An Optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks

Nicolas Veyrat-Charvillon[⋆], Benoît Gérard[⋆⋆], Mathieu Renauld[⋆ ⋆ ⋆], François-Xavier Standaert[†]

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

Abstract. Methods for enumerating cryptographic keys based on partial information obtained on key bytes are important tools in cryptanalysis. This paper discusses two contributions related to the practical application and algorithmic improvement of such tools.

On the one hand, we observe that modern computing platforms allow performing very large amounts of cryptanalytic operations, approximately reaching 2^{50} to 2^{60} block cipher encryptions. As a result, cryptographic key sizes for such ciphers typically range between 80 and 128 bits. By contrast, the evaluation of leaking devices is generally based on distinguishers with very limited computational cost, such as Kocher's Differential Power Analysis. We bridge the gap between these cryptanalytic contexts and show that giving side-channel adversaries some computing power has major consequences for the security of leaking devices. For this purpose, we first propose a Bayesian extension of non-profiled side-channel attacks that allows us to rate key candidates in function of their respective probabilities. Next, we investigate the impact of key enumeration taking advantage of this Bayesian formulation, and quantify the resulting reduction in the data complexity of the attacks.

On the other hand, we observe that statistical cryptanalyses usually try to reduce the number and size of lists corresponding to partial information on key bytes, in order to limit the time and memory complexity of the key enumeration. Quite surprisingly, few solutions exist that allow an efficient merging of large lists of subkey candidates. We provide a new deterministic algorithm that significantly reduces the number of keys to test in a divide-and-conquer attack, at the cost of limited (practically tractable) memory requirements. It allows us to optimally enumerate key candidates from any number of (possibly redundant) lists of any size, given that the subkey information is provided as probabilities. As an illustration, we finally exhibit side-channel cryptanalysis experiments where the correct key candidate is ranked up to position 2^{32}, in which our algorithm reduces the number of keys to test offline by an average factor 2^5 and a factor larger than 2^{10} in the worst observed cases, compared to previous solutions. We also suggest examples of statistical attacks in which the new deterministic algorithm would allow improved results.

[⋆] Postdoctoral researcher supported by the ERC project 280141 - CRASH.
[⋆⋆] Postdoctoral researcher supported by Walloon region MIPSs project.
[⋆ ⋆ ⋆] PhD student supported by Walloon region SCEPTIC project.
[†] Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

1 Introduction

Side-channel attacks represent an important threat to the security of cryptographic hardware products. As a consequence, evaluating the information leakage of microelectronic circuits has become an important part in the certification of secure devices. Most of the tools/attacks that have been published in this purpose are based on a so-called "divide-and-conquer" approach. That is, in a first "divide" part, the evaluator/adversary recovers information about different parts of the master key, usually denoted as subkeys (as a typical example, the target can be the 16 AES key bytes). Next, a "conquer" part aims to combine the information gathered in an efficient way, in order to recover the full master key.

Research over the last ten years has been intensive in the optimization of the divide part of attacks. Kocher et al.'s Differential Power Analysis (DPA) [17] and Brier et al.'s Correlation Power Analysis (CPA) [7] are notorious examples. One limitation of such approaches is their somewhat heuristic nature, as they essentially rank the subkeys according to scores that do not have a probabilistic meaning. As demonstrated by Junod in the context of linear cryptanalysis, such heuristic key ranking procedures may be suboptimal compared to Bayesian key recoveries [15]. The template attacks introduced by Chari et al. in 2002 typically aims to get rid of this limitation [8]. By carefully profiling a probabilistic model for the physical leakages, such attacks offer a direct path towards Bayesian subkey testing procedures. Template attacks are optimal from an information theoretic point of view, which makes them a prime tool for the worst-case security evaluation of leaking devices [35]. However, they also correspond to strong adversarial assumptions that may not be met in practice. Namely, actual adversaries are not always able to profile an accurate leakage model, either because of a lack of knowledge of the target devices or because of physical variability [29]. As a consequence, attacks profiling an "on-the-fly" leakage model such as the stochastic approach introduced by Schindler et al. [30] and discussed by Doget et al. [11] are an important complement to a worst-case security analysis.

By contrast, only little attention has been paid to the conquer part in side-channel analysis. That is, in most cases the attacks are considered successful if all the subkeys are recovered with high confidence, which generally implies an extremely small time complexity for the offline computations. This situation is typically exemplified by initiatives such as the DPA contest [25], where the success rate in recovering a master key is directly obtained as the success rates for the concatenated 16 subkeys ranked first. In fact, the most noticeable exceptions attempting to better exploit computational power in physical attacks are based on advanced techniques, e.g. exploiting the detection of collisions [5, 18, 31, 32], or taking advantage of algebraic cryptanalysis [6, 23, 27, 28], of which the practical relevance remains an open question (because of stronger assumptions). But as again suggested by previous works in statistical cryptanalysis, optimal key ranking procedures would be a more direct approach in order to better trade data and time complexities in "standard" side-channel attacks.

In this paper, we propose to improve both the divide and the conquer parts of side-channel attacks, with two main contributions. Regarding the divide part, we start with the observation that non-profiled side-channel attacks are usually limited by their heuristic use of scores when ranking subkey candidates. As a consequence, we propose a method for non-profiled attacks that allows deriving probability mass functions for the subkey hypotheses. This tool can be viewed as a natural extension of the stochastic approach, but is also applicable to DPA and CPA. Afterwards, obtaining the probability of a full key candidate boils down to multiplying the probabilities of the corresponding subkeys. Hence, it has applications both in the online analysis of an attack (e.g. in order to express the level of knowledge about a subkey) and in the combination of independent attacks (e.g. exploiting multiple points of interest in the leakage traces). More generally, expressing the information obtained through non-profiled side-channel attacks with probabilities allows us to connect them better with template attacks, where the scores are already expressed as subkey probability mass functions.

Second, and most importantly, we provide the first comprehensive investigation of the conquer part of side-channel attacks. For this purpose, we start from the motivation that testing several billions of key candidates on a modern computer is not far-fetched: being able to recover a master key after such a computation is indeed a security breach. Next, we observe that two main solutions for testing key candidates from partial information on the subkeys exist in the literature. On the one hand, Meier and Staffelbach proposed a probabilistic algorithm in 1991. However, it turns out that in our side-channel attack context, this solution implies significant overheads in terms of number of keys to test. On the other hand, Pan, van Woudenberg, den Hartog and Witteman described a deterministic key enumeration algorithms at SAC 2010. But large memory requirements prevent the application of this second solution when the number of keys to enumerate increases. For example in [24], the authors were limited to the enumeration of 2^{16} candidates. As none of these tools is perfectly suited to our side-channel attack context, we propose a new deterministic algorithm for key enumeration that is time and memory efficient, and allows the optimal enumeration of full keys by decreasing order of probabilities. It takes advantage of the probability mass functions of subkeys made available through our first contribution. The new algorithm can be viewed as a refinement of the SAC 2010 one, in which we reduce the memory complexity of the enumeration thanks to a recursive decomposition of the problem. Interestingly, and as previously observed in other cryptanalysis settings, this improvement of the key ranking strategy also has a significant impact on the data complexity of side-channel key recoveries.

In practice, and for illustration, we can output the best 2^{32} candidates of an attack using 2GB of memory, against only 2^{20} candidates using more than 7GB using the SAC 2010 proposal. Moreover, our solution also allows significantly improved execution speeds, as it is less affected by the access times in large data sets. Eventually, in a similar context (i.e. the enumeration of 2^{32} candidates) and compared to the probabilistic solution of Meier and Staffelbach, the deterministic algorithm allows reducing the number of keys to test offline by an average factor 2^5 and a factor larger than 2^{10} in the worst observed cases. Summarizing, this work brings an interesting complement to the evaluation framework in [35]. It allows stating standard side-channel attacks as a data complexity vs. time complexity tradeoff. On the theoretical side, the proposed key enumeration algorithm leads to a proper estimation of security metrics such as high-order success rates or guessing entropy, for block cipher master keys (this estimation was previously limited to subkeys or small orders). In practice, experimental results also exhibit that considering adversaries with a reasonable computing power leads to significant improvements of standard side-channel attacks. These gains are particularly interesting if we compare them with the ones obtained by only working on the statistics in the divide part of side-channel attacks [34]. Hence, we believe that the tools introduced in this paper have an important impact for the security evaluations of leaking devices, e.g. for certification laboratories. In this respect, it is worth noticing the gap between the computational complexities usually considered in the evaluation of side-channel attacks and the ones considered for evaluating security against mathematical attacks [19, 26].

We finally note that the tools introduced in this paper are generic and have potential impact in other cryptanalytic settings (e.g. based on faults [2], or statistical [1, 21]), although standard side-channel attacks are a very natural environment for using them. We briefly list possible applications in Section 6.

2 Background

The "standard" side-channel attacks considered in this work use a divide-and-conquer approach in which the side-channel subkey recovery phase focuses on one specific operation at a time [20]. In block ciphers like the AES, this operation is usually one 8-bit S-box in the first encryption round. We denote with p and k the byte of the plaintext and the byte of the key (i.e. the subkey) that are used in the attack, with $x = p \oplus k$ the input value of the S-box, and with $y = \mathsf{S}(x)$ the corresponding output value. The goal of a side-channel subkey recovery phase is to identify the best (and hopefully correct) subkey candidate $\hat{k}$ from the set of all possible key hypotheses $\mathcal{K}$, using q measured encryptions. For each target S-box the adversary collects a data set of pairs $\{(p_i, l_i)\}_{1 \leq i \leq q}$[1], with p_i the i^{th} plaintext byte involved in the target S-box computation, and l_i the corresponding leakage value. For simplicity, and because it has little impact on our following discussions, we assume unidimensional leakages. In addition, we assume leakage samples composed of a deterministic and a random part, with the deterministic part depending only on the S-box output (i.e. we use the EIS assumption introduced in [30]). The leakage samples can consequently be written as $l_i = \mathsf{L}(y_i) = \mathsf{f}(y_i) + n$, with n a Gaussian distributed noise. In general, side-channel attacks can be classified as profiled and non-profiled attacks, depending on whether the adversary can perform a training phase or not.

[1] In order to lighten the notations, we omit the index $1 \leq i \leq q$ after the data sets.

Profiled attacks, like template attacks [8], take advantage of their profiling phase to characterize the leaking device with a probabilistic model. This allows the adversary to rank the subkey hypotheses k according to their actual probabilities: $\hat{k} = \arg\max_k \Pr[k|\{(p_i, l_i)\}]$. These probabilities can then directly be used to build a Probability Mass Function (PMF): $f_K(k) = \Pr[k|\{(p_i, l_i)\}]$, with K the discrete random variable corresponding to the unknown subkey byte. This PMF will be needed to perform the key enumeration in Section 4.

By contrast, in the case of non-profiled attacks (e.g. DPA [17] or CPA [7]), the best subkey hypothesis is not chosen based on probabilities, but on the value produced by a statistical distinguisher (namely, a difference-of-means test for Kocher's DPA and Pearson correlation coefficient for CPA). For these non-profiled attacks, there is thus no straightforward way to produce the PMF we need to enumerate the master keys: the distinguisher outputs a ranking of the subkey candidates, which has no probabilistic meaning.

In order to apply our key enumeration algorithm, we need a way to extract probabilities from a non-profiled attack. For this purpose, we will use a natural extension of the non-profiled version of Schindler's stochastic approach [30]. Hence, we first recall how the non-profiled stochastic attack works [11]. A stochastic model $\theta(y)$ is a leakage model used to approximate the leakage function: $\mathsf{L}(y) \simeq \theta(y)$, where $\theta(y)$ is built from a basis $\mathsf{g}(y) = \{\mathsf{g}_0(y), ..., \mathsf{g}_{B-1}(y)\}$ chosen by the adversary (usually $\mathsf{g}_i(y)$ are polynomials in the bits of y). Evaluating $\theta(y)$ boils down to estimating the coefficients α_i such that the vector $\theta(y) = \sum_j \alpha_j \mathsf{g}_j(y)$ is a least-square approximation of the measured leakages l_i. The idea of a non-profiled stochastic attack is to build $|\mathcal{K}|$ stochastic models $\theta_k(y)$ by considering the data set $\{(p_i, l_i)\}$ under the assumption that the correct key hypothesis is k. These stochastic models are then used as a distinguisher: for a correct key hypothesis (and a relevant basis), the error between the predicted values and the actual leakage values should have a smaller standard deviation than for a wrong key hypothesis. The pseudo-code of the attack is given in Algorithm 1. In general, an interesting feature of such attacks is that they allow trading robustness for precision in the models, by adapting the basis $\mathsf{g}(y)$. That is, a simpler model with less parameters is more robust, but a more complex model can potentially more accurately approximate the real leakage function.

Algorithm 1 Non-profiled stochastic attack

```
1: Acquire $\{(p_i, l_i)\}_{1 \leq i \leq q}$.
2: Choose a basis $\mathsf{g}(y)$.
3: for $k \in \mathcal{K}$ do
4:     Compute the S-box output hypotheses $y_{i,k} = \mathsf{S}(p_i \oplus k)$.
5:     Use the basis $\mathsf{g}(y)$, the data set and the subkey hypothesis $k$
       in order to build a stochastic model $\theta_k$.
6:     Compute the error vector $e_k$: $e_{i,k} = l_i - \theta_k(y_{i,k})$.
7:     Evaluate the precision of the model: $\sigma_k =$ standard deviation$(e_k)$.
8: end for
9: Choose $\hat{k} = \arg\min_k \sigma_k$
```

3 Bayesian extension of non-profiled SCAs

As the straightforward application of a stochastic attack does not produce PMFs, we propose in this section to perform an additional Bayesian step after building the stochastic models. We show that this Bayesian model comparison produces probabilities, and that the criterion of maximizing the likelihood of the subkey is equivalent to minimizing the error vector standard deviation, meaning that this extension indeed ranks the subkeys in the same order as the standard non-profiled stochastic attack. As a bonus, we observe that this extension also gives us a very natural way to combine independent leakage samples in an attack.

In the Bayesian version of the non-profiled stochastic attack, we perform a Bayesian hypothesis test on subkey candidates (under the assumption that the basis used for the stochastic attack is valid). It consists in estimating the probability of the observed data set assuming that they are produced from the model θ_k (i.e. $\Pr[\{(p_i, l_i)\}|\theta_k]$). Then, we use Bayes' theorem to deduce the likelihood of the models (and thus the probabilities of the subkey hypotheses) given the data (i.e. $\Pr[\theta_k|\{(p_i, l_i)\}]$), as described by the pseudo-code of Algorithm 2.

Algorithm 2 Bayesian non-profiled stochastic attack

```
1 to 8: Same as Algorithm 1.
9: Perform a Bayesian model comparison: evaluate for each subkey hypothesis the
likelihood $\Pr[\theta_k|\{(p_i, l_i)\}]$ using Bayes' theorem.
10: Choose $\hat{k} = \arg\max_k \Pr[\theta_k|\{(p_i, l_i)\}]$.
```

We now show how to compute these probabilities, starting with the probability of observing the data set $\{(p_i, l_i)\}$ assuming it is produced by the model θ_k, using the subkey hypothesis k. This probability is computed by multiplying the probabilities of each individual event (p_i, l_i) of the data set:

$$\begin{aligned}\Pr[\{(p_i, l_i)\}|\theta_k] &= \Pr[\{(p_i, l_i)\}|\theta_k, K = k],\\ &= \prod_{i=1}^{q} \mathcal{N}(l_i|\theta_k(\mathsf{S}(p_i \oplus k)), \sigma_k),\end{aligned}$$

where $\mathcal{N}(x, \mu, \sigma)$ is the value of the normal distribution of mean μ and standard deviation σ evaluated at point x. If we denote the S-box output hypotheses as $y_{i,k} = \mathsf{S}(p_i \oplus k)$, the previous equation can be rewritten as:

$$\Pr[\{(p_i, l_i)\}|\theta_k] = \prod_{i=1}^{q} \frac{1}{\sqrt{2\pi}\ \sigma_k} \exp^{-\frac{1}{2\sigma_k^2}(l_i - \theta_k(y_{i,k}))^2}.$$

Since $\sigma_k^2 = \sum_{i=1}^{i=q}(l_i - \theta_k(y_{i,k}))^2/q$ (see Algorithm 1), if we use all q measurements, we can simplify the exponent and move all constants coefficients that do not depend on k in a normalization term Z, that is:

$$\Pr[\{(p_i, l_i)\}|\theta_k] = Z\sigma_k^{-q}. \tag{1}$$

Then, using Bayes' theorem, we deduce the probabilities of the subkeys from the respective likelihood values of the models θ_k given the data (in other words, we perform a Bayesian model comparison):

$$\begin{aligned}\Pr[k|\{(p_i,l_i)\}] &= \Pr[\theta_k|\{(p_i,l_i)\}],\\ &= \frac{\Pr[\{(p_i,l_i)\}|\theta_k].\Pr[\theta_k]}{\Pr[\{(p_i,l_i)\}]}, \quad \text{(Bayes' theorem)}\\ &= \frac{\Pr[\{(p_i,l_i)\}|\theta_k].\Pr[\theta_k]}{\sum_{k'\in\mathcal{K}}\Pr[\{(p_i,l_i)\}|\theta_{k'}].\Pr[\theta_{k'}]}.\end{aligned}$$

Assuming a uniform prior $\Pr[\theta_k]$=$\Pr[k]$=$\frac{1}{|\mathcal{K}|}$ and using Equation 1, we get:

$$\Pr[k|\{p_i,l_i\}] = \frac{\sigma_k^{-q}}{\sum_{k'}\sigma_{k'}^{-q}}.$$

From these probabilities, we can directly build the PMF required for key enumeration. Note that these likelihood values are not exactly the same as the ones we used in template attacks. In the last case, the characterization of a device allows exploiting a precise estimation of the leakage distributions. By contrast, in a Bayesian non-profiled stochastic attack, they depend on the basis $\mathsf{g}(y)$ chosen by the adversary. Finally, the subkey hypotheses can be ranked according to the likelihood values of the corresponding model given the data, that is:

$$\begin{aligned}\hat{k} &= \arg\max_k \sigma_k^{-q},\\ &= \arg\min_k \sigma_k,\end{aligned}$$

i.e. providing the same ranking as for the original non-profiled stochastic attack. Besides their use for key enumeration in the next section, an appealing property of using probabilities instead of other criteria (e.g. like a correlation coefficient) is that combining independent measurement points becomes very natural. Let us suppose that our implementation leaks information at two different times: $\mathsf{L}_{t_0} = \mathsf{f}_{t_0}(x)+n_{t_0}$ and $\mathsf{L}_{t_1} = \mathsf{f}_{t_1}(y)+n_{t_1}$ (with n_{t_i} a Gaussian noise). The Bayesian writing makes it straightforward to combine their corresponding probabilities, by multiplication and normalization. Note also that the proposed attack can additionally be seen as a generalization of DPA or CPA, by simply replacing the leakage basis by a single-bit or Hamming weight model.

4 Efficient algorithms for combining subkeys

Following the evaluation framework in [35], different metrics can be used to analyze the security of an implementation against side-channel attacks. Of particular interest in this work are the so-called "security metrics" (namely, the success rate and guessing entropy), of which the goal is to estimate the efficiency of a given distinguisher in exploiting the leakage to recover keys. Intuitively, a success rate of order o corresponds to the probability that the correct key is rated among the o first candidates provided by the distinguisher. The guessing entropy corresponds to the average number of keys to test before reaching the correct one. As suggested in [24], one can also consider a guessing entropy of order o, in order to capture the fact that in practice, only a maximum number of o keys can be tested by the evaluators. Empirical comparisons of distinguishers using such metrics have been performed in [34], but were limited to subkey recoveries (i.e. key bytes, typically). In the following, we consequently tackle the (most practically relevant) problem of how to efficiently estimate these metrics for master keys. In the extreme case (i.e. success rate of order 1), the solution is straightforward, as e.g. illustrated by the DPA contest [25]. We consider the general case of large lists and large orders, to carefully address the problem of the "conquer" part in a side-channel attack. The problem of extracting the rank of a correct key is equivalent to the problem of enumerating keys stated below.

Key-enumeration problem. The attacker obtains PMFs corresponding to d independent subkeys, each PMF taking n different values. The problem is to enumerate complete keys from the most probable to the least probable one.

The naive algorithm for solving this problem generates the list of all possible keys, computes the corresponding likelihood values (by multiplying subkey probabilities) and sorts them accordingly. It can be used only with key candidates lists of limited size. In the remainder of this section, we propose an algorithm that efficiently solves this problem and allows deterministic key-enumeration, even when the number and size of subkey lists makes the naive approach untractable.

4.1 An efficient and deterministic key-enumeration algorithm

The key enumeration problem can be more readily understood as a geometric problem. We first consider the simpler bi-dimensional case (that is 2 subkeys). The key space can be identified with a compartimentalized square of length 1. The enumeration process is illustrated in Figure 1. The 4 columns (resp. rows) correspond to the four possible values of the first (resp. second) subkey, sorted by decreasing order of probability. Width and height correspond to the probability of the corresponding subkey. Let us denote by $k_i^{(j)}$ the j^{th} likeliest value for the i^{th} subkey. Then, the intersection of row j_1 and column j_2 is a rectangle corresponding to the key $(k_1^{(j_1)}, k_2^{(j_2)})$ with probability equal to the area of the rectangle. Using this geometric view of the problem, an optimal key enumeration algorithm outputs compartments by decreasing order of area.

Step 1. The most likely key is $(k_1^{(1)}, k_2^{(1)})$. Hence, we output this one first (represented in dark gray). At this point, the only possible next key candidates are the successors $(k_1^{(2)}, k_2^{(1)})$ and $(k_1^{(1)}, k_2^{(2)})$, shown in light gray. We denote by $\mathcal{F}$ this set of potential next candidates (where $\mathcal{F}$ is standing for frontier).

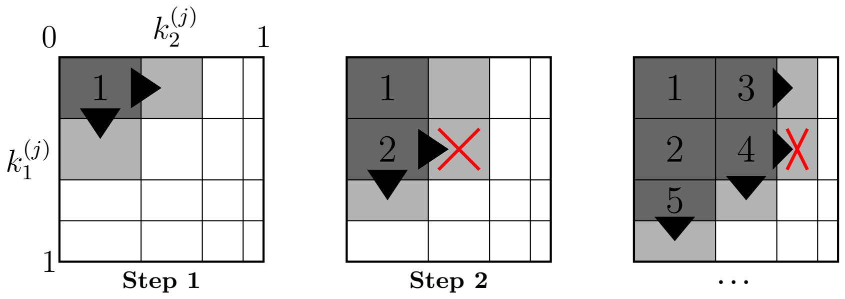

Fig. 1: Geometric representation of the proposed algorithm

Step 2. Any new candidate has to belong to the frontier set. We extract the most likely candidate from this set and output it. It corresponds to rectangle 2 in our example. $\mathcal{F}$ is updated by inserting the potential successors of this candidate.

Next steps. Step 2 is repeated until the correct key is output, or if the size of the frontier set $\mathcal{F}$ exceeds the available memory space.

Notice that in step 2, $(k_1^{(2)}, k_2^{(1)})$ is a more likely candidate than $(k_1^{(2)}, k_2^{(2)})$ by construction. Hence, $(k_1^{(2)}, k_2^{(2)})$ should not be inserted into $\mathcal{F}$ (this is represented on the figure by red crosses). There is a simple rule for handling such cases, which allows us to minimize the memory requirements of our algorithm.

Rule 1. The set $\mathcal{F}$ may contain at most one element in each column and row.

We explicit the process in Algorithm 3.

Algorithm 3 Optimal key-enumeration.

```
$\mathcal{F} \longleftarrow \{(k_1^{(1)}, k_2^{(1)})\}$;
while $\mathcal{F} \neq \emptyset$ do
  $(k_1^{(i)}, k_2^{(j)}) \longleftarrow$ most likely candidate in $\mathcal{F}$;
  Output $(k_1^{(i)}, k_2^{(j)})$;
  $\mathcal{F} \longleftarrow \mathcal{F} \setminus \{(k_1^{(i)}, k_2^{(j)})\}$;
  if $i+1 \leq \#k_1$ and no candidate in row $i+1$ then
    $\mathcal{F} \longleftarrow \mathcal{F} \cup \{(k_1^{(i+1)}, k_2^{(j)})\}$;
  end if
  if $j+1 \leq \#k_2$ and no candidate in column $j+1$ then
    $\mathcal{F} \longleftarrow \mathcal{F} \cup \{(k_1^{(i)}, k_2^{(j+1)})\}$;
  end if
end while
```

Operations on the frontier set can be performed efficiently if candidate keys are stored in an ordered structure. Indeed, these operations simply consist in finding the most likely element in the set, inserting and removing elements. Using balanced trees, these manipulations are logarithmic in the size of the set. The test of Rule 1 can be implemented using arrays of Boolean values.

Generalization to multiple lists. In practice, one often has to merge together more than two lists of subkeys. Direct extensions of our algorithm to higher dimensions lead to either suboptimal or slow roles for frontier set reduction. On the one hand, the direct transposition of Rule 1 will minimize memory, but implies adjacency tests in multiple dimensions, leading to unacceptable reductions of the enumeration speed. On the other hand, simplifying the rule in order to maintain a good enumeration speed implies the storage of many non-necessary candidates in the frontier set, which rapidly leads to unsustainable memory requirements. In order to obtain good results for more than two lists, we apply a recursive decomposition of the problem.

We only use the algorithm for merging two lists, and its outputs are used to form larger subkey lists which are in turn merged together. This way, merging n lists is done by merging two lists $n-1$ times. The order of merging is such that lists merged together are of similar sizes. In the case of the AES, we obtain something similar to a binary tree. Still considering the case of the AES, we notice that enumerating 128-bit keys is done by merging two lists of size 2^{64}. Such lists cannot be generated or stored efficiently. Fortunately, we can instead generate these lists only as far as required by the key enumeration. Whenever a new subkey is inserted in the candidate set, we get it from the enumeration algorithm applied to the lower level (64-bit subkeys obtained by merging two 2^{32} element lists), and so on. This ensures that the storage and enumeration effort are minimized.

The process is illustrated in Figure 2. Enumerating 16-byte keys consists in enumerating subkeys taken from the two 2^{64}-element lists $k^{(j)}_{0,\ldots,7}$ and $k^{(j)}_{8,\ldots,15}$, which in turn are built using four 2^{32}-element lists $k^{(j)}_{0,\ldots,3}$, $k^{(j)}_{4,\ldots,7}$, etc. This process is repeated until we reach the original 2^{8}-element subkey distributions. The recursive decomposition combined with our *lazy evaluation* technique keep computations and memory requirements to a minimum and allow us to enumerate a large number of key candidates. Note that since there are only few dependencies between the enumeration lists, the algorithm can additionally be accelerated through parallelization.

4.2 Key rank and high-order success rate

The algorithm used here is specifically designed in order to allow an adversary to recover encryption keys when it is not ranked first. A close but simpler problem is that of estimating the rank of the correct key, which does not necessarily require to enumerate all the keys candidates ranked before: we just need to estimate how many they are. This problem occurs for example during security evaluation, where the correct key is usually known and the question is to rate the recovery effort in terms of trace acquisitions and key trials.

Note that a deterministic algorithm, to the best of our knowledge, would have a complexity close to that of actual enumeration. We sketch a heuristic algorithm that provides a range estimate of the correct key rank. Considering the problem in d dimensions, the checker plane seen before becomes a partitionned

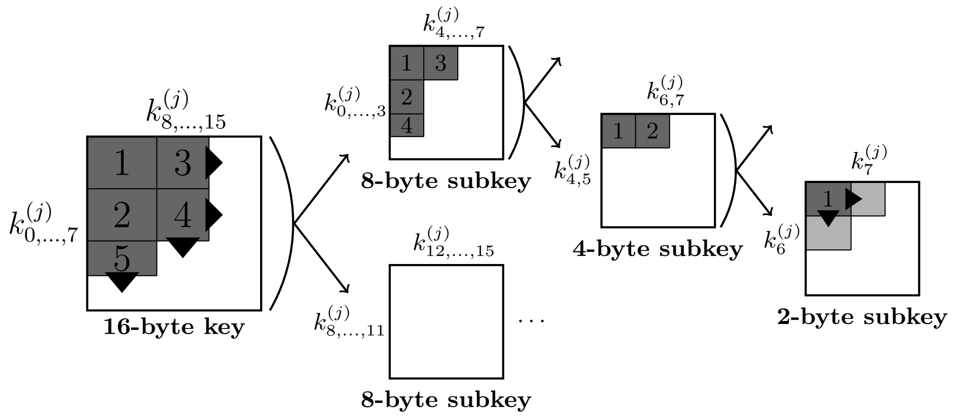

Fig. 2: Recursive enumeration from multiple lists of key candidates.

hypercube. If we imagine equipotential surfaces on candidate probability in this cube, the key rank is related directly to the number of candidates contained in the volume between the surface passing through the correct key point and the planes delimiting the hypercube. By sampling key candidates, it is possible to give higher and lower estimates of this value, thereby computing ranges for key ranks.

4.3 Comparison with previous works

The investigations in this paper have strong connections with previous works in the area of statistical cryptanalysis. In particular, the problem of merging *two* lists of subkey candidates was encountered by Junod and Vaudenay [16]. The small cardinality of the lists (2^{13}) was such that the simple approach that consists in merging and sorting the lists of subkeys was tractable. Markus Dichtl considered a similar problem of enumerating key candidates by decreasing order of probabilities, thanks to partial information obtained for each key bit individually [10]. We tackle the more general and challenging problem of exploiting any partial information on subkeys. A frequent reference for solving this problem, i.e. enumerating many keys from lists that cannot be merged, is the probabilistic algorithm that was proposed in [22]. In this work, the attacker had no access to the subkey distributions but was able to generate subkeys according to them. Hence, the solution proposed was to enumerate keys by randomly drawing subkeys according to these distributions. Implementing this algorithm is equivalent to uniformly picking up a point in the square of Figure 1 and testing the corresponding key. This does not require any memory but the most probable keys may be drawn many times, leading to useless repetitions. Indeed given a correct key with probability p the number of keys to try before it is found follows a geometric distribution with parameter p and thus has an expected value equal to $1/p$ with a variance of $\frac{1-p}{p^2}$. By contrast, for Algorithm 3, this number of keys to test is *at most* $\lfloor 1/p \rfloor$ (usually much less). Actually, our algorithm will output exactly n keys before the correct one if it is ranked in the n-th position, removing the variance issues of the probabilistic test. Overall, the probability-driven approach tends to lead to much more tests than optimal deterministic enumeration, as will be illustrated experimentally in the next section.

In terms of complexities, it is easy to see that the probability-driven algorithm can output new keys in constant time and has a very small memory requirement. The case of our deterministic enumeration algorithm is more difficult. The use of balanced trees for the frontier set and the recursive decomposition of the problem point towards a logarithmic time complexity. However, it appears from the experiments in the next section that the algorithm enumerates keys in amortized time close to constant. Summarizing, both methods lead to a linear time complexity in the total number of key candidates that are output, with the enumeration algorithm also requiring a sub-linear amount of memory.

As previously mentioned, an enumeration algorithm similar to ours was proposed in a paper by Pan, van Woudenberg, den Hartog and Witteman [24]. It also enumerates key candidates in optimal order, but the reduction rule 1 is not used, nor the recursive decomposition that allows us to efficiently apply rule 1 with more than two lists. Therefore, the frontier set of their algorithm is not minimal, and the memory requirements are much larger. In practice, since the main limitation for optimal key enumeration appears to be memory, this non-minimal version of enumeration does not allow an adversary to output a large number of key candidates. Moreover, handling a larger frontier set implies time complexity penalties, which makes our new algorithm faster than this previous one. Implementation results confirming these claims are given in the next section.

We note that another related problem is list decoding of convolutional codes through the Viterbi algorithm (see, e.g. [33]). However, such algorithms are generally designed to output a small number of most likely candidates determined a priori, whereas we target the enumeration of 2^{32} or more master key candidates.

Finally, the application of Algorithm 3 in the context of [22] may also be relevant. If keys can be generated according to an unknown distribution, this distribution can be estimated. Hence, by performing such a pre-processing phase, the deterministic algorithm can be used. The resulting enumeration avoids repetitions and tend towards optimal as the distribution estimates become accurate.

5 Experiments

In order to validate our approach, we led several experiments. The cipher under investigation was the AES, with key size of 128 bits. Our side-channel attacks targeted the output of the S-boxes in the first round, resulting in 16 independent subkey probability mass functions. We used the same assumptions as in Section 2 and considered simulated leakages, following a Hamming weight leakage model on the S-box output, with an independent additive Gaussian noise, i.e. $\mathsf{f}(y) = \mathrm{HW}(y) + \mathcal{N}(0, 4^2)$. Note that the type of experiments performed (i.e. analyzing the impact of key enumeration) is essentially independent of both the cipher and experimental setup. We carried out both template attacks with perfect profiling (since the leakage function is known) and non-profiled Bayesian stochastic attacks assuming a basis made of the S-box output bits.

5.1 Comparing optimal and probabilistic key-enumerations

Table 1 gives some performance results for key enumeration obtained on our setup (Intel core i7 920 running a 64-bit Ubuntu 11.04 distribution), further predicted for 2^{36} and 2^{40} key candidates. These comparative results show that both the sampling algorithm and ours can output key candidates at essentially the same speed. The results for the enumeration algorithm described in [24] are also given. As expected, they exhibit larger memory requirements, which bounds the number of key candidates that can be enumerated and increases time complexity. In practice, we were limited to 2^{20} candidates using this algorithm. By contrast, the memory requirements of our deterministic algorithm are low enough to allow us reaching 2^{32} key candidates in one hour, using less than 2GB. Predictions show that it is possible to reach 2^{40} using 70GB of memory in nine days. Note that enumeration time can be improved using parallel implementations.

Method	#trials	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}	2^{36}	2^{40}
Sampling	Time	0s	0.04s	0.31s	10.1s	160s	2560s	11h	182h
[24]	Time	0.05s	0.96s	18.1s	X	X	X	X	X
	Memory	7.7MB	118MB	7.7GB					
Proposed	Time	0s	0.03s	0.55s	9.2s	163s	3130s	12h	221h
	Memory	88KB	405KB	2.7MB	20MB	225MB	1.8GB	10GB	70GB

Table 1: Practical comparison of key-enumeration algorithms.

Next, as mentioned in Section 4, Algorithm 3 performs key trials in the best possible order, therefore minimizing the enumeration effort at the cost of a growing amount of memory space. By contrast, the probability-driven algorithm may miss some key candidates and output some more than once. In order to illustrate these differences we led the following experiment. A large number of side-channel attacks followed by a key recovery were performed, and we measured the key rank (which is also the number of trials for the optimal algorithm), the expected number of trials for the probability-driven algorithm and the memory used during optimal enumeration. Figure 3 illustrates the expected overhead of the probability-driven method over the deterministic one in terms of key trials (green dots, left scale) and the memory cost of the enumeration algorithm (blue dots, right scale). We observe that the probability-driven algorithm requires more key trials on average to complete an attack. The overhead increases consistently, and the median of the expected ratio value (green curve) appears to tend towards a linear relation on the log-log scale. An approximated power law gives $2^{3.2+log_2(r)/20}$, e.g. 16 for 2^{16}, 21 for 2^{32}, an extrapolated 36 for 2^{40}.

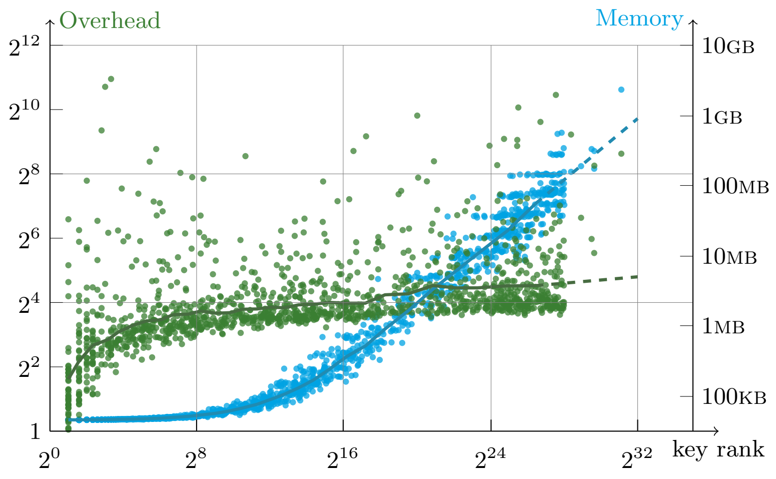

Fig. 3: Overhead of the probability-driven method in function of the key rank (green), and memory requirements of the deterministic enumeration (light blue).

In some cases, we also observe overheads very far from the median value (well over 1000), even when the correct key is ranked among the 4 first ones. On top of this *expected* number of trials, we have to consider that, since the probabilistic method follows a geometric law, the number of key trials will have a very large variance (approximately the square of expected value). This makes the probabilistic method both more costly and less reliable than our deterministic algorithm. Besides, the memory space requirement of the enumeration method also appears to follow a power law. We can see that enumerating up to 2^{32} requires about 1GB of memory. Extrapolations up to 2^{40} keys give an estimated 70GB, which is easily manageable using hard-disk drives in less than two weeks.

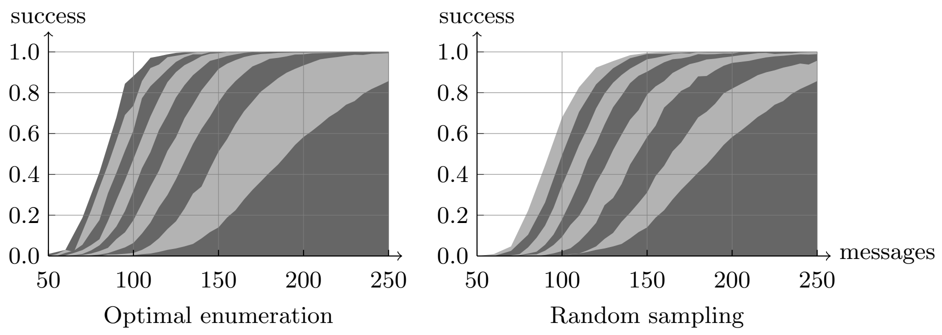

Fig. 4: Success rate of template attacks. Left: enumeration, right: sampling.

5.2 Application of key-enumeration to side-channel attacks

Figure 4 illustrates the success rate of different orders for a template attack, in function of the number of traces measured. The alternated light and gray zones correspond to the evolution of the success rate each time the number of tested keys is multiplied by 16. The rightmost dark gray curve is obtained by only testing the first key candidate, the first light gray curve by testing 2^4 keys, then 2^8, ... We again considered the optimal and the probabilistic enumerations, for different number of enumerated key candidates. The optimal enumeration was led up to 2^{32} candidates, and the probabilistic method up to 2^{28}. As expected, allowing more key candidates to be tested can dramatically increase the efficiency of a key recovery. For 120 messages measured, the best key candidate is the correct one about 2% of the time, while there is a 91% chance for the correct key to be found among the first 2^{24} candidates with the optimal enumeration algorithm (or an 84% chance with the probability driven method). In other words, increasing the number of key trials significantly improves the success rate, thereby providing a tradeoff between the data and time complexities of the attacks. As in the previous subsection, we also observe that the optimal enumeration algorithm leads to higher success rates compared to the random sampling for a given number of key trials, at the cost of additional memory requirements. Figure 5 shows a similar curve for the non-profiled variant of the stochastic attack described in Section 3, using the deterministic enumeration algorithm. It confirms the correctness of our Bayesian formulation[2].

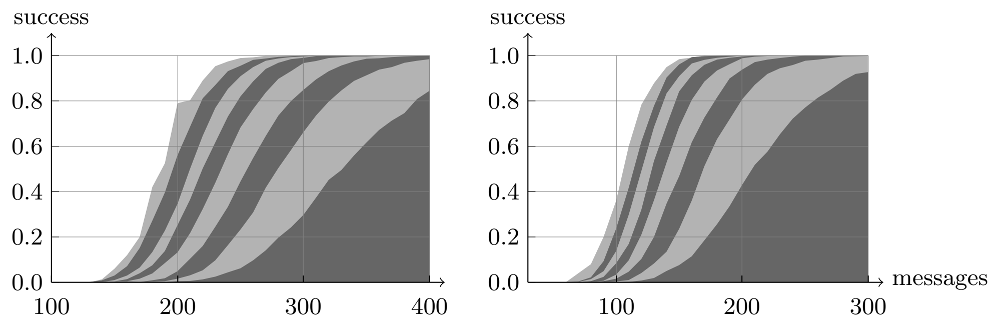

Fig. 5: Success rates of the Bayesian stochastic attack with optimal key enumeration. Left: 9-element linear basis. Right: 2-element Hamming weight basis.

Figure 6 provides an orthogonal view of the problem: for a given number of traces, one can increase the key success rate by enumerating more key candidates.

[2] Similar curves could also be obtained by performing a correlation attack, where the subkey probabilities are approximated using normalized correlation scores. Although such an approximation is not backed up by any theoretical argument, it worked well in simple implementation contexts (e.g. Hamming weight leakage model, typically).

The figure shows the cumulative probability function (CDF) of key recoveries for an attack with a fixed number of traces, in function of the number of key trials. The PMFs used for this experiment are output by two template attacks. The first attack (left) targets only 2 key bytes, the second (right) targets all 16 key bytes. As expected, the cumulative probability starts from 0 and reaches 1 once a sufficient number of keys have been tested. Also, the brute force testing is only possible in the left case (i.e. when we can enumerate the full list). In general, we see that the side-channel information allows obtaining high success rates with a limited number of key trials (here, up to 2^{30}). More importantly, the figure again confirms the interest of the deterministic algorithm in terms of "number of keys to test". Reaching a similar success rate with the probabilistic algorithm requires between 2^2 and 2^4 more tests, depending on the success rates.

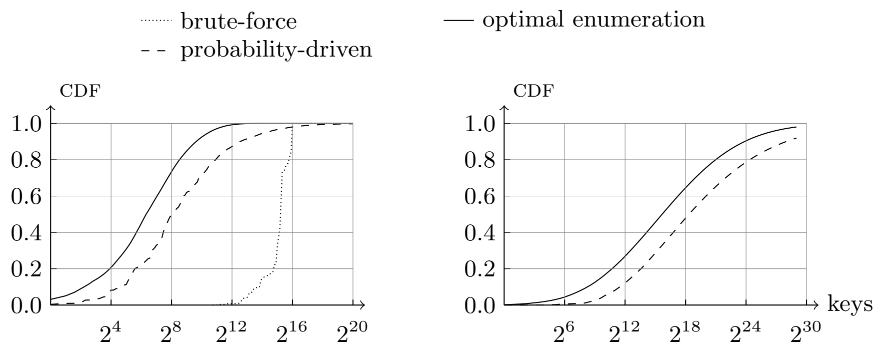

Fig. 6: Enumeration success rates. Left: 2 S-boxes, right: 16 S-boxes.

Finally, Figure 7 summarizes the previous observations and further illustrates the tradeoff between measurements and computations in the case of a template attack. The classical "best-key" success rate curve is shown in thick. The other curves give the number of key trials that are necessary in order to reach some success rate, given a number of measured encryptions, and for different quantiles. For example, let us consider the success median which corresponds to a 50% chance of key recovery. This rate can be acheived using 190 messages if only the best key candidate is considered (bottom black dot), 135 messages for 2^8 key trials (middle black dot), or only 110 messages for 2^{16} key trials (top black dot). Finally, the thick gray curve in the figure gives the entropy of the key distribution while the template attack is performed. This entropy is simply obtained by summing up the 16 independent entropies of the subkeys. Interestingly, it can be noticed that this entropy (that can be evaluated "on-the-fly" during an attack) is a good indicator of the remaining workload to recover the master key.

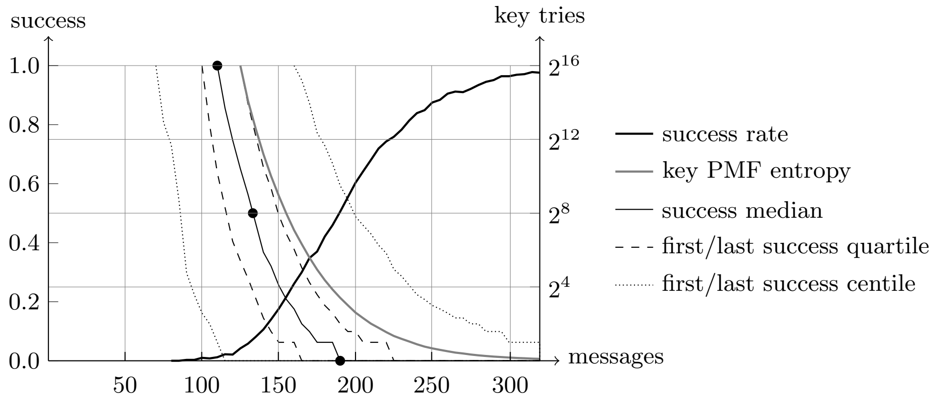

Fig. 7: Success rate for the best key candidate (thick), entropy of the key candidates distribution (gray), and key tries needed to reach a given success probability (dotted: 1% and 99%, dashed: 25% and 75%, solid: 50%).

6 Other applications

As mentioned earlier, the interest of the proposed key-enumeration algorithm is not restricted to the context of side-channel cryptanalysis: the problem of enumerating keys from subkey distributions may also arise in the case of multiple statistical cryptanalysis. Indeed, when considering a single statistical characteristic, one obtains a list of subkeys (typically corresponding to key bits used for a partial decryption) with their probabilities. When more characteristics are used, an attacker obtains different lists of subkeys and has to combine them.

This is the case, for instance, in multiple differential cryptanalysis [4] and multiple linear cryptanalysis [3]. Both types of cryptanalyses belong to the same family of so-called last-rounds attacks that consist in partially decrypting the available ciphertexts (e.g. by inverting few S-boxes), in order to compute some statistic related to a chosen characteristic (typically, a differential or linear approximation of the target cipher). As only k key-bits are required to perform the partial decryption, one can decrypt the ciphertexts for all possible values of this k-bit subkey, obtaining as many statistics. From them, the attacker derives the likelihood of being the correct subkey for each candidate. Using more than one characteristic directly leads to different lists of candidates and their probabilities. In general, the choice of characteristics is limited by the total number of key bits guessed. Indeed, if all the key bits are involved in at least one characteristic, recovering the full key by generating the list of candidates will be more time-consuming than exhaustive search. However, using an optimal enumeration algorithm, the set of characteristics can be split in many groups - each involving a small number of key-bits - so that generating the list of subkeys is tractable and the master key can then be enumerated from these lists.

This idea can be illustrated with the multiple linear cryptanalysis of Serpent [9]. In this attack, three different sets of linear approximations are formed, leading to attacks recovering $32, 40$ and 44 key-bits. Merging the three lists obtained from these groups would dramatically increase the time and memory complexities, leading to a worse attack than using only one group. Using the optimal key enumeration algorithm, we obtain an attack that outperforms the ones proposed in the original paper. A similar gain can be observed for the linear cryptanalysis of DES proposed in [12]. Here, using all the linear approximations, the number of involved key bit reaches 56 (i.e. the full-key size). Hence, authors only used a subset of approximations (32968 out of 74086) leading to a list of 2^{42} candidates. Again, the optimal enumeration algorithm allows us to extract more information from the available plaintext/ciphertext pairs, leading to a more efficient attack.

Subkey-bits overlap. We note that it is unlikely that sets of key bits corresponding to different characteristics are disjoints. Hence, Algorithm 3 has to be patched to be applied to multiple statistical cryptanalyses. This can be easily done by adding a rule in successor insertion, that skips inconsistant couples of subkeys when updating the frontier set. The time overhead induced by this additional rule remains small as the proportion of overlapping bits is typically small.

7 Conclusion

This paper complements standard side-channel cryptanalysis by investigating the attack improvements obtained by adversaries with non-negligible computational power. For this purpose, we first proposed an extension of non-profiled stochastic attacks that outputs probability mass functions, providing us with the likelihood values of subkey candidates. Next, we proposed a new and deterministic key enumeration algorithm, in order to take advantage of these likelihood values. Our experiments show that this order-optimal enumeration algorithm leads to more efficient attacks than a sampling-based algorithm from Eurocrypt 1991. In particular, the probabilistic algorithm suffers from its underlying geometric law, that implies an increasingly large overhead over the deterministic method (in terms of key trials), as the number of keys to enumerate increases. The deterministic method additionally allows removing the possibility of computationally intensive worst cases, which makes it a particularly suitable solution for side-channel security evaluations, in which we want to derive security metrics with high confidence. Finally, our proposal significantly reduces the memory requirements of a deterministic algorithm from SAC 2010, making it the best practical solution for enumeration of up to 2^{40} key candidates. As a result, the solutions in this paper allow us to properly trade side-channel measurements for offline computations. They create a bridge between classical DPA and brute-force key recovery, where information extracted through side-channels is used to improve an exhaustive search. Hence, an interesting scope for further research is to compare computationally-enhanced DPA attacks with other types of more computational side-channel attacks, e.g. based on the detection of collisions.

Note finally that the complete key recoveries we considered in this work can possibly be performed in a ciphertext-only context. That is, the adversary can evaluate a key candidate by partially decrypting the ciphertext and computing the probability of the leakages in the previous rounds. This probability will only be non-negligible if the ciphertext was decrypted with the correct key. Interestingly, in this particular setting, the test of one key candidate is especially expensive, making the optimal enumeration algorithm particularly relevant.

References

1. Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.
2. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.
3. Alex Biryukov, Christophe De Cannière, and Michaël Quisquater. On multiple linear approximations. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2004.
4. Céline Blondeau and Benoît Gérard. Multiple differential cryptanalysis: Theory and practice. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2011.
5. Andrey Bogdanov. Improved side-channel collision attacks on aes. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2007.
6. Andrey Bogdanov, Ilya Kizhvatov, and Andrei Pyshkin. Algebraic methods in side-channel collision attacks and practical collision detection. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2008.
7. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Joye and Quisquater [14], pages 16–29.
8. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
9. Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improved and multiple linear cryptanalysis of reduced round serpent. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Inscrypt*, volume 4990 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2007.
10. Marcus Dichtl. A new method of black box power analysis and a fast algorithm for optimal key search. Proceedings of COSADE 2011, February 2011. Darmstadt, Germany.
11. Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. Journal of Cryptographic Engineering, 2011. To appear.
12. Benoît Gérard and Jean-Pierre Tillich. On linear cryptanalysis with many linear approximations. In Matthew G. Parker, editor, *IMA Int. Conf.*, volume 5921 of *Lecture Notes in Computer Science*, pages 112–132. Springer, 2009.
13. Thomas Johansson, editor. *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*. Springer, 2003.

14. Marc Joye and Jean-Jacques Quisquater, editors. *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*. Springer, 2004.
15. Pascal Junod. On the optimality of linear, differential, and sequential distinguishers. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2003.
16. Pascal Junod and Serge Vaudenay. Optimal key ranking procedures in a statistical cryptanalysis. In Johansson [13], pages 235–246.
17. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
18. Hervé Ledig, Frédéric Muller, and Frédéric Valette. Enhancing collision attacks. In Joye and Quisquater [14], pages 176–190.
19. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.
20. Stefan Mangard, Elisabeth Oswald, and Francois-Xavier Standaert. One for all - all for one: Unifying standard dpa attacks. Cryptology ePrint Archive, Report 2009/449, 2009. `http://eprint.iacr.org/`.
21. Mitsuru Matsui. Linear cryptoanalysis method for des cipher. In *EUROCRYPT*, pages 386–397, 1993.
22. Willi Meier and Othmar Staffelbach. Analysis of pseudo random sequence generated by cellular automata. In *EUROCRYPT*, pages 186–199, 1991.
23. Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. Algebraic side-channel analysis in the presence of errors. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2010.
24. Jing Pan, Jasper G. J. van Woudenberg, Jerry den Hartog, and Marc F. Witteman. Improving dpa by peak distribution analysis. In *Selected Areas in Cryptography*, pages 241–261, 2010.
25. Télécom ParisTech. Dpa contest v2. http://www.dpacontest.org/v2/index.php, 2009/2010.
26. Cryptographic Key Length Recommendation. http://www.keylength.com/.
27. Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.
28. Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the aes: Why time also matters in dpa. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.
29. Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011.
30. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

31. Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A collision-attack on aes: Combining side channel- and differential-attack. In Joye and Quisquater [14], pages 163–175.
32. Kai Schramm, Thomas J. Wollinger, and Christof Paar. A new class of collision attacks and its application to des. In Johansson [13], pages 206–222.
33. Nambirajan Seshadri and Carl-Eric W. Sundberg. List viterbi decoding algorithms with applications. *IEEE Transactions on Communications*, 42(2/3/4):313–323, 1994.
34. François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008.
35. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.