\overrightarrow{\ell}_i)_{i \leqslant N}$ and the corresponding plaintexts $(x_i)_{i \leqslant N}$, a model function $\mathtt{m}(\cdot)$
Output: A candidate sub-key $\hat{k}$

`/* Leakage mean and variance processing */`
for $i = 0$ **to** $N - 1$ **do**
  $\mu_{\overrightarrow{\mathbf{L}}} = \mu_{\overrightarrow{\mathbf{L}}} + \overrightarrow{\ell}_i$
  $\sigma_{\overrightarrow{\mathbf{L}}} = \sigma_{\overrightarrow{\mathbf{L}}} + \overrightarrow{\ell}_i^2$
$\mu_{\overrightarrow{\mathbf{L}}} = 1/N \cdot \mu_{\overrightarrow{\mathbf{L}}}$
$\mathrm{var}_{\overrightarrow{\mathbf{L}}} = 1/N \cdot \sigma_{\overrightarrow{\mathbf{L}}} - \mu_{\overrightarrow{\mathbf{L}}}^2$

`/* Hypotheses mean and variance processing */`
for $\hat{k} = 0$ **to** $2^n - 1$ **do**
  for $i = 0$ **to** $N - 1$ **do**
    $z \leftarrow \mathtt{m}(F(x_i, \hat{k}))$
    $\mu_{\hat{k}} = \mu_{\hat{k}} + z$
    $\sigma_{\hat{k}} = \sigma_{\hat{k}} + z^2$
  $\mu_{\hat{k}} = \mu_{\hat{k}}/N$
  $\mathrm{var}_{\hat{k}} = \sigma_{\hat{k}}/N - \mu_{\hat{k}}^2$

`/* Correlations processing */`
for $\hat{k} = 0$ **to** $2^n - 1$ **do**
  `/* Test hypothesis` $\hat{k}$ `for all leakage coordinates */`
  for $u = 0$ **to** $d - 1$ **do**
    `/* Instantaneous attack (at time` u`) */`
    $\mathrm{cov} = 0$
    for $i = 0$ **to** $N - 1$ **do**
      $\mathrm{cov} = \mathrm{cov} + \mathtt{m}(F(x_i, \hat{k})) \times \overrightarrow{\ell}_i[u]$
    $\mathrm{cor}[\hat{k}][u] = (1/N \times \mathrm{cov} - \mu_{\hat{k}} \times \mu_{\overrightarrow{\mathbf{L}}}[u]) / \sqrt{\mathrm{var}_{\hat{k}} \times \mathrm{var}_{\overrightarrow{\mathbf{L}}}[u]}$

`/* Most likely candidate selections */`
$\mathrm{candidate} = \mathrm{argmax}_{\hat{k}}(\max_u \mathrm{cor}[\hat{k}][u])$
return candidate

4.3 On the CPA Efficiency

The law of total expectation implies:

$$\mathrm{cov}(X,Y) = \mathbb{E}\left[\mathbb{E}\left[\big(X - \mathbb{E}[X]\big)\big(Y - \mathbb{E}[Y]\big) \mid X\right]\right] \ ,$$

where the conditional means are viewed as random variables that functionally depend on X. Due to the linearity of the mean, this latter equation is also equivalent to $\mathrm{cov}(X,Y) = \mathbb{E}\left[(X - \mathbb{E}[X])\,\mathbb{E}\big[(Y - \mathbb{E}[Y]) \mid X\big]\right]$. When applied in the context of the correlation coefficients computation in Algorithm 4, this equation leads to a significant efficiency improvement. Prior to Steps 13-18 in Alg. 4 (instantaneous attacks), we propose to insert the following pre-processing in order (1) to partition the leakage traces $\overrightarrow{\ell}_i$ according to the corresponding input values x_i and (2) to compute the mean for each partition. We denoted by $\mathcal{A}_{\overrightarrow{\mathbf{L}}}$ the matrix whose x^{th} column vector equals the mean of the elements in $\{\overrightarrow{\ell}_i; x_i = x\}$.

```
/* Conditional Leakage Averaging for each possible x          */
Erase A_L
for i = 0 to N − 1 do
    A_L[x_i] = A_L[x_i] + ℓ_i
    count[x_i] = count[x_i] + 1
for x = 0 to 2^n do
    A_L[x] = A_L[x]/count[x]
```

Eventually, Steps 18-28 in Alg. 4 are replaced by the following ones where, for each of the 2^n instantaneous attacks, the proposed efficiency improvement essentially allows for the replacement of $N-1$ additions and multiplications by 2^n ones. Taking the leakage conditional averaging step into account, this gives a replacement of $d \times 2^n \times (2N+6)$ elementary operations[11] by $2N + 2^n$ plus $d \times 2^n \times (2^n + 6)$. A detailed counting of the elementary operations for each attack is given in Appendix A.

4.4 Experiments

We experimented the classical and normalized CPA against the same targets as those used in Section 3.4. For both attacks, we used the classical Hamming weight function for $\mathtt{m}(\cdot)$. Similarly as in Section 3.4, average ranks have been computed. They are plotted in Figure 5 for each device. It may first be noticed that the CPA is very efficient against Devices B and C as it respectively succeeds, on average, in less than 400 and 300 traces. The same attack against Device A

[11] *i.e.* on n-bit words or real values

```
/* Correlations processing                                          */
for k̂ = 0 to 2^n − 1 do
    /* Test hypothesis k̂ for all leakage coordinates               */
    for u = 0 to d − 1 do
        /* Instantaneous attack (at time u)                         */
        cov = 0
        for x = 0 to 2^n − 1 do
            cov = cov + m(F(x, k̂)) × A_L⃗[x][u]
        cor[k̂][u] = (1/N × cov − μ_k̂ × μ_L⃗[u])/√(var_k̂ × var_L⃗[u])
```

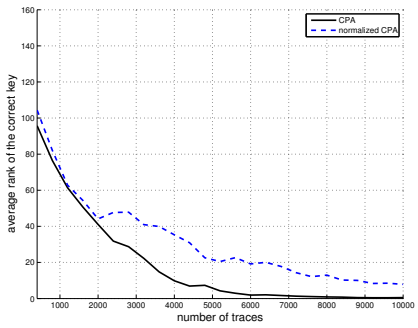
is much less efficient (it requires around 9000 traces to succeed). This is a consequence of the low measurements SNR and the fact that the leaking information is not always well captured by the Hamming weight model (see Fig. 4(a)). Also, and as expected from our analysis in Section 4.2, the normalized CPA is, on average, not better than the classical CPA. The relative ineffectiveness of the former with respect to the CPA seems to be all the more important than the SNR is small. Confirming this in other contexts could be of interest for further research. Eventually, the CPA on device A is less efficient than the LRA attack on the same device.

In the following two figures, we detail the evolution of the ranks of the four different targeted sub-keys in function of the number N of observations. For each of them we only consider the rank at the time coordinate where the CPA finally succeeds.

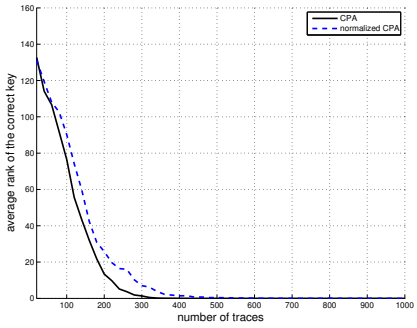
Figure 6(a) must be studied together with the LRA experiments reported in Figure 4(a). It may indeed by observed that the CPA attack is particularly efficient against the manipulations that leak close to the Hamming weight (bytes 0 and 3) and lose efficiency for the two other bytes (1 and 2). Surprisingly (at a first glance), this phenomenon cannot be observed for Figures 6(b) and 4(c). Actually, this is a consequence of the fact that CPA and the LRA do not always succeed for the same time coordinate. The unbalancedness observation made for 4(c) is therefore not relevant to interpret the results in 6(b). This is a consequence of the fact that the CPA sub-keys ranking evolution and the leakage characterisation correspond to different leakage times (in other words, the best leakage time for the LRA and the CPA do not equal for Device C).

5 Practical Evaluation of Template Attacks

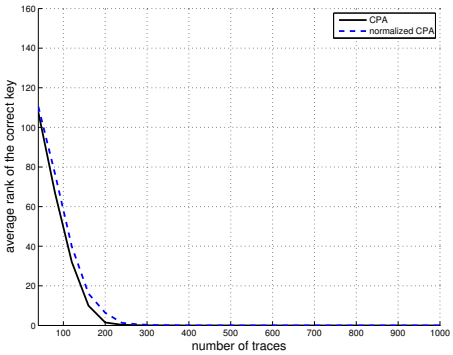
Template attacks have been introduced in 2002 by Chari *et al.* [9]. Subsequent works have then been published which either show how to apply them against particular implementations (*e.g.* AES, RSA or ECDSA) or propose efficiency/effectiveness improvements [1, 3, 31, 33]. In [31], the authors reduce the complexity of template attacks by first applying a pre-processing on the mea-



(a) `CPA` on Device A

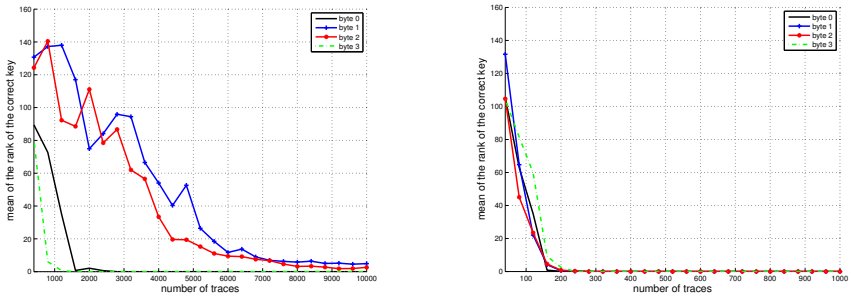


(b) `CPA` on Device B



(c) `CPA` on Device C

Fig. 5. `CPA` campaign – Rank evolution *versus* number of observations



(a) `CPA` on Device A (b) `CPA` on Device C

Fig. 6. `CPA` campaign – Rank evolution *versus* number of observations for the four sbox outputs

surements (to go from time domain to frequency domain) and then by applying dimension reduction techniques (*e.g.* PCA). The latter idea is also followed in [1] and [3]. In all those papers, the improvement of the template attacks efficiency is not studied at the algorithmic level. Moreover, the reported template attack experiments involve the same device for the profiling and matching phases of the attacks, which strongly reduces the practical significance of the argumentations. Indeed, as the profiling phase requires a full access to the device (and in particular the ability to chose the secret parameter), the latter experiments do not fit with the large majority of real attack/evaluation contexts where the adversary has no (or very few) control on the target device. In a more realistic attacker model the profiling phase is conducted on a different device. For such a model, we have the following well known question about the efficiency of template attack: how sound/relevant is a profiling done on a device A when attacking another device B ?[12] The first work, and to the best of our knowledge, the single one reporting on template attacks in such context is due to Renauld *et al.* [33]. On the latter article, the two devices used for the experiments are test chips implementing an AES s-box and made in 65-nanometer CMOS technology.

The results presented in the rest of this section improve the state-of-the-art recalled previously on two points. First, the efficiency improvement is done at the algorithmic level. It can hence be combined with the previous improvements which essentially correspond to measurements traces pre-processing. Secondly, the reported experiments concern a full AES implementation running on 3 different samples of 3 different technologies. This allowed us to complete the analyses done in [33] and to draw, for the first time, conclusions about the template attack efficiency for realisitic scenarios.

[12] This question is sometimes also related to the statistical problem of pdf estimations robustness [29].

5.1 Attack Description

A template attack (TA for short) assumes that a preliminary profiling step has been performed on an open copy of the targeted device. During this phase, the adversary has measured N' leakage traces $\overrightarrow{\ell}_i'$ for which he knows exactly the values taken by the corresponding sensitive value Z (which also implies that he knows the corresponding sub-key k). Those leakages have then been used to compute estimations $f_z(\cdot)$ of the *probability density function* of $(\overrightarrow{\mathbf{L}} \mid Z = z)$ for all possible z (which imposes $N' \gg 2^m$). The pdf estimations $f_z(\cdot)$ will play in a template attack, a similar role as the model functions in a CPA or LRA.

Once the adversary has the set of pdf estimations $(f_z(\cdot))_{z \in \mathbb{F}_2^m}$ in hand, a TA against the set of traces $(\overrightarrow{\ell}_i)_{i \leqslant N}$ (for which the secrets are unknown) follows essentially the same outlines as the CPA or the LRA: the hypothesis $\hat{k}$ is tested by first computing the predictions $\hat{z}_i = F(x_i, \hat{k})$ and then by calculating the product $\prod_{i \leqslant N} f_{\hat{z}_i}(\overrightarrow{\ell}_i)$. Usually, the pdf of the variables $(\overrightarrow{\mathbf{L}} \mid Z = z)$ is estimated by a multivariate normal law, which implies that f_z can be developed s.t.:

$$f_z(\overrightarrow{\ell}_i) = \frac{1}{(2\pi)^d \det(\Sigma_z)} \exp\Big(-\frac{1}{2} \big(\overrightarrow{\ell}_i - \overrightarrow{\mu}_z\big)^\top \Sigma_z^{-1} \big(\overrightarrow{\ell}_i - \overrightarrow{\mu}_z\big)\Big) \ , \qquad (2)$$

where Σ_z denotes the $(d \times d)$-matrix of covariances of $\overrightarrow{\mathbf{L}} \mid Z = z$ and where the (d)-dimensional vector $\overrightarrow{\mu}_z$ denotes its mean.

To minimize approximation errors induced by the processing of the product of exponential values, one usually prefers, in practice, a log-maximum likelihood processing to the classical maximum likelihood[13]. Together with (2), this leads to the following computation to test the hypothesis $\hat{k}$:

$$\mathcal{ML}[\hat{k}] = -\sum_{i \leqslant N} \big(\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i}\big)^\top \Sigma_{\hat{z}_i}^{-1} \big(\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i}\big) - \sum_{i \leqslant N} \log\big((2\pi)^{d+1} \det\big(\Sigma_{\hat{z}_i}\big)\big) \ . \ (3)$$

We give in Alg. 5 the pseudo-code corresponding to the TA attack discussed previously.

[13] The two processes discriminate equivalently.

Algorithm 5: `TA` - Template Attacks

Input : a set of d-dimensional leakages $(\overrightarrow{\ell}_i)_{i \leqslant N}$ and the corresponding plaintexts $(x_i)_{i \leqslant N}$, a set of pdf estimations $(\overrightarrow{\mu}_z, \Sigma_z)_{z \in \mathbb{F}_2^m}$

Output: A candidate sub-key $\hat{k}$

`/* Pre-Processing of the` 2^m `log-determinants` $\log(2\pi^{d+1}\Sigma_z)$ `and inverse-matrices` Σ_z^{-1} `*/`

for $z = 0$ **to** $2^m - 1$ **do**
  $\text{logDet}_z = \log(2\pi^{d+1}\Sigma_z)$
  $\text{invCov}_z = \Sigma_z^{-1}$

`/* Instantaneous TA attacks Processing */`

for $\hat{k} = 0$ **to** $2^n - 1$ **do**
  `/* Test hyp.` $\hat{k}$ `*/`
  $\mathcal{ML}[\hat{k}] = 0$
  for $i = 0$ **to** $N - 1$ **do**
    $\hat{z} = F(x_i, \hat{k})$
    $\mathcal{ML}[\hat{k}] = \mathcal{ML}[\hat{k}] - (\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}})^\top \times \text{invCov}_{\hat{z}} \times (\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}}) - \text{logDet}_{\hat{z}}$

`/* Most likely candidate selections */`

candidate $= \text{argmax}_{\hat{k}}(\max \mathcal{ML}[\hat{k}])$

return candidate

5.2 On the `TA` Effectiveness

The idea developed in previous sections to improve the selection of the best candidate among the results of several instantaneous attacks is not relevant here. Indeed, for both the profiling and attack phases, a template attack exploits, by nature, all the leakage coordinates of the $\overrightarrow{\ell}_i$ simultaneously. There is consequently no need to compare the results of several (different) instantaneous attacks.

5.3 On the `TA` Efficiency

Applying the same idea as for the `CPA` and the `LRA`, we propose hereafter an alternative writing of $\mathcal{ML}[\hat{k}]$ that leads to a much faster attack processing. For such a purpose, we focus on the term $(\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i})^\top \Sigma_{\hat{z}_i}^{-1} (\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i})$ in (3).

After denoting by $\mathcal{L}_i$ each $(d \times d)$-matrix $(\overrightarrow{\ell}_i[u]\overrightarrow{\ell}_i[u'])_{u,u'}$, we get the following rewriting of the latter term:

$$\sum_{u,u'} \left(\mathcal{L}_i[u][u'] \times \Sigma_{\hat{z}_i}^{-1}[u][u']\right) - \overrightarrow{\mu}_{\hat{z}_i}^\top \times \left(\Sigma_{\hat{z}_i}^{-1} + \Sigma_{\hat{z}_i}^{-1^\top}\right) \times \overrightarrow{\ell}_i + \overrightarrow{\mu}_{\hat{z}_i}^\top \times \Sigma_{\hat{z}_i}^{-1} \times \overrightarrow{\mu}_{\hat{z}_i} \ .$$

After recalling that $\hat{z}_i$ equals $F(x_i, \hat{k})$ and after denoting $F(x, \hat{k})$ by $\hat{z}$ and $\#\{i, x_i = x\}$ by N_x, we deduce that the sum $\sum_i (\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i})^\top \Sigma_{\hat{z}_i}^{-1} (\overrightarrow{\ell}_i - \overrightarrow{\mu}_{\hat{z}_i})$ may be rewritten:

$$\sum_{x\in\mathbb{F}_2^n}\left(\sum_{u,u'}\Big(\sum_{i,x_i=x}\mathcal{L}_i\Big)[u][u']\times\Sigma_{\hat{z}}^{-1}[u][u']\right.$$
$$\left.-\overrightarrow{\mu}_{\hat{z}}^{\top}\times\left(\Sigma_{\hat{z}}^{-1}+\Sigma_{\hat{z}}^{-1^{\top}}\right)\times\Big(\sum_{i,x_i=x}\overrightarrow{\ell}_i\Big)+N_x\times\overrightarrow{\mu}_z^{\top}\times\Sigma_z^{-1}\times\overrightarrow{\mu}_z\right)\ .$$

As a consequence, if the 2^n possible sums $\sum_{i,x_i=x}\mathcal{L}_i$ and $\sum_{i,x_i=x}\overrightarrow{\ell}_i$ have been precomputed, then the complexity of evaluating (3) for each $\hat{k}$ goes from $O(Nd^2)$ to $O(2^nd^2)$. Algorithm 6 describes the improved `TA` attack.

Algorithm 6: `TA` - Template Attacks (Improved Version)

Input : a set of N leakages $(\overrightarrow{\ell}_i)_i$ and the corresponding plaintexts $(x_i)_i$, a set of pdf estimations $(\overrightarrow{\mu}_z,\Sigma_z)_{z\in\mathbb{F}_2^m}$
Output: A candidate subkey $\hat{k}$

`/* Pre-Processing of the predictions data */`
1 **for** $z=0$ **to** 2^m-1 **do**
2 $\quad\log\mathrm{Det}_z=\log(2\pi^{d+1}\Sigma_z)$; $\mathrm{invCov}_z=\Sigma_z^{-1}$; $\mathrm{meanCov}_z=\overrightarrow{\mu}_z^{\top}\times\mathrm{invCov}_z\times\overrightarrow{\mu}_z$; $\mathrm{sumMeanCov}_z=\overrightarrow{\mu}_z^{\top}\times(\Sigma_z^{-1}+\Sigma_z^{-1^{\top}})$

`/* Pre-Processing of the leakage data`[14] $\overline{\ell_x}=\sum_{i,x_i=x}\overrightarrow{\ell}_i$, $\overline{\mathcal{L}}_x=\sum_{i,x_i=x}\mathcal{L}_i$ `and` $N[x]=\#\{i;x_i=x\}$. `*/`
3 **for** $i=0$ **to** $N-1$ **do**
4 $\quad x=x_i$; $N[x]=N[x]+1$; $\overline{\mathcal{L}}_x=\overline{\mathcal{L}}_x+\overrightarrow{\ell}_i\overrightarrow{\ell}_i^{\top}$; $\overline{\ell_x}=\overline{\ell_x}+\overrightarrow{\ell}_i$

`/* Instantaneous TA attacks Processing */`
5 **for** $\hat{k}=0$ **to** 2^n-1 **do**
$\quad$`/* Test hyp.` $\hat{k}$ `*/`
6 $\quad\mathcal{ML}[\hat{k}]=0$
7 $\quad$**for** $x=0$ **to** 2^n-1 **do**
8 $\quad\quad\hat{z}=F_j(x,\hat{k})$
9 $\quad\quad$**for** $u=0$ **to** $d-1$ **do**
10 $\quad\quad\quad$**for** $u'=0$ **to** $d-1$ **do**
11 $\quad\quad\quad\quad\mathcal{ML}[\hat{k}]=\mathcal{ML}[\hat{k}]-\overline{\mathcal{L}}_x[u][u']\times\mathrm{invCov}_{\hat{z}}[u][u']$
12 $\quad\quad\mathcal{ML}[\hat{k}]=\mathcal{ML}[\hat{k}]+\mathrm{sumMeanCov}_z\times\overline{\ell_x}-N[x]\times(\mathrm{meanCov}_{\hat{z}}+\log\mathrm{Det}_{\hat{z}})$

`/* Most likely candidate selections */`
13 candidate $=\mathrm{argmax}_{\hat{k}}(\max\mathcal{ML}[\hat{k}])$
14 **return** candidate

5.4 Experiments

To study the effectiveness of `TA` attacks in practice (and to confirm the observations reported in [33]) we experimented them against the families of devices A, B and C for three different scenarios. In the first scenario (referred to as "copy

1 → copy 1"), the profiling and the attacks are performed on the same device copy. In the second and third scenarios (respectively referred to as "copy 1 → copy 2" and "copy 1 → copy 3"), the profiling made for copy 1 is used to attack the second and third copies. For each of these 9 attacks frameworks, we plot in Figure 7 the average rank of the correct sub-key (in color) with respect to both the number of traces used for the profiling (in ordinate) and the number of traces used for the attack (in abscissa). The rank averaging has been done over 10 attacks.

In the first scenario, a profiling done on 15000 (resp. 47000) traces on Device B (resp. Device C) allows for a very efficient attack phase (the correct sub-key ranked first with less than ten traces). Moreover, it may be observed that a profiling on 8000 traces for Devices B and C is sufficient to have a successful attack in less than 23 (resp. 90) traces for device B (resp. C). For Device A, the `TA` attack in Scenario 1 is one order of magnitude less efficient (roughly speaking the values are multiplied per ten w.r.t. the traces for devices B and C).

Attacks on Devices B (resp. C) perform quite similarly in Scenarios 2 and 3. For Device A, a profiling performed on copy 1 for 18000 traces is sufficient to successfully attack copies 2 and 3 with less than 10 traces. Moreover, a profiling on 8000 traces enables successful attacks for less than 30 traces. For Device C, it may be observed that, even for a profiling performed on 50000 traces, the attacks on copies 2 and 3 require at least 80 traces to succeed. However, a profiling on 9000 traces is sufficient to have the `TA` succeeding in less than 130 traces.

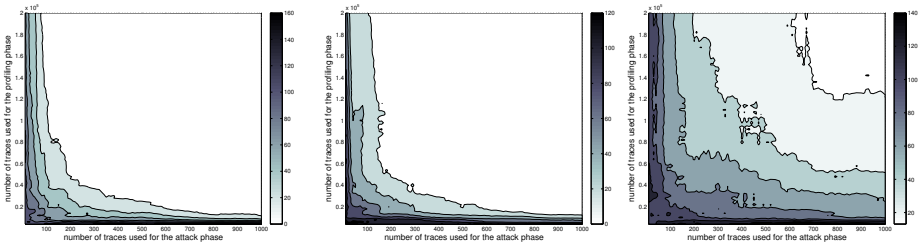
As expected, we may observe a significant variability for the attack results in Scenarios 2 and 3 for Device A: templates done on copy 1 are almost as efficient to attack copy 2 than they were to attack copy 1 itself. They are however much less informative on the copy 3 behaviour since the profiling on copy 1 must be performed on at least 130000 traces to see the attack working on copy 3 with less than 700 traces. This observation is in-line with those done in [32] about the high variability of nano-scale technologies (we recall that Device A is made in a 90nm CMOS technology).

In Appendix C, we report on similar experiments results when only the leakage means (and not the covariance matrices) are involved in the templates. This approach indeed seems to be a natural alternative to the attacks described here since the traces contain instantaneous leakages. Our results actually confirm this feeling and it can even be noticed that it leads to improve the `TA` efficiency for Scenario 3 on Device A[15]. Another general remark on these simplified templates is that they perform much better than the full ones when the number of traces used for the profiling is small (around 4000).

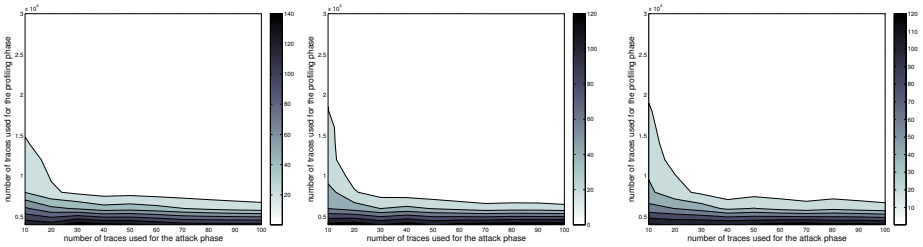
6 conclusion

In this paper, we have studied the effectiveness and efficiency of the `CPA`, the `LRA` and the `TA` attacks when performed in a context where the exact time of

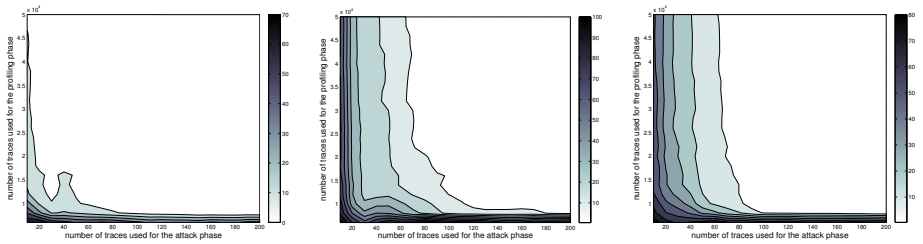
[15] This could be explained by the fact that the technology variability has more impact on the electromagnetic leakage variances than it has on the means.



(a) Dev. A: copy 1→copy 1 (b) Dev. A: copy 1→copy 2 (c) Dev. A: copy 1→copy 3



(d) Dev. B: copy 1→copy 1 (e) Dev. B: copy 1→copy 2 (f) Dev. B: copy 1→copy 3



(g) Dev. C: copy 1→copy 1 (h) Dev. C: copy 1→copy 2 (i) Dev. C: copy 1→copy 3

Fig. 7. `TA` campaigns – Rank evolution *vs.* nb. of traces for the attack phase (x-axis) and the profiling (y-axis)

the sensitive computations is not known. In this situation, and even after the application of pattern matching or resynchronization techniques, the exploited leakage traces may be composed of several thousands of points and the same attack must be processed for each of those points. We noticed that the study of the side channel attacks effectiveness and efficiency in this multivariate context is an over-estimated problem. Most of the time, it is indeed assumed that the adversary succeeded in significantly reducing the traces size (*e.g.* by priorly processing a SNR, or a test attack, or even a dimension reduction). However, as argued in this paper, those techniques are either unrealistic or may lead to a significant loss of useful information (a dimension reduction technique like the PCA may be sound for one attack – *e.g.* the `CPA` – and not for another one – *e.g.* the `MIA` or the `LRA` –). As a consequence, there was no work discussing about the rule to apply in order to select a candidate among all of those returned by a same attack performed against several time coordinates. To the best of our knowledge, the *de facto* rule was hence to simply choose the key candidate maximising all the attacks scores. In this paper, we have shown that this rule does not work for a `LRA` attack and we have conducted a statistical analysis to deduce a new selection rule that renders it effective in practice, even when the traces are composed of huge number of points. By following the same approach for the multivariate `CPA`, we have observed that the classical *total maximisation* selection rule was indeed very effective in practice and that the normalization did not led to any improvement. In this paper, we have also tackled out the efficiency problem for the multivariate `CPA`, `LRA` and `TA` attacks. For each of them, we have followed a similar approach which led us to significantly reduce their complexity when the number of traces and their dimension are high. Eventually, all our results and analyses have been illustrated by several attack experiments on three different copies of three different technologies. In particular, the latter experiments have enabled us to confirm the practicability of template attacks when the profiling phase and the attack are performed on different copies of the same device.

References

1. C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template Attacks in Principal Subspaces. In Goubin and Matsui [14], pages 1–14.
2. J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede. Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. In O. Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2012.
3. M. Bär, H. Drexler, and J. Pulkus. Improved Template Attacks. In *Constructive Side-Channel Analysis and Secure Design - First International Workshop, COSADE 2010, Darmstadt, Germany, Feb. 4-5, 2010. Proceedings*, 2010.
4. T. Bartkewitz and K. Lemke-Rust. A high-performance implementation of differential power analysis on graphics cards. In Rijmen and Prouff [34], pages 252–265.
5. L. Batina, B. Gierlichs, and K. Lemke-Rust. Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, editors, *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2008.

6. L. Batina, J. Hogenboom, and J. G. J. van Woudenberg. Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In O. Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2012.
7. L. Bohy, M. Neve, D. Samyde, and J.-J. Quisquater. Principal and Independent Component Analysis for Crypto-systems With Hardware Unmasked Units. In *Proceedings of E-Smart*, 2003.
8. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
9. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski Jr., **c**C. Ko**c**c, 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. G. Dabosville, J. Doget, and E. Prouff. A New Second Order Side Channel Attack Based on Linear Regression. *IEEE Transactions on Computers*, 2012. To appear.
11. J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate Side Channel Attacks and Leakage Modeling. *Journal of Cryptographic Engineering*, 1(2):123–144, 2011.
12. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In Oswald and Rohatgi [27], pages 426–442.
13. B. Gierlichs, K. Lemke-Rust, and C. Paar. Templates vs. Stochastic Methods. In Goubin and Matsui [14], pages 15–29.
14. L. Goubin and M. Matsui, editors. *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*. Springer, 2006.
15. A. Heuser, W. Schindler, and M. Stöttinger. Revealing Side-Channel Issues of Complex Circuits by Enhanced Leakage Models. In W. Rosenstiel and L. Thiele, editors, *DATE*, pages 1179–1184. IEEE, 2012.
16. M. Kasper, W. Schindler, and M. Stoettinger. A Stochastic Method for Security Evaluation of Cryptographic FPGA Implementations. In *IEE International Conference on Field-Programmable Technology (FPT'10)*. IEEE Press, Dec. 2010.
17. I. Kizhvatov. Side Channel Analysis of AVR XMEGA Crypto Engine. In D. N. Serpanos and W. Wolf, editors, *WESS*. ACM, 2009.
18. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
19. T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servire, and J.-L. Lacoume. A Proposition for Correlation Power Analysis Enhancement. In Goubin and Matsui [14], pages 174–186.
20. S. J. Lee, S. C. Seo, D.-G. Han, S. Hong, and S. Lee. Acceleration of Differential Power Analysis through the Parallel Use of GPU and CPU. *IEICE Transactions*, 93-A(9):1688–1692, 2010.
21. S. Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In T. Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
22. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
23. S. Mangard, E. Oswald, and F.-X. Standaert. One for All - All for One: Unifying Standard DPA Attacks. *IET Information Security*, 2011.

24. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant software. In **c**C. Ko**c**c and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
25. D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In Preneel and Takagi [28], pages 207–222.
26. D. Oswald and C. Paar. Improving Side-Channel Analysis with Optimal Linear Transforms. In S. Mangard, editor, *CARDIS*, volume 7771 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2012.
27. E. Oswald and P. Rohatgi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*. Springer, 2008.
28. B. Preneel and T. Takagi, editors. *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
29. W. Press. *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Fortran Numerical Recipes. Cambridge University Press, 1992.
30. J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In I. Attali and T. Jensen, editors, *Smart Card Programming and Security – E-smart 2001*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
31. C. Rechberger and E. Oswald. Practical Template Attacks. In C. H. Lim and M. Yung, editors, *WISA 2004*, volume 3325 of *Lecture Notes in Computer Science*, pages 443–457. Springer, 2004.
32. M. Renauld, D. Kamel, F.-X. Standaert, and D. Flandre. Information Theoretic and Security Analysis of a 65-Nanometer DDSLL AES S-Box. In Preneel and Takagi [28], pages 223–239.
33. M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011.
34. V. Rijmen and E. Prouff, editors. *Smart Card Research and Advanced Applications, 10th International Conference – CARDIS 2011*, Lecture Notes in Computer Science. Springer, 2011.
35. W. Schindler. Advanced Stochastic Methods in Side Channel Analysis on Block Ciphers in the Presence of Masking. *Journal of Mathematical Cryptology*, 2:291–310, 2008.
36. W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
37. F.-X. Standaert and C. Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In Oswald and Rohatgi [27], pages 411–425.
38. F.-X. Standaert, B. Gierlichs, and I. Verbauwhede. Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In P. J. Lee and J. H. Cheon, editors, *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2009.

39. F.-X. Standaert, F. Koeune, and W. Schindler. How to Compare Profiled Side-Channel Attacks? In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 485–498, 2009.
40. F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework For The Analysis of Side-Channel Attacks. In *EUROCRYPT*, volume 5479 of *Lecture Notes In computer Science*, pages 443–461. Springer, 2009.
41. F.-X. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard. The World is not Enough: Another Look on Second-Order DPA. In M. Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.
42. Télécom ParisTech. DPA Contest v1 and v2. `http://www.dpacontest.org/`, retrieved on aug. 1, 2012.
43. C. Whitnall and E. Oswald. A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2011.
44. C. Whitnall and E. Oswald. A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. *J. Cryptographic Engineering*, 1(2):145–160, 2011.
45. C. Whitnall, E. Oswald, and L. Mather. An Exploration of the Kolmogorov-Smirnov Test as Competitor to Mutual Information Analysis. In Rijmen and Prouff [34], pages 316–334.

A Detailed Complexities of the LRA and CPA attacks

Method	Operations Complexity[16]	Memory Complexity
classical LRA	$2^n dN(2sd + 2s + 3)$	$d + 2^n * N * s + 2^n * N * s + N * d$
LRA with improvement	$N(d+1) + 2^n d(2^n(sd + 2s + 3) + 3)$	$d + 2^{2n} * s + 2^{2n} * s + 2^n * d$

Number of operations w.r.t. the number of traces N, the dimension d of the traces and the size s of the regression basis.

Method	Operations Complexity[16]	Memory Complexity
classical CPA	$d \times 2^n \times (2N + 6)$	$2^{n+1} + 2d$
CPA with improvement	$2N + 2^n \times (1 + d \times (2^n + 6))$	$2^n(d + 4))$

Number of operations w.r.t. the number of traces N and the dimension d of the traces.

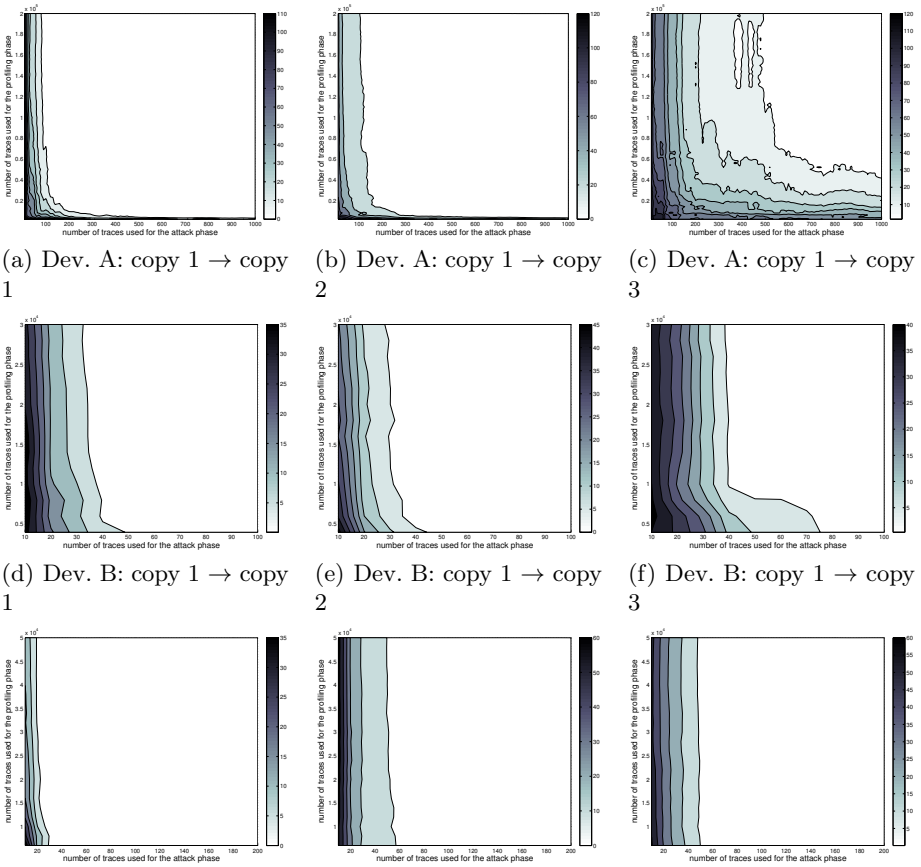
B Brief Description of Algorithm 4

Steps 1-5 process the (sampling) mean and the (sampling) variance of each coordinates of the leakage vectors. It results in two vectors $\mu_{\overrightarrow{\mathbf{L}}}$ and $\sigma_{\overrightarrow{\mathbf{L}}}$ with same dimension as $\overrightarrow{\mathbf{L}}$. Steps 6-12 process the (sampling) mean and the (sampling) variance of the hypotheses sample $(\mathtt{m}(F(x_i, \hat{k})))_i$ for each key candidate $\hat{k}$. When N grows, those latter mean and variance quickly tend toward $\mathbb{E}\left[(\mathtt{m}(F(X, \hat{k}))\right]$ and $\mathrm{var}[(\mathtt{m}(F(X, \hat{k}))]$, which can be directly deduced from $\mathtt{m}(\cdot)$ and F (*e.g.* if F is an

[16] in terms of elementary n-bit words operations

sbox and $\mathtt{m}(\cdot)$ is the Hamming weight function, we have $\mathbb{E}\left[(\mathtt{m}(F(X,\hat{k}))\right] = m/2$ and $\mathrm{var}[(\mathtt{m}(F(X,\hat{k}))] = m/4$ assuming X is uniform). Eventually, Steps 13-18 process, for each key candidate $\hat{k}$, the correlation between $(\mathtt{m}(F(x_i,\hat{k})))_i$ and each sample $(\overrightarrow{\ell}_i[u])_i$ where u denotes the u^{th} coordinate of the leakage vectors.

C Additional Figures



(a) Dev. A: copy 1 → copy 1 (b) Dev. A: copy 1 → copy 2 (c) Dev. A: copy 1 → copy 3

(d) Dev. B: copy 1 → copy 1 (e) Dev. B: copy 1 → copy 2 (f) Dev. B: copy 1 → copy 3

(g) Dev. C: copy 1 → copy 1 (h) Dev. C: copy 1 → copy 2 (i) Dev. C: copy 1 → copy 3

Fig. 8. `TA` campaigns with mean templates only – Rank evolution *vs.* nb. of observations

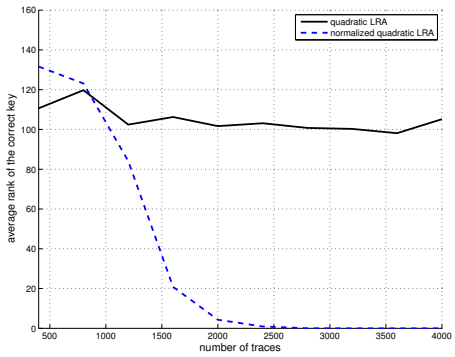


Fig. 9. Quadratic `LRA` campaign – Rank evolution *versus* number of observations