J Cryptogr Eng (2013) 3:45–58
DOI 10.1007/s13389-013-0051-9

CHES 2012

Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version

Benoît Gérard · François-Xavier Standaert

Received: 15 November 2012 / Accepted: 15 January 2013 / Published online: 20 February 2013

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 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 improve 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 differential power analysis (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.

Part of this work has been done as a 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).

B. Gérard (✉) · F.-X. Standaert
UCL Crypto Group, Université catholique de Louvain,
Place du Levant 3, 1348, Louvain-la-Neuve, Belgium
e-mail: a-b.gerard@orange.fr

B. Gérard
DGA-MI, Bruz, France

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. [11] differential power analysis (DPA), Brier et al. (CPA) [5] correlation power analysis and Chari et al. [6] template attacks (TA) 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. [21] at FSE 2003 are among the most investigated solutions for this purpose. While initially dedicated to the DES, they have then been applied to the AES [20] and improved in different directions over the last years, as witnessed by the recent works of Ledig et al. [12], Bogdanov [2–4], Moradi et al. [15,16] and Clavier et al. [7].

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 [25], e.g. in the case of devices with strongly non-linear leakage functions. Hence, although the existence of such devices remains an open question [18], 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. [15].

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 et al. [19] stochastic approach, 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 where the leakage function is highly non-linear (i.e. in the pathological example from [25]), collision attacks lead to successful key recoveries whereas all non-profiled attacks fail.

Finally, note that this paper is an extended version of the work presented at CHES 2012 [10] that contains additive features concerning list-decoding. More precisely, we provide more details about using a decoding algorithm that returns more than a single codeword (in order to trade data for computations for instance). We have been able to experimentally compare different set of parameters for the proposed procedure due to a recently published algorithm for key rank estimation [24]. The proposed procedure is detailed in Sect. 4.4, corresponding experiments can be found both in Sect. 5.2 and Appendix C.

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:

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

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 first-round S-box computations $t^{(i)} \stackrel{\text{def}}{=} (t_1^{(i)}, \ldots t_{16}^{(i)})$. Each sub-trace is again composed of m points (or samples). Hence, the sub-trace corresponding to the a-th S-box will be denoted as $t_a^{(i)} \stackrel{\text{def}}{=} (t_{a,1}^{(i)}, \ldots, t_{a,m}^{(i)})$. 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 [20] or Pearson's correlation coefficient [21], 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. [15]. 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 [15] combined their attack with a pre-processing of the traces, that consists in building "on-the-fly" templates of the form:

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

That is, traces corresponding to an a-th S-box input of value x are averaged to form the template trace $\bar{t}_a^{(x)}$. 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 [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, a collision between S-boxes a and b provides information on the variable:

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

Next, observe that the system composed of these 120 relations is of rank 15 that is the vector $\Delta K \stackrel{\text{def}}{=} (\Delta K_{1,2}, \Delta K_{1,3}, \ldots, \Delta K_{15,16})$ only determines the key K up to 2^8 equivalent keys. 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 from a noisy vector ΔK. 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 Sect. 5, this combination of LDPC decoding and Bayesian statistics can indeed lead to very efficient attacks.

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.

```
(t̄_1, ..., t̄_16) ← PreProcessTraces(x^(1), ..., x^(n_t), t^(1), ..., t^(n_t));
foreach 1 ≤ a < b ≤ 16 do
    S_{a,b} ← ComputeStatistics(t̄_a, t̄_b);
Pr[ΔK] ← ExtractDistributions(S_{1,2}, ..., S_{15,16});
{S_1, ..., S_ℓ} ← LDPCDecode(Pr[ΔK]);
foreach system S_i and key candidate k compatible with equations in S_i do
    if TestKey(k) then
        return k;
return failure;
```

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 [3,15] will be discussed in Sect. 4.1. Next, for each pair of S-boxes, the vector $S_{a,b} \stackrel{\text{def}}{=} \left(S_{a,b}(\delta)\right)_{\delta \in \mathbb{F}_{256}}$ containing scores for the 256 possible values δ of the variable $\Delta K_{a,b}$ is extracted (`ComputeStatistics` procedure). Different techniques have again been proposed for this purpose in the literature (see Sect. 4.2 for some examples). 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 Sect. 4.2, this can be emulated by normalizing scores, or obtained 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 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 Sect. 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 [2–4] attacks take advantage of averaging (by measuring the power consumption of the same plaintext several times), in order to reduce the measurement noise. Similarly, Moradi et al. [15] 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 Sect. 5 will exploit it.

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 Sect. 3.1, one can emulate those distributions 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 *a posteriori* probabilities $\Pr\left[\Delta K_{a,b} = \delta \middle| S(a,b,\delta)\right]$. 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.

4.2.1 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:*

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

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 posterior distribution of $\Delta K_{a,b}$ can be expressed using only $\Pr_{\mathcal{D}_c}[s_i]$ and $\Pr_{\mathcal{D}_{nc}}[s_i]$ as stated in the following Lemma.

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

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

where $\propto$ stands for "proportional to" and $\Pr_{\mathcal{D}_c}[s_i]$ (resp. $\Pr_{\mathcal{D}_{nc}}[s_i]$) denotes the probability of the statistic to be equal to s_i when resulting from the comparison between to identical (resp. different) inputs. Moreover, if for any i, $\Pr_{\mathcal{D}_{nc}}[s_i]$ is non-zero, then:

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

Proof The first line is a direct application of Bayes' relation: $\Pr\left[\Delta K_{a,b}\middle|\Sigma\right]=\frac{\Pr\left[\Sigma\middle|\Delta K_{a,b}\right]\Pr[\Delta K_{a,b}]}{\Pr[\Sigma]}$. Indeed, observing that $\Pr\left[\Delta K_{a,b}\right]$ and $\Pr[\Sigma]$ are positive constants and that observed statistics are supposed independent allow to conclude. The second line results from Model 1, and the final formula is obtained dividing by $\prod_i \Pr_{\mathcal{D}_{nc}}[s_i]$. □

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 $\mathcal{D}_{nc}$:*

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

4.2.2 Specialization to the Euclidean distance detection

The Euclidean distance (ED) has been proposed as a detection tool in [20] and investigated in a profiled setting in [4]. The Euclidean distance[1] between two traces T and T' equals:

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

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 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 $\mathrm{ED}\left(\bar{T}_a^{(i_a)},\bar{T}_b^{(i_b)}\right)$ as the score to normalize:

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

where d_0 is chosen such that all values

$$d_0-\max_{x_a^{(ia)}\oplus x_b^{(i_b)}=\delta}\mathrm{ED}\left(\bar{T}_a^{(i_a)},\bar{T}_b^{(i_b)}\right)$$

are strictly positive. Note that if a single trace is given, only a single Euclidean distance can be computed between each

[1] Actually this corresponds to the square root of the Euclidean distance but the root does not alter the ordering hence there is no reason to consider it.

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. In that case we have to consider all these distances to obtain a single score. We made the heuristic choice of using the max function to select which Euclidean distance will be retained to compute the scores. We have no other justification than the fact that this was the one that has shown best performances among different metrics (excluding the Bayesian one of course). Let us now consider the application of the Bayesian framework to the use of ED. We will use $\mathrm{ED_B}$ 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 [19] 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 Sect. 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:

$$\mathrm{ED}_B(T, T') \stackrel{\mathrm{def}}{=} \sum_{j=1}^{m} \frac{(T_j - T'_j)^2}{2\sigma_j^2}, \tag{3}$$

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 $\mathcal{D}_{\mathrm{c}}$ is a χ^2 distribution with m degrees of freedom. In the case of $\mathcal{D}_{\mathrm{nc}}$, 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 $\mathcal{D}_{\mathrm{c}}$ using Lemma 2. Experiments show that the shape of $\mathcal{D}_{\mathrm{nc}}$ quickly tends towards a Gaussian distribution when increasing the number of points ℓ. Combining these observations, we obtain the following score:

$$\begin{aligned}\mathrm{BayExt}(S_{\mathrm{ED}}(a,b,\delta)) &\stackrel{\mathrm{def}}{=} \mathrm{Norm}\left(\exp\left[\sum_{x_a^{(i_a)} \oplus x_b^{(i_b)} = \delta} \frac{\left(\mathrm{ED_B}\left(\bar{T}_a^{(i_a)}, \bar{T}_b^{(i_b)}\right) - \mu_{nc}\right)^2}{2\sigma_{nc}^2}\right.\right. \\ &\qquad \left.\left. - \mathrm{ED_B}\left(\bar{T}_a^{(i_a)}, \bar{T}_b^{(i_b)}\right)\right]\right). \end{aligned} \tag{4}$$

Remark If averaging of traces is 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 $\bar{T}_a^{(i_a)}$.

4.2.3 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:

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

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 [15], 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 $S_{\mathrm{CE}}(a, b, \delta) \stackrel{\mathrm{def}}{=} \mathrm{Norm}\left(\rho\left(\bar{T}_a, \bar{T}_b^{\oplus\delta}\right)\right)$.

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 [14]:

$\mathrm{CE_B}(a,b,\delta) \stackrel{\text{def}}{=} \operatorname{arctanh} S_{\mathrm{CE}}(a,b,\delta)$. Asymptotically, the random variables $\mathrm{CE_B}(a,b,\delta)$ 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:

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

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:

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

Again, results in Sect. 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

First of all, notice that no efficient exact maximum-likelihood decoding algorithms are available in our context. That is, algorithms that would ensure that the returned codeword is the most likely one would have prohibitive costs compared to "approximated" (yet relevant) solutions as the one we consider here.

Such a soft-decoding algorithm for non-binary LDPC codes can be found in [1] and is presented in Algorithm 1. 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:

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

The only proofs of convergence of this algorithm that can be found are for graphs having no cycles or class of random graphs. They cannot be applied to our context where the graph is fixed and has many small cycles. Nevertheless, for all our experiments (presented in Sect. 5) the algorithm has converged in a relatively small number of iterations (at most 10 in a very worst case). 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 1 has a complexity of $\Theta(q \ln q)$.

Algorithm 2: Proposition for `LDPCDecode` procedure.

Input: The distributions $\Pr\left[\Delta K_{a,b} = \delta\right]$.
Output: The likeliest consistent system $\mathcal{S}$.
foreach $1 \le a < b \le 16$, $\delta \in \mathbb{F}_{256}$ **do**
  $P_{a,b}(\delta) \leftarrow \Pr\left[\Delta K_{a,b} = \delta\right]$;
while $\left(\operatorname{argmax}_\delta P_{1,2}(\delta), \ldots, \operatorname{argmax}_\delta P_{15,16}(\delta)\right)$ *is not a codeword* **do**
  foreach $1 \le a < b \le 16$ **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 $\left(\operatorname{argmax}_\delta P_{1,2}(\delta), \ldots, \operatorname{argmax}_\delta P_{15,16}(\delta)\right)$;

Note that this algorithm returns a single codeword where the `LDPCDecode` procedure defined in Algorithm 1 may return $\ell > 1$ codewords. This problematic of list decoding is addressed in the following section.

About non-linear collisions Collision attacks can be enhanced by considering not only the first round but the second round too. In that case relations between key bytes are not linear anymore but still the main structure of Algorithm 2 can be used. Indeed, this Belief Propagation algorithm applies to any kind of constraints over data bytes. The point is that the distribution update that is performed using a convolution in Algorithm 2 would then require heavier computations. More precisely, the general complexity of such update over $\mathbb{F}_q$ for a check node of degree d is $\Theta(q^{d-1})$. We leave the investigation of check-node degrees obtained when considering more than the first round and the application of such decoding algorithm to this context to further research.

4.4 About list decoding

The problem of soft list-decoding of non-binary LDPC codes has not triggered so much attention in the coding community. More generally, list-decoding for large values of ℓ is far from transmission preoccupations. We hence have no on-the-shelf decoding algorithm to propose.

Such list-decoding procedure with large list size is indeed of interest for cryptanalytic purposes. The reasons why a cryptanalysis could benefit from such algorithm is twofold. First, it enables trade-offs between data complexity and off-line computations. Second, and as we will see in Sect. 5, in some situations, the correct codeword will be among the most likely but not the most likely one whenever billions of traces are used. In such case where first-order success rate is stuck to 0 but for a small o, the o-th order success rate is close to 1, being able to test other likely keys is of great interest.

In this section we discuss a solution for the attacker to enumerate codewords (hence keys) in a relevant order (that is an order close to the decreasing probability order). This solution combines the base idea of information set decoding and a key enumeration algorithm provided in [23].

Two different scenarios may be considered here.

(i) The attacker wants to recover the key whenever he would have to test all possible keys.
(ii) The attacker plans to test at most 2^8 ℓ keys.

In the first scenario we are less interested in the set of codewords returned by the algorithm (that is all codewords) than in the ordering of these codewords. Technically, the ordered list of codewords cannot be stored and hence the decoding algorithm should enumerate codewords and test the corresponding keys on-the-fly. Such algorithm should enumerate codewords in an order as close as possible from the likelihood-decreasing order and possibly without repetition. This problem is precisely the one addressed in [23] in the particular context where positions are independent (that is a context without redundancy). Adapting this algorithm to redundant situation is possible but would induce an intractable overhead in our setting.

The solution proposed here is to restrict ourselves to 15 positions of the code such that two distinct codewords have two distinct truncated 15-byte values (in coding theory, such a set is named *information set*). Doing so we can take profit of the algorithm in [23] to enumerate key classes (equivalently codewords) during an attack and also take profit of the rank estimation procedure of [24] in our experiments (that does not handle redundancy). Once an information set is chosen, we enumerate codewords (and test the corresponding 2^8 keys) until the correct one is found. The drawback of such technique is that while the algorithm in [23] ensures that keys are enumerated in decreasing order of probabilities, this property does not hold here since we only consider 15 out of 120 positions. Indeed, the probability of a codeword is computed based on the product of the value probabilities for all positions of the code and not only 15 and thus the ordering induced by those 15 positions does not necessarily correspond to the actual one.

The second scenario is closer to reality in the sense that keys may have limited lifetime (session keys for instance) and thus attackers should have a limited time to perform the attack. We assume that the attacker has enough memory to store the list of ℓ codewords (if not the case he should take a smaller ℓ or switch to context (i)). As we previously explained, no efficient exact list-decoding algorithm is available. A solution is to consider a subset of codewords and to return the ordered list of the ℓ most likely. Contrarily to the approach proposed for scenario (i), we ensure that the first returned codeword is more likely than the second since we compute the probabilities using all the available positions. Nevertheless, since considering a subset of codewords we might miss likely codewords and possibly the correct one.

The solution proposed for scenario (ii) is actually based on the idea of information set decoding. While the contexts are really different the basic idea of information set decoding, that can be summed-up as "performing many bounded decoding over different smaller codes", applies here. Indeed, a way of considering a subset of relevant codewords is to consider different information sets, perform bounded list-decoding using the algorithm of [23] and store codewords in an updated structure containing the ℓ most likely codewords encountered so far.

Notice that the solution proposed for scenario (i) actually is a particular case of the solution (ii) where a single information set is used and ℓ is set to 256^{15}. We summarize the algorithm proposed.

Algorithm 3: Proposed list decoding algorithm.

Input:
- distributions $\Pr\left[\Delta K_{a,b} = \delta\right]$
- a number of iterations N

Output: The list $\mathcal{L}$ of the ℓ likeliest codewords found.

$I_0 \longleftarrow$ information set;
$\mathcal{L} \longleftarrow$ the ℓ most likely codewords on I_0;
for $1 \leq i \leq N$ **do**
  $I_i \longleftarrow$ new information set;
  $c \longleftarrow$ most likely codeword on I_i;
  while c *is more likely than* $c_{\min} \stackrel{\text{def}}{=} \operatorname{argmin}\ \mathcal{L}$ **do**
    Change $c_{\min}$ for c in $\mathcal{L}$;
    $c \longleftarrow$ next most likely codeword on I_i;
return $\mathcal{L}$;

Different parameters should be adjusted depending on the context:

- the number of different information sets considered ($N+1$ in the algorithm);
- the way information sets are chosen;
- the application of belief propagation steps before/during the process.

This second approach is only meaningful if the correct key rank has a large variability depending on the chosen information set. This will depend on the parameters used for the decoding procedure (a discussion on these parameters can be found in Appendix C). We observed in our experiments that the key rank was not varying so much when carefully choosing information sets (typically at most a factor 2^5).

Evaluation and context (ii) We would like to emphasize that Algorithm 3 is not as suited for evaluation than the approach (i). Using such an algorithm, an evaluator can only estimate complexities of attacks he is able to mount (and this will be computationally intensive). On the contrary, approach (i) allows the use of the estimation algorithm in [24] to estimate key ranks beyond the computational capabilities of the evaluator. Hence, even in the case where the information set choice heavily influences the key rank, and for evaluation purpose, we suggest to use approach (i) on different information sets then taking as final rank the minimum observed correct key rank.

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 [17]) is presented. The main observation is that small code optimizations may lead 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 20 MHz, 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 [19] (DPA stoch.), described in [8] and using a 9-element basis including the target S-boxes output bits. Collision attacks have been performed using the instantiation of `PreProcessTraces` procedure from [15], defined by (1). As mentioned in Sect. 1, and for all our experiments, we assumed that we were able to divide each leakage trace 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 Sect. 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.

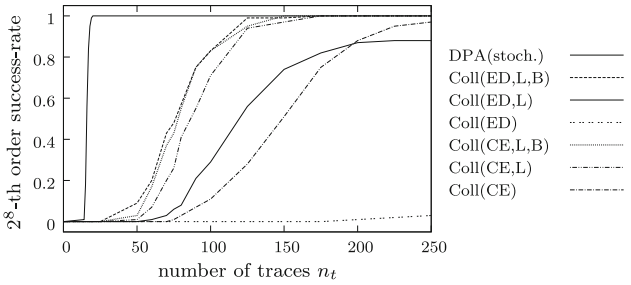

Fig. 1 Order 2^8 success rates of attacks using the homemade implementation

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 [22]) obtained in this case are given in Fig. 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 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

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

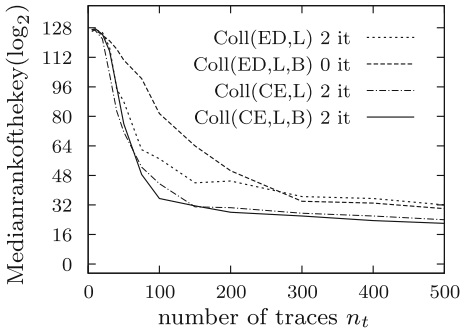

Fig. 2 Median key rank for *furious* implementation (best parameters)

challenging target, since the S-box layer and the ShiftRows operation are interleaved. Moreover, the table look-ups 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. This has been done by running the key rank estimation of [24] on the 15 distributions corresponding to the positions of the chosen information set. Within a few seconds we obtain very precise bounds on the key rank that we translate into a key rank distribution from which we extract the median rank.

Remark Notice that this way we obtain the key median rank corresponding to a particular choice of information set and not the actual rank. Nevertheless, the attacker himself should not be able to enumerate keys in a decreasing order.

We discuss both the choice of the information set and the application of the belief propagation in Appendix C.

Figure 2 shows results obtained using the best parameters for the four similarity metrics (namely the standard and Bayesian versions of both Euclidean distance and correlation coefficient). They all correspond to choosing the information set according to posterior entropies (the number of information sets being given in legend).

In this particular context, the models on which Bayesian extensions are derived does not hold. Nevertheless, we observe that the Bayesian extensions still provide a non-negligible advantage compared to standard metrics.

As expected, since two S-boxes have drastically different behaviors than others, the attacks are stuck to 2^{23}. Intuitively we have no information about two of the 15 bytes determined by the attack plus one byte due to the rank of the linear-collision systems. Thus, we are somehow exhaustively searching a key among $2^{24=3\cdot 8}$ what explains the median rank of 2^{23}.

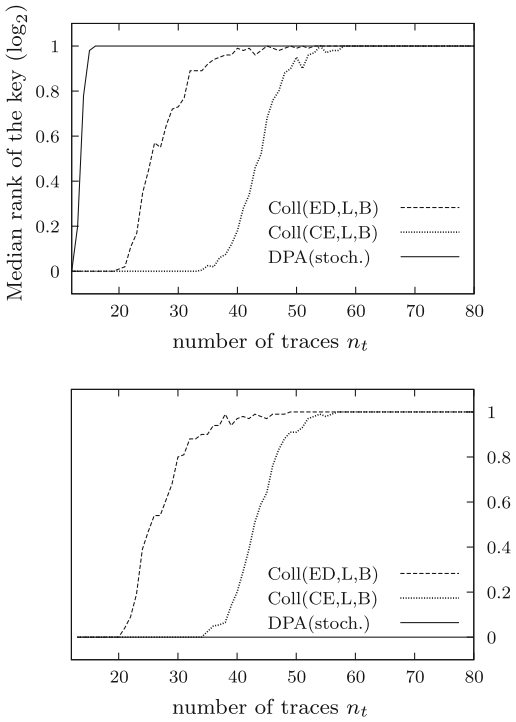

Fig. 3 2^8-th order success rate for linear (*top*) and non-linear (*bottom*) leakage functions

5.3 Simulated experiments with non-linear leakages

In the previous experiments, the non-profiled variant of Schindler et al. 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 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 [25]). The results corresponding to these alternative scenarios are in Fig. 3. As expected, the linear leakages in the top part of the figure are efficiently exploited by all attacks, with an improved data complexity for the stochastic approach. By contrast, in the bottom 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 colli-

[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.

sion 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.

6 Conclusion

LDPC codes provide a natural way to describe side-channel collision attacks. In this paper, we first propose a general framework for collision attacks based on this LDPC representation of the problem. We then show that previous works can naturally be expressed using this framework, derive Bayesian extensions to scores classically used as collision detection metrics.

We also propose the use of two LDPC decoding algorithms for recovering the key. One can be found in coding theory literature while the other is a homemade list-decoding procedure based on the underlying idea of information set decoding.

The relevance of the introduced tools is confirmed by performing attacks against two different implementations of the AES in a low-cost microcontroller. In addition, we observed that simple code modifications (optimizations) can make the detection of collisions challenging in software implementations. Such situations, and more generally the need of enabling trade-offs between number of traces and off-line computations, stress the importance of list-decoding with large list sizes in cryptanalysis. We suggest further research in this direction since the proposed algorithm is rather heuristic and since only few parameters have been investigated in the present work.

Eventually, we discussed the relevance of such collision attacks in security evaluations. In particular, we exhibited that they can be necessary in situations where profiling the target device is impossible and meaningful leakage models (needed for standard DPA to succeed) cannot be obtained by engineering intuition.

Appendix 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:

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

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 $\mathbb{V}(X) = 2^{-16}\sigma_c^2 + (1-2^{-8})^2\sigma_{nc}^2$.

Appendix 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' look-ups have similar leakage functions, and the *furious* implementation. Codes corresponding to one look-up are presented in Table 1 and commented below.

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

Table 1 S-box code for used AES implementations

Reference	Furious
`mov SR, ST22`	`mov H1, ST21`
`mov ZL, SR`	`mov ZL, ST22`
`lpm SR, Z`	`lpm ST21, Z`
`mov ST22,SR`	

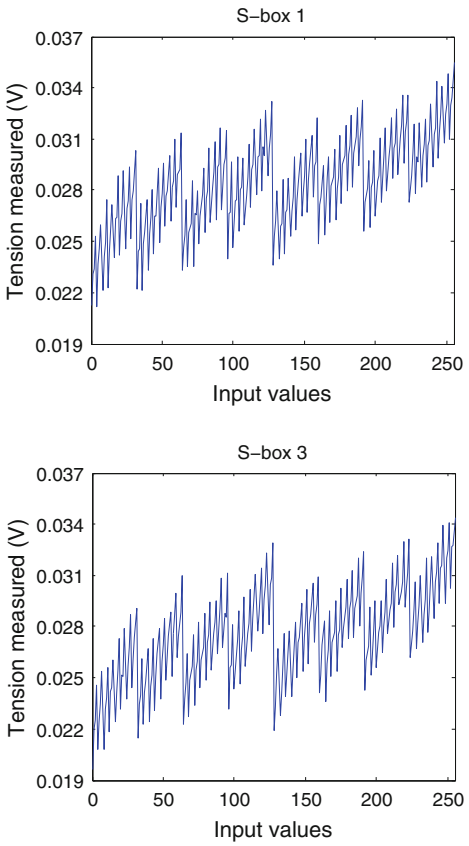

Fig. 4 Leakage functions for the reference implementation

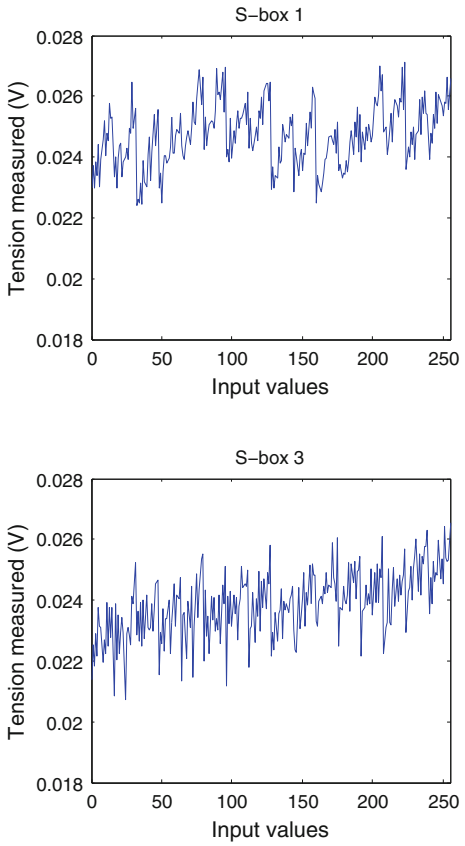

Fig. 5 Leakage functions for the *furious* implementation

illustration, we plotted the templates of the leakage points used in the attacks for different S-boxes in Fig. 4 (for the reference implementation) and Fig. 5 (for the *furious* one).

Appendix C: Discussion on parameters for list decoding

In this section we discuss two parameters for the list decoding procedure presented in Sect. 4.4 namely the application of belief propagation and the information set choice.

Using belief propagation in the procedure Choosing an information set and using it to enumerate codewords seems to be not optimal since we actually do not benefit from information contained in other positions (redundancy). Note that this is not obvious when different information sets are used and when the overall probability of a codeword is computed using the 120 positions. However, in the case we consider a single information set is used and codewords are enumerated according to a partial probability computed from the 15 chosen positions.

To take profit of redundancy in this situation, the application of the belief propagation step should be used to propagate information according to the linear constraints of the code. The question is then how many iterations should we perform?

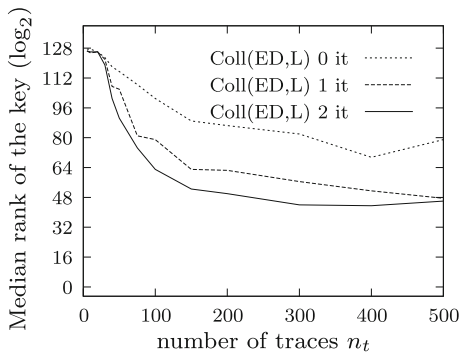

Fig. 6 Median ranks using Euclidean distance metric with different number of belief propagation steps

We performed experiments for 0 (no application), 1 and 2 iterations and obtained results in favor of the application of the belief propagation. We do not provide results obtained for correlation coefficient since in both standard and Bayesian cases applying twice the propagation step led to better results. This is not so clear for Euclidean distance as can be seen in Figs. 6 and 7.

Indeed, we see that for the Bayesian extension of the Euclidean distance, performing belief propagation alters results as soon as the number of traces are available reaches

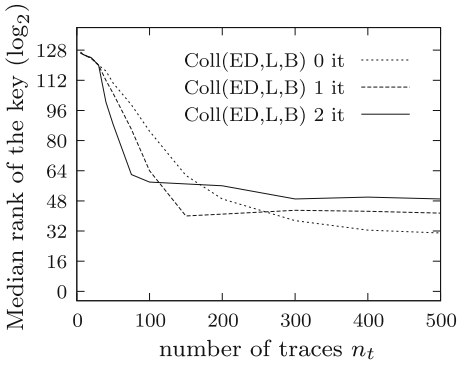

Fig. 7 Median ranks using Bayesian Euclidean distance metric with different number of belief propagation steps

some point (between 100 and 250 traces depending on the number of iterations considered).

Such degradation of performances may be due to the fact that in the context of *Furious* implementation the base hypothesis from which we built the Bayesian extension is not fulfilled anymore. The extension for correlation coefficient seems to be not (or at least less) impacted. This difference may comes from the nature of hypotheses: for Euclidean distance an hypothesis is made for the behavior of (infrequent and informative) colliding events while for correlation coefficient the hypothesis refers to (less informative) non-colliding events.

Choice of the information set The second important point in the procedure is the choice of the information set. Ideally we would like to chose the least faulty positions.

One may assume that when a position is faulty (due to the lack of information) the corresponding distribution looks "more uniform" than a position where the correct value has a good rank. This heuristic can be partially confirmed as positions depending on the two particular S-boxes have a larger entropy than others.

This remark implies that to overcome the problem encountered when attacking the *Furious* implementation, one could try to detect such error prone S-boxes, remove them from the analysis and use the single output decoding algorithm.

Since a list-decoding algorithm also aims at trading data for computation time, the former remark does not mean that such algorithm is useless. Hence we investigated the use of entropy to choose the information set.

We previously saw that applying belief propagation improved attacks we ran. Thus, the question is should we consider entropy posterior or prior to this step? Intuitively, applying the belief propagation should correct errors or confirm good positions thanks to redundancy. Thus, it is expected that positions having small entropy after the belief propagation steps are less faulty than the ones before.

This is confirmed by experiments where the choice of information set with prior entropies shows median larger ranks with a typical (and significant) factor of 2^5.

Appendix D: About time complexity

Metrics used in Sect. 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`	$O\left(n_s^2 n_t^2 m\right)$
`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 $\Theta\left(n_s n_t m\right)$, the complexity of the procedure `Compute Statistics` is decreased to $O\left(n_s^2 2^{2n_b} m\right)$. 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.

We stress that no precise timing is given here because the core of the attacks performed took less than a single second on a traditional PC. Hence we consider that such information is not relevant regarding the time spent to read traces from the drive or to perform enumeration/rank estimation.

References

1. Bennata, A., Burshtein, D.: Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels. IEEE Trans. Inform. Theory **52**, 549–583 (2006)
2. Bogdanov, A.: Improved side-channel collision attacks on AES. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography-SAC 2007, vol. 4876 of LNCS, pp. 84–95. Springer, Heidelberg (2007)
3. Bogdanov, A.: Multiple-differential side-channel collision attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2008, vol. 5154 of LNCS, pp. 30–44. Springer, Heidelberg (2008)
4. Bogdanov, A., Kizhvatov, I.: Beyond the limits of DPA: combined side-channel collision attacks. IEEE Trans. Comput. **61**(8), 1153–1164 (2011)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2004, vol. 3156 of LNCS, pp. 16–29. Springer, Heidelberg (2004)
6. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B., Koç, C.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2002, vol. 2523 of LNCS, 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 pro-

tected AES. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2011, vol. 6917 of LNCS, pp. 49–62. Springer, Heidelberg (2011)
8. Doget, J., Prouff, E., Rivain, M., Standaert, F.-X.: Univariate side channel attacks and leakage modeling. J. Cryptogr. Eng. **1**(2), 123–144 (2011)
9. Gallager, R.G.: Low density parity check codes. Trans. IRE Prof. Group Inform. Theory IT. **8**, 21–28 (1962)
10. Gérard, B., Standaert, F.-X.: Unified and optimized linear collision attacks and their application in a non-profiled setting. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2012, vol. 7428 of LNCS, pp. 175–192. Springer, Heidelberg (2012)
11. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology-CRYPTO 1999, vol. 1666 of LNCS, pp. 388–397. Springer, Heidelberg (1999)
12. Ledig, H., Muller, F., Valette, F.: Enhancing collision attacks. In: Joye, M., Quisquater, J.-J. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2004, vol. 3156 of LNCS, pp. 176–190. Springer, Heidelberg (2004)
13. Lomne, V., Roche, T.: Collision-correlation attack against some 1st-order Boolean masking schemes in the context of secure devices. In: Prouff, E. (ed.) Constructive Side-Channel Analysis and Secure Design: COSADE, LNCS. Springer (2013, to appear)
14. Mangard, S.: Hardware countermeasures against DPA? a statistical analysis of their effectiveness. In: Okamoto, T. (ed.) Topics in Cryptology-CT-RSA 2004, vol. 2964 of LNCS, pp. 222–235. Springer, Heidelberg (2004)
15. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2010, vol. 6225 of LNCS, pp. 125–139. Springer, Heidelberg (2010)
16. Moradi, A.: Statistical tools flavor side-channel collision attacks. In: Johansson, T., Pointcheval, D. (eds.) Advances in Cryptology-EUROCRYPT 2012, vol. 7237 of LNCS, pp. 428–445. Springer, Heidelberg (2012)
17. Poettering, B.: Fast AES implementation for Atmel's AVR microcontrollers. http://point-at-infinity.org/avraes/
18. Renauld, M., Standaert, F.-X., Flandre, D.: Information theoretic and security analysis of a 65-nanometer DDSLL AES S-box. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2011, vol. 6917 of LNCS, pp. 223–239. Springer, Heidelberg (2011)
19. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2005, vol. 3659 of LNCS, pp. 30–46. Springer, Heidelberg (2005)
20. Schramm, K., Leander, G., Felker, P., Paar, C.: A collision-attack on AES: Combining side channel and differential-attack. In: Joye, M., Quisquater, J.-J. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2004, vol. 3156 of LNCS, pp. 163–175. Springer, Heidelberg (2004)
21. Schramm, K., Wollinger, T.J., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) Fast Software Encryption-FSE 2003, vol. 2887 of LNCS, pp. 206–222. Springer, Heidelberg (2003)
22. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) Advances in Cryptology-EUROCRYPT 2009, vol. 5479 of LNCS, pp. 443–461. Springer, Heidelberg (2009)
23. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography-SAC 2012, vol. 7707 of LNCS, pp. 390–406. Springer, Heidelebrg (2012)
24. Veyrat-Charvillon, N., Gérard, B., Standaert, F.-X.: Security evaluations beyond computing power: how to analyze side-channel attacks you cannot mount? To be published at EUROCRYPT (2013) (Preliminary work can be found at http://eprint.iacr.org/2012/578)
25. Veyrat-Charvillon, N., Standaert, F.-X.: Generic side-channel distinguishers: Improvements and limitations. In: Rogaway, P. (ed.) Advances in Cryptology-CRYPTO 2011, vol. 6841 of LNCS, pp. 354–372. Springer, Heidelberg (2011)