Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting

Benoît Gérard* and François-Xavier Standaert**

UCL Crypto Group, Université catholique de Louvain
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium

Abstract. Side-channel collision attacks are one of the most investigated techniques allowing the combination of mathematical and physical cryptanalysis. In this paper, we discuss their relevance in the security evaluation of leaking devices with two main contributions. On the one hand, we suggest that the exploitation of linear collisions in block ciphers can be naturally re-written as a Low Density Parity Check Code decoding problem. By combining this re-writing with a Bayesian extension of the collision detection techniques, we succeed in improving the efficiency and error tolerance of previously introduced attacks. On the other hand, we provide various experiments in order to discuss the practicality of such attacks compared to standard DPA. Our results exhibit that collision attacks are less efficient in classical implementation contexts, e.g. 8-bit microcontrollers leaking according to a linear power consumption model. We also observe that the detection of collisions in software devices may be difficult in the case of optimized implementations, because of less regular assembly codes. Interestingly, the soft decoding approach is particularly useful in these more challenging scenarios. Finally, we show that there exist (theoretical) contexts in which collision attacks succeed in exploiting leakages whereas all other non-profiled side-channel attacks fail.

1 Introduction

Most side-channel attacks published in the literature and used to evaluate leaking cryptographic devices are based on a divide-and-conquer strategy. Kocher et al.'s Differential Power Analysis (DPA) [10], Brier et al.'s Correlation Power Analysis (CPA) [5] and Chari et al.'s Template Attacks (TA) [6] are notorious examples. However, alternatives to these standard approaches have also been investigated, e.g. by trying to combine side-channel information with classical cryptanalysis. The collision attacks introduced by Schramm et al. at FSE 2003 are among the most investigated solutions for this purpose [19]. While initially dedicated to the DES, they have then been applied to the AES [18] and improved in different directions over the last years, as witnessed by the recent works of Ledig et al. [11], Bogdanov [2,3,4], Moradi et al. [13,14] and Clavier et al. [7].

* Postdoctoral researcher supported by Walloon region MIPSs project.
** Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC project 280141 (acronym CRASH).

E. Prouff and P. Schaumont (Eds.): CHES 2012, LNCS 7428, pp. 175–192, 2012.

From an application point of view, collision attacks differ from standard side-channel attacks by their underlying assumptions. Informally, divide-and-conquer distinguishers essentially assume that a cryptographic device leaks information that depends on its intermediate computations, under a given leakage model. The leakage model is generally obtained either from engineering intuition, in the case of non-profiled attacks such as DPA and CPA, or through a preliminary estimation of the chip measurements probability distribution, in the case of profiled attacks such as TA. By contrast, collision attacks do not require a precise knowledge of the leakage distribution. They rather trade this need for a combination of two other assumptions: (*i*) the distribution of a couple of measurements corresponding to the intermediate computation of identical values can be distinguished from the one corresponding to different values; (*ii*) the adversary is able to divide each measurement trace corresponding to the encryption of a plaintext into sub-traces corresponding to elementary operations, e.g. the execution of block cipher S-boxes. In other words, collision attacks trade the need of precise leakage models for the need to detect identical intermediate computations, together with a sufficient knowledge of the operations scheduling in the target device. Interestingly, the knowledge of precise leakage models has recently been shown to be problematic in non-profiled attacks [22], e.g. in the case of devices with strongly non-linear leakage functions. Hence, although the existence of such devices remains an open question [16], they at least create a theoretical motivation for understanding the strengths and weaknesses of collision attacks.

This paper brings two main contributions related to this state-of-the-art.

First, we observe that many previous collision attacks do not efficiently deal with errors (i.e. when the correct value of a key-dependent variable is not the likeliest indicated by the leakages), and rely on add-hoc solutions for this purpose. In order to handle erroneous situations more systematically, we introduce two new technical ingredients. On the one hand, we propose to re-write side-channel collision attacks as a Low Density Parity Check (LDPC) decoding problem. On the other hand, we describe a (non-profiled) Bayesian extension of collision detection techniques. We show that these tools are generic and allow successful key recoveries with less measurement data than previous ones, by specializing them to two exemplary attacks introduced by Bogdanov [2,3] and Moradi et al. [13].

Second, we question the relevance of side-channel collision attacks and their underlying assumptions, based on experimental case studies. For this purpose, we start by showing practical evidence that in "simple" scenarios, the efficiency of these attacks is lower than the one of more standard attacks, e.g. the non-profiled extension of Schindler's stochastic approach [17], described in [8]. We then observe that in actual software implementations, the detection of collisions can be difficult due to code optimizations. As a typical example, we observe that the leakage behavior of different AES S-boxes in an Atmel microcontroller may be different, which prevents the detection of a collision with high confidence for these S-boxes. We conclude by exhibiting an (hypothetical) scenario were the leakage function is highly non-linear (i.e. in the pathological example from [22]), collision attacks lead to successful key recoveries whereas all non-profiled attacks fail.

2 Background

2.1 Notations

In order to simplify the understanding of the paper, we will suppose that the targeted block cipher is the AES Rijndael. Hence, the number of S-boxes considered is 16, and these S-boxes manipulate bytes. Nevertheless, all the following statements can be adapted to another key alternating cipher, by substituting the correct size and number of S-boxes. In this context, the first-round subkey and plaintexts are all 16-byte states. We respectively use letters k and x for the key and a plaintext, and use subscripts to point to a particular byte:

$$x \stackrel{\text{def}}{=} (x_1, x_2, \ldots, x_{16}) \quad , \quad k \stackrel{\text{def}}{=} (k_1, k_2, \ldots, k_{16}).$$

Next, the attackers we will consider have access to a certain number of side-channel traces, corresponding to the encryption of different plaintexts encrypted using the same key k. We denote with n_t the number of different inputs encrypted, and with $x^{(1)}, \ldots, x^{(n_t)}$ the corresponding plaintexts. Each trace obtained is composed of 16 sub-traces corresponding to the 16 S-box computations $t^{(i)} \stackrel{\text{def}}{=} (t_1^{(i)}, \ldots t_{16}^{(i)})$. Each sub-trace is again composed of a number ℓ of points (or samples). Hence, the sub-trace corresponding to the a-th S-box will be denoted as $t_a^{(i)} \stackrel{\text{def}}{=} (t_{a,1}^{(i)}, \ldots, t_{a,\ell}^{(i)})$. Furthermore, we will use the corresponding capital letters X, K and T to refer to the corresponding random variables.

2.2 Linear Collision Attacks

Linear collision attacks are based on the fact that if an attacker is able to detect a collision between two (first-round) S-box executions, then he obtains information about the key. Indeed, if a collision is detected, e.g. between the computation of S-box a for plaintext $x^{(i_a)}$ and S-box b for plaintext $x^{(i_b)}$, this attacker obtains a linear relation between the two corresponding input bytes:

$$x_a^{(i_a)} \oplus k_a = x_b^{(i_b)} \oplus k_b.$$

This relation allows him to decrease the dimension of the space of possible keys by 8, removing keys for which $k_a \oplus k_b \neq x_a^{(i_a)} \oplus x_b^{(i_b)}$. A linear system can then be built by combining several equations, and solving this system reveals (most of) the key. Naturally, the success of the attack mainly depends on the possibility to detect collisions. Two main approaches have been considered for this purpose.

In the first approach, simple statistics such as the Euclidean distance [18] or Pearson's correlation coefficient [19], are used as detection metrics. In this case, the detection of a collision can be viewed as a binary hypothesis test. It implies to define an acceptance region (i.e. a threshold on the corresponding statistic). As a result, a collision may not be detected and a false collision may be considered as a collision. This second point is the most difficult to overcome, as a false-collision implies adding a false equation in the system, which in turn implies the

attack failure. Heuristic solutions based on binary and ternary vote have then been proposed in [3] to mitigate this issue. In binary vote, the idea is to observe the same supposed collision using many traces, and to take a hard decision by comparing the number of times the collision detection procedure returns `true` with some threshold. Ternary vote is based on the fact that if there is a collision between two values, then the output of the collision-detection procedure should be the same when comparing both traces with a third one.

An alternative approach is the correlation-enhanced attack introduced by Moradi et al. [13]. This approach is somehow orthogonal to the first one, since we are not in the context of binary hypothesis testing anymore. Namely, instead of only returning `true` or `false`, a comparison procedure directly returns the score obtained using the chosen statistic (e.g. Pearson's correlation coefficient). Hence, when comparing two sub-traces $t_a^{(i)}$ and $t_b^{(j)}$, we obtain a score that is an increasing function of the likelihood of $K_a \oplus K_b$ being equal to $x_a^{(i)} \oplus x_b^{(j)}$.

Besides, the authors of [13] combined their attack with a pre-processing of the traces, that consists in building "on-the-fly" templates of the form:

$$\bar{t}_a^{(x)} = \frac{\sum_{i, x_a^{(i)} = x} t_a^{(i)}}{\#\{i, x_a^{(i)} = x\}}. \tag{1}$$

Such a pre-processing is typically useful to extract first-order side-channel information (i.e. difference in the mean values of the leakage distributions).

3 General Framework for Linear Collision Attacks

In this section, we propose a general framework for describing the different linear collision attacks that have been proposed in the literature. One important contribution of this framework is to represent these attacks as a decoding problem. In particular, we argue that a natural description of collision attacks is obtained through the theory of LDPC codes, designed by Gallager in 1962 [9].

3.1 Collision Attacks as an LDPC Decoding Problem

We start with the definition of LDPC codes.

Definition 1. *LDPC codes (graph representation). Let $\mathcal{G}$ be a bipartite graph with m left nodes and r right nodes. Let us denote by $\mathcal{G}_E$ the set of edges i.e. $(i, j) \in \mathcal{G}_E$ if and only if the i-th left node and the j-th right node are adjacent. This graph defines a code $\mathcal{C}$ of length m over $\mathbb{F}_q^m$, such that for $w = (w_1, w_2, \ldots, w_m) \in \mathbb{F}_q^m$, we have:*

$$w \in \mathcal{C} \Longleftrightarrow \forall 1 \leq j \leq r, \bigoplus_{i,(i,j)\in\mathcal{G}_E} w_i = 0.$$

This code is said to be an (m, i, j) LDPC code if the maximum degree for a left nodes is i and the maximum degree for a right nodes is j.

In general, left nodes are called message nodes while right nodes are named check nodes, since they correspond to conditions for code membership. This definition can be directly related to our collision attack setting. First observe that a collision between S-boxes a and b provides information on the variable:

$$\Delta K_{a,b} \stackrel{\text{def}}{=} K_a \oplus K_b.$$

It follows that the vector $\Delta K \stackrel{\text{def}}{=} (\Delta K_{1,2}, \ldots, \Delta K_{15,16})$ determines a coset of K of size 2^8. Hence, it can be seen as a codeword of an LDPC code of dimension 15 and length 120. This LDPC code corresponding to our problem has a very particular structure: the set of check nodes only contains right nodes of degree equal to 3. These nodes correspond to the linear relationships:

$$\Delta K_{a,b} \oplus \Delta K_{a,c} = \Delta K_{b,c}, \quad \forall\, 1 \leq a < b < c \leq 16.$$

Therefore, finding the key in a linear collision attack consists in finding the likeliest codeword of the aforementioned LDPC code, and then exhaustively testing the keys derived from this system by setting K_1 to each of its 2^8 possible values. This LDPC formulation for the linear collision attack problem allows the use of a decoding algorithm to recover the likeliest system of equations. In general, it is well known that the performances of such a decoder can be drastically improved when soft information is available. Interestingly, soft information is naturally available in our context, e.g. through the scores obtained for each possible value of a variable $\Delta K_{a,b}$. Nevertheless, these scores do not have a direct probabilistic meaning. This observation suggests that a Bayesian extension of the statistics used for collision detection, where the scores would be replaced by actual probabilities, could be a valuable addition to collision attacks, in order to boost the decoder performances. As will be shown in Section 5, this combination of LDPC decoding and Bayesian statistics can indeed lead to very efficient attacks.

3.2 General Framework

A general description of linear collision attacks is given in Algorithm 1 and holds in five main steps. First, the traces may be prepared with a `PreProcessTraces` procedure. For example, signal processing can be applied to align traces or to remove noise. Instantiations of this procedure proposed in previous attacks [3,13] will be discussed in Section 4.1. Next, the scores $S_{a,b} \stackrel{\text{def}}{=} (S_{a,b}(\delta))_{\delta \in \mathbb{F}_{256}}$ corresponding to the possible values δ of the variables $\Delta K_{a,b}$ are extracted (with the `ComputeStatistics` procedure). Different techniques have again been proposed for this purpose in the literature. In order to best feed the LDPC decoder, the scores can be turned into distributions for the variables $\Delta K_{a,b}$, thanks to an `ExtractDistributions` procedure. As will be discussed in Section 4.2, this can be obtained by normalization, or by applying a Bayesian extension of the computed statistics. In particular, we will show how meaningful probabilities can be outputted for two previously introduced similarity metrics (in a non-profiled setting). Using these distributions, the `LDPCDecode` procedure then returns a list of the ℓ most likely codewords that correspond to the most likely consistent

Algorithm 1. General framework for linear-collision attacks

Input: n_t plaintexts $x^{(1)}, \ldots, x^{(n_t)}$ and the corresponding traces $t^{(1)}, \ldots, t^{(n_t)}$.
Output: The key k used by the targeted device.
$(\bar{t}_1, \ldots, \bar{t}_{16}) \leftarrow$ `PreProcessTraces`$(x^{(1)}, \ldots, x^{(n_t)}, t^{(1)}, \ldots, t^{(n_t)})$;
foreach $1 \leq a < b \leq 16$ **do**
  $S_{a,b} \leftarrow$ `ComputeStatistics`$(\bar{t}_a, \bar{t}_b)$;
$\Pr[\Delta K] \leftarrow$ `ExtractDistributions`$(S_{1,2}, \ldots, S_{15,16})$;
$\{\mathcal{S}_1, \ldots, \mathcal{S}_\ell\} \leftarrow$`LDPCDecode`$(\Pr[\Delta K])$;
foreach *system* $\mathcal{S}_i$ *and key candidate* k *compatible with equations in* $\mathcal{S}_i$ **do**
  if `TestKey`(k) **then**
    return k;
return `failure`;

systems $\{\mathcal{S}_1, \ldots, \mathcal{S}_\ell\}$ of 120 equations (with $\mathcal{S}_i$ more likely than $\mathcal{S}_{i+1}$). Such a decoding algorithm is detailed in Section 4.3 for the case $\ell = 1$. Finally, the 2^8 full keys fulfilling $\mathcal{S}_1$ are tested in the `TestKey` procedure. The correct key is returned if found otherwise keys fulfilling $\mathcal{S}_2$ are tested and so on. If the correct key does not fulfill any of the $\mathcal{S}_i$'s, then `failure` is returned.

4 Instantiation of the Framework Procedures

Following the previous general description, we now propose a few exemplary instantiations of its different procedures. Doing so, we show how to integrate previously introduced collision attacks in our framework.

4.1 Pre-processing

Pre-processing the traces is frequently done in side-channel analysis, and collision attacks are no exceptions. For example, Bogdanov's attacks take advantage of averaging (by measuring the power consumption of the same plaintext several times), in order to reduce the measurement noise [2,3,4]. Similarly, Moradi et al. [13] start by building the "on-the-fly" templates defined in Equation (1). This latest strategy shows good results in attacks against unprotected implementations with first-order leakages and our experiments in Section 5 will exploit it[1].

4.2 Information Extraction

The use of an LDPC soft-decoding algorithm requires to extract distributions for the variables $\Delta K_{a,b}$. As mentioned in Section 3.1, such distributions can be obtained heuristically, by normalizing scores obtained with classical detection techniques. But the optimal performances of a soft-decoder are only reached when these distributions correspond to actual probabilities $\Pr\left[\Delta K_{a,b} = \delta \middle| S(a,b,\delta)\right]$.

[1] By contrast, averaging is detrimental in the case of masked implementations with only second-order leakages, as detailed in [7]. As an alternative, the authors of this paper concatenate sub-traces $t_a^{(i)}$ corresponding to the same plaintext x and form a vector $\bar{t}_a$ by collecting these concatenated sub-traces for different plaintext values.

While such probabilities are easily computed in profiled attacks, obtaining them in a non-profiled setting requires more efforts and some assumptions. In this section, we first introduce general tools that may be applied to any given detection technique for this purpose. For illustration, we then apply them to both the Euclidean distance (ED) and the correlation-enhanced (CE) detection techniques.

Bayesian Extensions: General Principle. The naive approach for extracting distributions from scores $S(a, b, \delta)$, obtained for a candidate $\Delta K_{a,b} = \delta$, is to apply normalization:

$$\text{Norm}(S(a, b, \delta)) \stackrel{\text{def}}{=} \frac{S(a, b, \delta)}{\sum_{\delta'} S(a, b, \delta')}.$$

As already mentioned, such normalized scores are not directly meaningful since they do not correspond to actual probabilities $\Pr\left[\Delta K_{a,b} = \delta \middle| S(a, b, \delta)\right]$. Therefore, and as an alternative, we now propose a Bayesian technique for computing scores that corresponds to these probabilities and is denoted as:

$$\text{BayExt}(S(a, b, \delta)) \approx \Pr\left[\Delta K_{a,b} = \delta \middle| S(a, b, \delta)\right],$$

where the $\approx$ symbol recalls that the distributions are estimated under certain (practically relevant) assumptions. For this purpose, we introduce the next model.

Model 1. *Let T (resp. T') be the sub-trace corresponding to the execution of an S-box with input X (resp. X'). Let $S(T, T')$ be a statistic extracted from the pair of traces (T, T') (typically the Euclidean distance or a correlation coefficient). Then, there exists two different distributions $\mathcal{D}_c$ and $\mathcal{D}_{nc}$ such that:*

$$\Pr\left[S(T, T') = s\right] = \begin{cases} \Pr_{\mathcal{D}_c}\left[S = s\right] & \text{if } X = X', \\ \Pr_{\mathcal{D}_{nc}}\left[S = s\right] & \text{otherwise.} \end{cases}$$

We note that in theory, the distribution of the statistic in the non-collision case should be a mixture of different distributions, corresponding to each pair of non-colliding values. However, in the context of non-profiled attacks, estimating the parameters of these distributions (mean and variance, typically) for each component of the mixture would require a large amount of measurement traces (more than required to successfully recover the key). Hence, we model this mixture as a global distribution. As will be clear from our experimental results, this heuristic allows us to perform successful attacks with small amounts of measurement traces. Model 1 directly implies that the distribution of $\Delta K_{a,b}$ can be expressed using $\Pr_{\mathcal{D}_c}[\cdot]$ and $\Pr_{\mathcal{D}_{nc}}[\cdot]$, as stated in the following Lemma.

Lemma 1. *Let $\Sigma \stackrel{\text{def}}{=} (s_i, \Delta x_i)_{1 \le i \le n}$ be the set of observed statistics s_i and the corresponding suggested value Δx_i for a given XOR of key bytes $\Delta K_{a,b}$. Then:*

$$\begin{aligned} \Pr\left[\Delta K_{a,b} = \delta \middle| \Sigma\right] &\propto \prod_{i=1}^{n} \Pr\left[S = s_i \middle| \Delta x_i, \Delta K_{a,b} = \delta\right], \\ &\propto \prod_{i, \Delta x_i = \delta} \Pr_{\mathcal{D}_c}\left[S = s_i\right] \prod_{i, \Delta x_i \neq \delta} \Pr_{\mathcal{D}_{nc}}\left[S = s_i\right], \end{aligned}$$

where $\Pr_{\mathcal{D}_c}[S = s_i]$ *(resp.* $\Pr_{\mathcal{D}_{nc}}[S = s_i]$*) denotes the distribution of the statistic* S *when resulting from the comparison between to identical (resp. different) inputs. Moreover, if for any* i, $\Pr_{\mathcal{D}_{nc}}[S = s_i]$ *is non-zero, then:*

$$\Pr\left[\Delta K_{a,b} = \delta \middle| \Sigma\right] \propto \prod_{i,\Delta x_i = \delta} \frac{\Pr_{\mathcal{D}_c}[S = s_i]}{\Pr_{\mathcal{D}_{nc}}[S = s_i]}.$$

Proof. The first line is a direct application of Bayes' relation, the second results from Model 1, and the final formula is obtained dividing by $\prod_i \Pr_{\mathcal{D}_{nc}}[S = s_i]$.$_\diamond$

In order to solve our estimation problem, we have no other *a priori* information on $\mathcal{D}_c$ and $\mathcal{D}_{nc}$ than their non-equality. This problem is a typical instance of data clustering. That is, the set of observations s_i is drawn from a mixture of two distributions $\mathcal{D}_c$ and $\mathcal{D}_{nc}$, with respective weights 2^{-8} and $(1 - 2^{-8})$. For both detection metrics in this paper, we show next that it is easy to theoretically predict one out of the two distributions. We will then estimate the parameters of the other distribution based on this prediction and some additional measurements. Lemma 2 (proven in Appendix A) provides formulas to estimate the non-collision distribution parameters based on the collision ones. Moving from the collision to the non-collision distribution can be done similarly.

Lemma 2. *Let* $\mathcal{D}$ *be a mixture of two distributions* $\mathcal{D}_c$ *and* $\mathcal{D}_{nc}$ *with respective weights* 2^{-8} *and* $1 - 2^{-8}$*. Let us denote by* $\bar{\mu}$ *and* $\bar{\sigma^2}$ *estimates for the expected value and variance of* $\mathcal{D}$ *obtained from observed values. Similarly, we denote* $(\bar{\mu}_c, \bar{\sigma^2}_c)$ *estimates obtained for* $\mathcal{D}_c$*. Then, we can derive the following estimates for expected value and variance of* $\mathcal{D}_{nc}$*:*

$$\bar{\mu}_{nc} = \frac{\bar{\mu} - 2^{-8}\bar{\mu}_c}{1 - 2^{-8}}, \quad \text{and} \quad \bar{\sigma^2}_{nc} = \frac{\bar{\sigma^2} - 2^{-16}\bar{\sigma^2}_c}{(1 - 2^{-8})^2}.$$

Specialization to the Euclidean Distance Detection. The Euclidean distance (ED) has been proposed as a detection tool in [18] and investigated in a profiled setting in [4]. The Euclidean distance between two traces T and T' equals:

$$ED(T, T') \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} (T_j - T'_j)^2.$$

Let us first detail a natural non-Bayesian use of this similarity metric. Then, we will specialize the aforementioned framework in order to provide formulas to compute actual probabilities $\Pr\left[\Delta K_{a,b} \middle| \bar{t}_a, \bar{t}_b\right]$ from observed Euclidean distances. In general, the smaller is the Euclidean distance between traces $\bar{T}_a^{(i_a)}$ and $\bar{T}_b^{(i_b)}$, the more probable is the value $x_a^{(i_a)} \oplus x_b^{(i_b)}$ for the variable $\Delta K_{a,b}$ is. Hence, we will consider the opposite of $ED\left(\bar{T}_a^{(i_a)}, \bar{T}_b^{(i_b)}\right)$ as the score to normalize:

$$S_{ED}(a, b, \delta) \stackrel{\text{def}}{=} \text{Norm}\left(d_0 - \max_{x_a^{(i_a)} \oplus x_b^{(i_b)} = \delta} ED\left(\bar{T}_a^{(i_a)}, \bar{T}_b^{(i_b)}\right)\right), \tag{2}$$

where d_0 is chosen such that all values $d_0 - \max_{x_a^{(i_a)} \oplus x_b^{(i_b)} = \delta} ED\left(\bar{T}_a^{(i_a)}, \bar{T}_b^{(i_b)}\right)$ are strictly positive. Note that if a single trace is given, only a single Euclidean distance can be computed between each pair of S-boxes a and b. By contrast, many Euclidean distances can be computed per pair of S-boxes when the number of traces increases. This justifies the use of a (heuristic) max function to select which Euclidean distance will be retained to compute the scores. Let us now consider the application of the Bayesian framework to the use of ED. We will use ED_B to refer to this Bayesian extension of ED. As mentioned earlier, efficiently deriving probabilities from scores in a non-profiled setting requires to make some assumptions. In the following, we consider the frequent case where the leakage is the sum of a deterministic part (that depends on an intermediate computation result) and a white Gaussian noise, as proposed in [17] and stated in Model 2.

Model 2. *For any input byte X_a^i and for any point j in the corresponding sub-trace T_a^i, the power consumption $T_{a,j}^i$ is the sum of a deterministic value $L_j(X_a^i)$ and some additive white Gaussian noise $N_{a,j}^i$ of variance σ_j^2:*

$$T_{a,j}^i = L_j(X_a^i) + N_{a,j}^i.$$

To lighten notation, we will omit superscripts and subscripts when a statement applies for all inputs and sub-traces. This model admittedly deviates from the distribution of actual leakage traces, since the noise in different samples can be correlated. We note again that in non-profiled attacks, it is not possible to obtain any information on the covariances of this Gaussian noise. Nevertheless, taking points in the trace that are far enough ensures that these covariances are small enough for this approximation to be respected in practice, as will be confirmed experimentally in Section 5. In such a context, the variables $(T_j - T'_j)$ are drawn according to the Gaussian distribution $\mathcal{N}(L_j(X) - L_j(X'), 2\sigma_j^2)$. As a result, the normalized Euclidean distance becomes:

$$ED_B(T, T') \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} \frac{(T_j - T'_j)^2}{2\sigma_j^2}, \tag{3}$$

and can be modeled using χ^2 distribution family (i.e. sums of squared Gaussian random variables). Indeed, each term of the sum is distributed according to a non-central χ^2 distribution with non central parameter $(T_j - T'_j)^2$. In the case of a collision, this parameter vanishes and all the terms are drawn according to a central χ^2 distribution. Hence $\mathcal{D}_c$ is a χ^2 distribution with ℓ degrees of freedom. In the case of $\mathcal{D}_{nc}$, the attacker has no knowledge of $(L_j(X) - L_j(X'))^2$ and is unable to directly estimate the distribution. Yet, as previously mentioned it is possible to obtain a good approximation of this distribution from the parameters of $\mathcal{D}_c$ using Lemma 2. Experiments show that the shape of $\mathcal{D}_{nc}$ quickly tends towards a Gaussian distribution when increasing the number of points ℓ. Combining these observations, we obtain the following score:

$$\text{BayExt}(S_{ED}(a,b,\delta)) \stackrel{\text{def}}{=}$$
$$\text{Norm}\Big(\exp\big[\tfrac{1}{2}\sum_{x_a^{(i_a)}\oplus x_b^{(i_b)}=\delta} \frac{\left(ED_B\left(\bar{T}_a^{(i_a)},\bar{T}_b^{(i_b)}\right)-\mu_{nc}\right)^2}{\sigma_{nc}^2} - ED_B(\bar{T}_a^{(i_a)},\bar{T}_b^{(i_b)})\big]\Big). \quad (4)$$

Remark. If averages are performed during the `PreProcessTraces` procedure, the number of traces used to compute values $\bar{T}_a^{(i_a)}$ may be different. Hence, the normalized Euclidean distance ED_B has to take this into account when comparing sub-traces $\bar{T}_a^{(i_a)}$ and $\bar{T}_b^{(i_b)}$. This is done by replacing $2\sigma_j^2$ by $\sigma_j^2\left(\frac{1}{\#(a,i_a)}+\frac{1}{\#(b,i_b)}\right)$ in (3), where $\#(a,i_a)$ is the number of traces averaged to obtain $\bar{T}_a^{(i_a)}$.

Specialization to the Correlation-Enhanced Detection. We now consider the use of Pearson's correlation coefficient as detection tool. Let us recall that for two vectors U and V having the same length and mean values $\bar{U}$ and $\bar{V}$, the correlation coefficient is defined as:

$$\rho(U,V) \stackrel{\text{def}}{=} \frac{\sum_i (U_i-\bar{U})(V_i-\bar{V})}{\sqrt{\sum_i (U_i-\bar{U})^2}\sqrt{\sum_i (V_i-\bar{V})^2}}.$$

Many papers take advantage of this comparison metric in the side-channel literature. In the following, we focus on the correlation-enhanced solution proposed in [13], as it generally provides the best results. This attack applies to "on-the-fly" templates $\bar{t}$ such that $\bar{T}_a^{(x)}$ contains the sub-traces obtained by averaging the computations of an S-box a with plaintext byte x. The detection is based on the fact that if $\Delta K_{a,b}=\delta$, then traces $\bar{T}_a^{(x)}$ should correspond to (i.e. be similar with) traces $\bar{T}_b^{(x\oplus\delta)}$. We denote the permutation of the vector $\bar{T}_b$ that contains the $\bar{T}_b^{(x\oplus\delta)}$'s for increasing x values as $\bar{T}_b^{\oplus\delta}$. Then, the normalized score for a given δ is obtained with $S_{CE}(a,b,\delta) \stackrel{\text{def}}{=} \text{Norm}\left(\rho\left(\bar{T}_a,\bar{T}_b^{\oplus\delta}\right)\right)$. In practice, some values of $T_a^{(x)}$ or $T_b^{(x\oplus\delta)}$ may not be defined if few traces are used. In the case where at least one of the two traces is undefined, the coordinate will be ignored in the computation of the correlation coefficient. As for the Euclidean distance, we propose to apply the Bayesian extension to the use of the correlation-enhanced detection. The distribution of the correlation coefficient is not easy to handle, but we can approximate it with a Gaussian one using the Fisher transform of this coefficient, as proposed in [12]: $CE_B(a,b,\delta) \stackrel{\text{def}}{=} \text{arctanh}\, S_{CE}(a,b,\delta)$. Asymptotically, the random variables $CE_B(a,b,\delta)$ are normally distributed with mean equal to the expected value of $\rho\left(\bar{T}_a,\bar{T}_b^{\oplus\delta}\right)$, and variance $(N-3)^{-1}$, where N is the number of coordinates used to compute the correlation coefficient. Given these modified statistics, we now derive the corresponding Bayesian extension. For non-collisions, the correlation coefficient has an expected value of 0. Let us denote the expected value of the correlation when a collision occurs as μ_c. Since both distributions have the same variance (say σ^2), Lemma 1 translates into:

$$\Pr\left[\Delta K = \delta \middle| CE_B(a,b,\delta) = s\right] \propto \frac{e^{-\frac{(s-\mu_c)^2}{\sigma^2}}}{e^{-\frac{s^2}{\sigma^2}}} \propto \exp\left[\frac{s^2-(s-\mu_c)^2}{\sigma^2}\right] \propto \exp\left[\frac{2s}{\sigma^2}\right].$$

In practice, it turned out that distributions are really close to Gaussian, even for a small number of traces used, but their variance did not tend towards the expected value $(N-3)^{-1}$. Hence, in our following experiments, we rather used:

$$S_{CE\_B}(a,b,\delta) \stackrel{\text{def}}{=} \text{Norm}\left(e^{2\,CE_B(a,b,\delta)}\right). \tag{5}$$

Again, results in Section 5 show that even if based on slightly incorrect models, the impact of these Bayesian extensions on the attack efficiency is positive.

4.3 LDPC Decoding

A soft-decoding algorithm for non-binary LDPC codes can be found in [1] and is presented in Algorithm 2. It consists in iterating a belief propagation stage a certain number of times. Let us recall that a code can be represented using a bipartite graph: left nodes are message nodes and correspond to positions of the codeword; right nodes are check nodes that represent redundancy constraints. The attacker receives distributions for the message nodes. The belief propagation step boils down to updating these message node distributions according to the adjacent message nodes (that is, message nodes sharing a common check node). Such a decoding algorithm actually works for any linear code, but quickly becomes intractable as the degree of check nodes increases. In the case of LDPC codes, this degree is small (by definition) which makes the algorithm run efficiently. In our particular context where check nodes have degree 3, information from adjacent nodes can further be exploited through the convolution:

$$P_{a,c} * P_{b,c}(\delta) \stackrel{\text{def}}{=} \sum_{\alpha \in \mathbb{F}_{256}} P_{a,c}(\delta) P_{b,c}(\delta \oplus \alpha).$$

In addition, the corresponding graph has small cycles and the propagation is very fast (after only two iterations, a position has been influenced by all others). Hence, the number of iterations of the `while` loop can be considered as a constant. Note that the convolution of two probability tables can be computed using a fast Walsh transform. Indeed, for a field of q elements, the q convolutions can be computed in $\Theta(q \ln q)$. Hence, Algorithm 2 has a complexity of $\Theta(q \ln q)$.

5 Experiments

We now present experiments obtained in three different settings. The first set of attacks targets a reference implementation of the AES and confirms the relevance of the tools we introduced. Following, a second set of attacks targeting an optimized implementation of AES (namely, the *furious* implementation from [15]) is presented. The main observation is that small code optimizations may lead

Algorithm 2. Proposition for `LDPCSoftDecoding` procedure

Input: The distributions $\Pr[\Delta K_{a,b} = \delta]$.
Output: The likeliest consistent system $\mathcal{S}$.
foreach $1 \leq a < b \leq 16$, $\delta \in \mathbb{F}_{256}$ **do**
  $P_{a,b}(\delta) \leftarrow \Pr[\Delta K_{a,b} = \delta]$;
while $\left(\underset{\delta}{\text{argmax}}\ P_{1,2}(\delta), \ldots, \underset{\delta}{\text{argmax}}\ P_{15,16}(\delta)\right)$ *is not a codeword* **do**
  foreach $1 \leq a < b \leq 16$ **do**
    foreach $\delta \in \mathbb{F}_{256}$ **do**
      $P_{a,b}(\delta) \leftarrow P_{a,b}(\delta) \cdot \prod_{c \notin \{a,b\}} P_{a,c} * P_{b,c}(\delta)$;
    $P_{a,b} \leftarrow \frac{P_{a,b}}{\|P_{a,b}\|_1}$;
return $\left(\underset{\delta}{\text{argmax}}\ P_{1,2}(\delta), \ldots, \underset{\delta}{\text{argmax}}\ P_{15,16}(\delta)\right)$;

to variations in S-boxes leakage functions, which in turn results in less efficient attacks. For these two first sets of attacks, we measured the power consumption of an Atmel microcontroller running the target AES implementations at 20MHz, by monitoring the voltage variations over a small resistor inserted in the supply circuit. We then conclude this paper by investigating a theoretical setting where leakage functions are not linear. This final experiment motivates the potential interest of collision attacks compared to other non-profiled distinguishers.

In the following, we compare collision attacks (Coll) using the Euclidean distance (ED) and the correlation-enhanced (CE) detection techniques, with the non-profiled variant of Schindler et al.'s stochastic approach [17], described in [8] and using a 9-element basis including the target S-boxes output bits. Collision attacks have been performed using the instantitation of `PreProcessTraces` procedure from [13], defined by (1). As mentioned in the introduction, and for all our experiments, we assumed that we were able to divide each leakage traces in 16 sub-traces corresponding to the 16 AES S-boxes. For the real measurements, the detection metrics were directly applied to these sub-traces, following the descriptions in the previous sections. As for the simulated ones in Section 5.3, we generated univariate leakages for each S-box execution, according to the hypothetical (linear and non-linear) leakage functions and a Gaussian noise.

5.1 Attacking the Reference Implementation

In this first experiment, we consider a favorable setting where each table look-up is performed with the same register (our ASM code provided in Appendix B). The goal is to emphasize the gain obtained from the LDPC formulation of the problem and the Bayesian extension. The 2^8-th order success rates (defined in [20]) obtained in this case are given in Figure 1. Original collision attacks directly extract the key from scores obtained with the ED or CE metrics. Attacks taking advantage of the LDPC decoder are marked with an (L), and the use of the

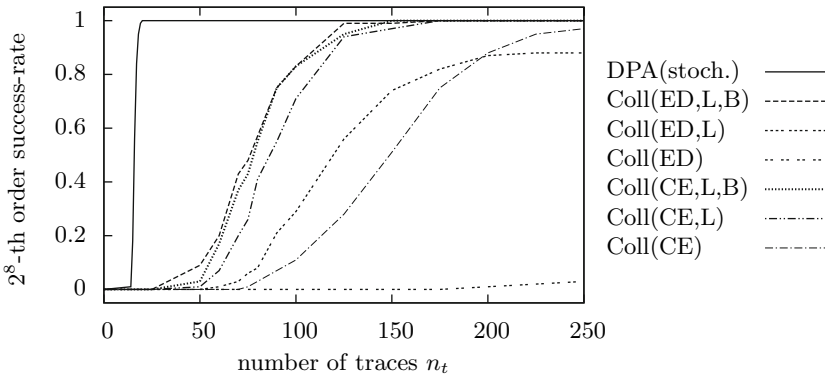

Fig. 1. Order 2^8 success rates of attacks using the homemade implementation

Bayesian extension is denoted by $(B)^2$. As expected, using the LDPC decoding algorithm greatly improves the attacks performances. Moreover, using the Bayesian extension also provides a non-negligible gain. Interestingly, when both tools are combined, ED and CE detection metrics seem to be equivalent in terms of data complexity. This may be a good empirical indication that the error correcting codes approach we propose really extracts all the available information.

5.2 Attacking the Optimized Implementation

In this next experiment, we targeted the AES *furious* implementation. This optimized implementation is a more challenging target, since the S-box layer and the ShiftRows operation are interleaved. Moreover, the table looks-up are performed from different registers. Due to these optimizations, the leakage functions of the different S-boxes are not so similar anymore (see Appendix B). Hence, the correct key is unlikely to correspond to the most likely codeword. A direct consequence of this more challenging context is that the success rate of order 2^8 is not suited to evaluate the attack performances (i.e. the correct key may be rated beyond the 2^8 first ones by the attack). As an alternative, we estimated the median rank of the correct key among the 2^{128} possible values.

In that case, one should use a decoding algorithm with $\ell > 1$ and test the $2^8 \cdot \ell$ corresponding keys afterwards. Unfortunately, the efficiency of such a list-decoding algorithm depends on the shape of distributions $\Pr[\Delta K]$, themselves being highly dependent on the similarity metrics used to compare traces, and the device running the cipher. Therefore, we left the design of such an algorithm as a scope for further research. Yet, and in order to be able to analyze the attack performances, we used an ad-hoc list-decoding algorithm that consists in enumerating key classes from a subset of 15 positions of dimension 15 (using the algorithm in [21]), for which the corresponding distributions have a small

[2] Since the Bayesian extension does not modify the ordering of the scores, using it only makes sense when applying the LDPC decoding algorithm.

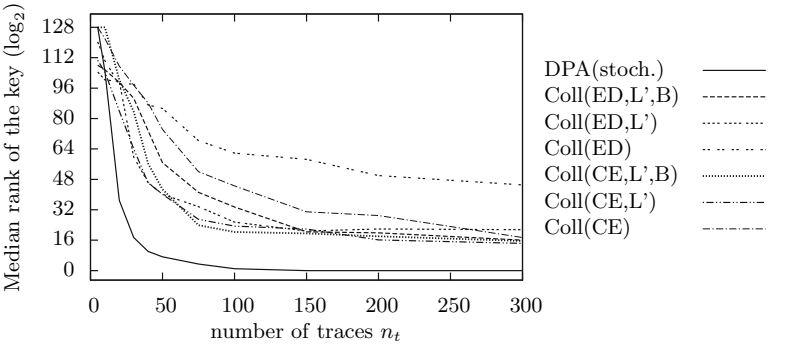

Fig. 2. Estimates of the median key rank when attacking the *furious* implementation

entropy. Note that the belief propagation part of the decoder in Algorithm 2 can be used (or not) before performing the enumeration. To avoid confusion, we will denote by (L') the use of this belief propagation step before enumerating. Computing the median rank of the key also becomes intensive as this rank becomes large. Hence, we decided to enumerate keys up to the 2^{20}-th first ones, and in the cases where the correct key was not found, estimated the key rank by multiplying correct subkey ranks. These heuristics naturally have to be taken into consideration when analyzing the results in Figure 2. However, we believe that they provide a fair understanding of the different attacks we investigated. Namely, as in Figure 1, the soft-decoding algorithm allows great performance improvements in the *furious* implementation case-study. By contrast, the Bayesian extensions were less directly useful. This observation again relates to the different leakage models observed for different S-boxes. As the parameter estimation in the Bayesian extensions requires a sufficient precision to be exploitable, they were only useful after approximately 150 traces in this more challenging scenario. Note finally that most collision attacks are stuck around rank 2^{15}. This can be explained by one of the S-boxes leaking in a drastically different way than the others in our implementation. As a result, we were only able to recover 14 bytes out of the 15 ones from this optimized implementation (even with large number of measurements). This leads to a median rank of roughly 2^7 for the correct system, an a median rank $2^{15=7+8}$ for the correct master key.

5.3 Simulated Experiments with Non-linear Leakages

In the previous experiments, the non-profiled variant of Schindler et al.'s stochastic approach consistently gave better results than the (improved) collision attacks investigated. As a result, one can naturally question the interest of such attacks in a security evaluation context. In order to discuss this issue, this final section analyzes the relevance of collision attacks in a purely theoretical setting. We used two different sets of simulated traces for this purpose: the first ones were generated using a leakage function of which the output is a linear function of the S-boxes output bits; the second ones were generated with a leakage function of

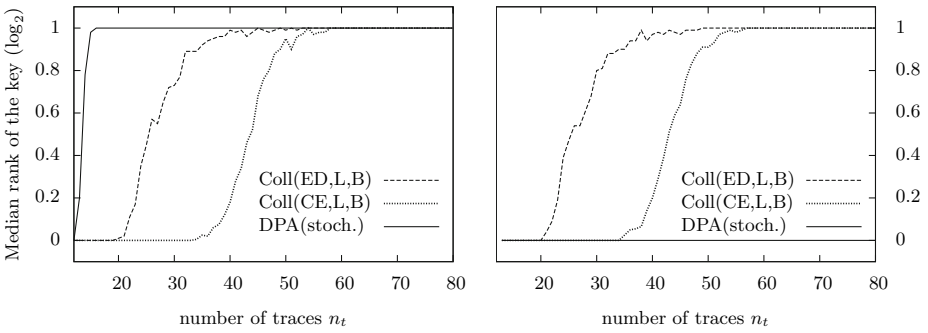

Fig. 3. 2^8-th order success rate for linear (left) and non-linear (right) leakage functions

which the output is a highly non-linear function of the S-boxes output bits (i.e. a situation emulating the worst case scenario from [22]). The results corresponding to these alternative scenarios are in Figure 3. As expected, the linear leakages in the left part of the figure are efficiently exploited by all attacks, with an improved data complexity for the stochastic approach. By contrast, in the right part of the figure, the stochastic approach is unable to exploit the non-linear leakages, and only collision attacks lead to successful key recoveries[3]. This confirms that there exist situations in which non-profiled collision attacks are able to exploit information leakage that no other non-profiled attack can. We leave the quest for such leakage functions (or protected circuits) as a scope for further research.

Acknowledgements. We would like to thank Jean-Pierre Tillich for providing useful information and advices about the decoding of LDPC codes.

References

1. Bennata, A., Burshtein, D.: Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels. IEEE Transactions on Information Theory 52, 549–583 (2006)
2. Bogdanov, A.: Improved Side-Channel Collision Attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
3. Bogdanov, A.: Multiple-Differential Side-Channel Collision Attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008)
4. Bogdanov, A., Kizhvatov, I.: Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. IEEE Transactions on Computers 61(8), 1153–1164 (2012)
5. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

[3] Increasing the basis with non-linear elements would not allow solving this issue as long as only non-profiled attacks are considered. It would lead to more precise leakage models both for the correct key candidates and the wrong ones, by over-fitting.

6. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
7. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved Collision-Correlation Power Analysis on First Order Protected AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 49–62. Springer, Heidelberg (2011)
8. Doget, J., Prouff, E., Rivain, M., Standaert, F.-X.: Univariate side channel attacks and leakage modeling. Journal of Cryptographic Engineering 1(2), 123–144 (2011)
9. Gallager, R.G.: Low density parity check codes. Transactions of the IRE Professional Group on Information Theory IT-8, 21–28 (1962)
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
11. Ledig, H., Muller, F., Valette, F.: Enhancing Collision Attacks. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 176–190. Springer, Heidelberg (2004)
12. Mangard, S.: Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004)
13. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
14. Moradi, A.: Statistical Tools Flavor Side-Channel Collision Attacks. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 428–445. Springer, Heidelberg (2012)
15. Poettering, B.: Fast AES implementation for Atmel's AVR microcontrollers, `http://point-at-infinity.org/avraes/`
16. Renauld, M., Kamel, D., Standaert, F.-X., Flandre, D.: Information Theoretic and Security Analysis of a 65-Nanometer DDSLL AES S-Box. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 223–239. Springer, Heidelberg (2011)
17. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
18. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES: Combining Side Channel and Differential-Attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
19. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
20. Standaert, F.-X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
21. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. Cryptology ePrint Archive, Report 2011/610 (2011), `http://eprint.iacr.org/2011/610`
22. Veyrat-Charvillon, N., Standaert, F.-X.: Generic Side-Channel Distinguishers: Improvements and Limitations. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 354–372. Springer, Heidelberg (2011)

A Proof of Lemma 2

We want to express the mean and the variance of a mixture of two distributions as a function of the means and variances of these distributions. Let us recall that $\mathcal{D}$ is a mixture of two distributions $\mathcal{D}_c$ and $\mathcal{D}_{nc}$ with respective weights 2^{-8} and $1-2^{-8}$. We denote by μ and σ^2 the expected value and variance of $\mathcal{D}$, by (μ_c, σ_c^2) the expected value and variance of $\mathcal{D}_c$, and by $(\mu_{nc}, \sigma_{nc}^2)$ the expected value and variance of $\mathcal{D}_{nc}$. Let X_c and X_{nc} be respectively drawn according to $\mathcal{D}_c$ and $\mathcal{D}_{nc}$ and X be the mixture $2^{-8}X_c + (1-2^{-8})X_{nc}$. Then, due to the linearity of the operator, $\mathbb{E}(X) = \mathbb{E}\left(2^{-8}X_c + (1-2^{-8})X_{nc}\right) = 2^{-8}\mu_c + (1-2^{-8})\mu_{nc}$. Thus, it follows that $\mu_{nc} = \frac{\mu - 2^{-8}\mu_c}{1-2^{-8}}$. Concerning the variance, a slightly more difficult calculus leads to the claimed result. First, we use the relationship $\mathbb{V}(X) = \mathbb{E}\left(X^2\right) - \mathbb{E}(X)^2$ and we develop its first term in:

$$\mathbb{E}\left(X^2\right) = \mathbb{E}\left(2^{-16}X_c^2 + 22^{-8}(1-2^{-8})X_cX_{nc} + (1-2^{-8})^2X_{nc}^2\right).$$

Since X_c and X_{nc} are independent variables, we have:

$$\mathbb{E}\left(X^2\right) = 2^{-16}\mathbb{E}\left(X_c^2\right) + 2^{-7}(1-2^{-8})\mathbb{E}(X_c)\,\mathbb{E}(X_{nc}) + (1-2^{-8})^2\mathbb{E}\left(X_{nc}^2\right).$$

We then notice that $\mathbb{E}\left(X_c^2\right) = \sigma_c^2 + \mu_c^2$ (the same holds for $\mathbb{E}\left(X_{nc}^2\right)$). Hence:

$$\mathbb{E}\left(X^2\right) = 2^{-16}(\sigma_c^2 + \mu_c^2) + 2^{-7}(1-2^{-8})\mu_c\mu_{nc} + (1-2^{-8})^2(\sigma_{nc}^2 + \mu_{nc}^2).$$

Now returning to $\mathbb{V}(X) = \mathbb{E}\left(X^2\right) - \mathbb{E}(X)^2$, we finally observe that many terms in μ vanish to yield $\mathbb{V}(X) = 2^{-16}\sigma_c^2 + (1-2^{-8})^2\sigma_{nc}^2$.

B Additional Details about the Target Implementations

Our experiments are based on two different implementations of the AES: a reference one that has been designed such that S-boxes looks-ups have similar leakage functions, and the *furious* implementation. They respectively execute the AES S-box in four instructions (`mov SR,ST22`; `mov ZL,SR`; `lpm SR,Z`; `mov ST22,SR`) and three instructions (`mov H1,ST21`; `mov ZL,ST22`; `lpm ST21,Z`). We observed that the most leaking operation in the S-box computation was the `mov` operation, that stores the input of the S-box in the `ZL` register. Hence, in the reference implementation, we first copy intermediate values in a register `SR`, that is the same for all 16 S-boxes computations. Then, `SR` is updated and the output is copied back to the initial state register. On the contrary, we can see that in the *furious* implementation, the `ZL` register is directly updated from the state register (here `ST22`), and the answer directly goes back to the state register. In addition, the *furious* implementation combines the S-box layer with the `ShiftRows` operation. It explains why the output is stored in `ST21` and not `ST22`. The optimizations in the *furious* implementation are the main reason of the poor results of the attacks performed. As a simple illustration, we plotted the templates of the leakage points used in the attacks for different S-boxes in Figure 4 (for the reference implementation) and Figure 5 (for the *furious* one).

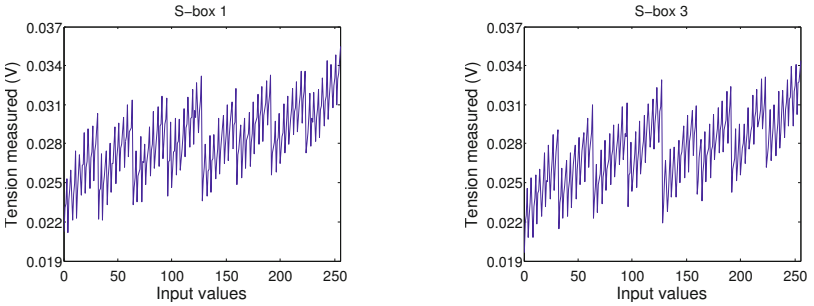

Fig. 4. Leakage functions for the reference implementation

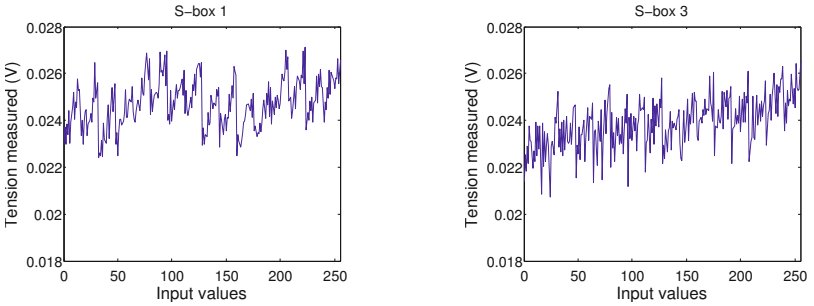

Fig. 5. Leakage functions for the *furious* implementation

C About Time Complexity

Metrics used in Section 5 for analyzing experiments only consider the success rate of the attacks as a function of their data complexity. We consider the time complexity of the proposed collision attack in this section. When attacking n_s S-boxes processing n_b-bit words, these complexities for our different procedures are:

`ComputeStatistics`	$O\left(n_s^2 n_t^2 \ell\right)$
`ExtractDistributions`	$O\left(n_s^2 (n_t^2 + 2^{n_b})\right)$
`LDPCDecode`	$O\left(n_s^2 n_b 2^{n_b}\right)$

When using the pre-processing technique that has a cost $\Theta\left(n_s n_t \ell\right)$, the complexity of the procedure `ComputeStatistics` is decreased to $O\left(n_s^2 2^{2n_b} \ell\right)$. Hence, it turns out that, in realistic contexts, collision attacks can be performed in a negligible time compared to the on-line acquisition and the final key search phases (a similar comment applies to stochastic attacks). Furthermore, by carefully profiling the number of cycles needed to perform the different steps of the attacks, we observed that the slight time overhead induced by the use of a Bayesian extension and/or an LDPC decoding algorithm is positively balanced by the reduction of the data complexity, hence leading to globally more efficient attacks.