Comparing Approaches to Rank Estimation for Side-Channel Security Evaluations

Romain Poussier, Vincent Grosso, François-Xavier Standaert.

ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.

Abstract. Rank estimation is an important tool for side-channel evaluations laboratories. It allows determining the remaining security after an attack has been performed, quantified as the time complexity required to brute force the key given the leakages. Several solutions to rank estimation have been introduced in the recent years. In this paper, we first clarify the connections between these solutions, by organizing them according to their (maximum likelihood or weak maximum likelihood) strategy and whether they take as argument a side-channel distinguishers' output or some evaluation metrics. This leads us to introduce new combinations of these approaches, and to discuss the use of weak maximum likelihood strategies for suboptimal but highly parallel enumeration. Next, we show that the different approaches to rank estimation can also be implemented with different mixes of very similar tools (e.g. histograms, convolutions, combinations and subsampling). Eventually, we provide various experiments allowing to discuss the pros and cons of these different approaches, hence consolidating the literature on this topic.

1 Introduction

Most side-channel attacks published in the literature proceed with a divide-and-conquer strategy. That is, they first extract information about independent pieces of a master key (next called subkeys), and then combine this information in order to recover their concatenation. Typical tools for the subkey information extraction include Kocher et al.'s Differential Power Analysis (DPA) [9], Brier et al.'s Correlation Power Analysis (CPA) [3], Chari et al.'s Template Attacks (TA) [4], Schindler et al.'s Linear Regression (LR) based attacks [11] and many others. As for the recombination part, two typical situations can happen. First, when the attack is close enough to succeed, *key enumeration* can be used, in order to list the most likely key candidates in decreasing order of likelihood [13]. With current computation power, this approach is typically successful when the correct (master) key is ranked up to positions 2^{40}-2^{50} in such a list. Second, when the enumeration becomes too intensive for being conducted in practice, *rank estimation* can be used [14]. In this case, one additionally requires the knowledge of the master key, hence it is only applicable in an evaluation context (while key enumeration is also applicable in an attack context). Based on the value of the master key and the subkey information, rank estimation aims to efficiently approximate the correct key rank (with a given accuracy).

	Sampling-based	Metric-based
wML	`How:` subsampling + combinations	
	`What:` *SR lower bound* (+ *subopt. parallel enumeration*)	`What:` SR lower bound (with limited sampling)
	`Acronym:` SLB, `Ref.`[∅]	`Acronym:` MLB, `Ref.` [5, 15]
ML	`How:` histograms + convolutions	
	`What:` SR estimation (tight even for large keys)	`What:` *SR upper bound* (*with limited sampling*)
	`Acronym:` SE, `Ref.` [1, 8, 14]	`Acronym:` MUB, `Ref.` [∅]

Table 1. Approaches to key rank estimation (*italic cases are new*).

Rank estimation is especially useful for evaluation laboratories. Indeed, it is a tool of choice for quantifying the security of an implementation whenever it goes beyond the computing power of the evaluator (i.e. whenever an implementation is not trivially insecure). As a result, a number of works have investigated solutions to improve the original algorithm from [14]. In particular, Glowacz et al. presented a more efficient rank estimation tool at FSE 2015, that is based on a simple convolution of histograms and allows obtaining tight bounds for the key rank of (even large) keys [8]. A comparable result was developed independently by Bernstein et al. [1].[1] In parallel, Ye et al. investigated an alternative solution based on a weak Maximum Likelihood (wML) approach [15], rather than a Maximum Likelihood (ML) one for the previous examples. They additionally combined this wML approach with the possibility to approximate the security of an implementation based on "easier to sample" metrics, e.g. starting from the subkey Success Rates (SR) rather than their likelihoods, typically. Eventually, at Eurocrypt 2015 Duc et al. described a simple alternative to the algorithm of Ye et al. and provided an "even easier to sample" bound on the subkey SR, by exploiting their formal connection with a Mutual Information metric [5].

This state-of-the-art suggests the informal classification of approaches to key rank estimation in Table 1, based on whether the algorithms consider ML or wML adversaries/evaluations, and whether they are sampling-based or metric-based.[2] Looking at this table, it is clear that from the "quality of evaluation" point-of-view, the sampling-based ML approach is the most accurate. Indeed, the wML approach corresponds to a suboptimal adversary, hence can only lead to a Sampled Lower Bound (SLB). Besides, a straightfroward metric-based evaluation can only lead to a Metric-based Lower Bound (MLB), because of a Jensen inequality (i.e. since it combines the average success rates of several subkeys

[1] Their "Polynomial Rank Outlining" algorithm can be viewed as similar to the FSE 2015 one, by considering the multiplication of two polynomials as the convolution of coefficient vectors, and the coefficient in the polynomials as histogram counts.

[2] Quite naturally, metrics such as the subkey success rates also need to be sampled somehow. So the term "metric-based" only refers to the type of inputs provided to the rank estimation algorithms, to be compared with the sampling-based approach where the sampled probabilities output by a side-channel attack are used directly.

which lower bounds the average success of combined attacks against several subkeys). As a result, one can naturally question the interest of these alternative approaches, for which we can put forward two main motivations:

1. For the wML approach, in addition to the rank estimation, it outputs an *effort distributor* which indicates how the enumeration effort should be spread over the subkeys, and therefore directly leads to a suboptimal but parallel enumeration algorithm with minimum memory requirements (which is in contrast with the optimal but more expensive and serial solution in [13]).
2. For the metric-based approach, the main motivation is to speed up the evaluations, i.e. to run the rank estimation once based on the subkey metrics in order to obtain a master key metric, rather than running it many times to obtain many master key rank samples to be "metrified" (e.g. averaged).

In view of this state-of-the-art, our contribution is threefold. We start by investigating two cases from Table 1 that were not experimented so far. Namely, we first show that the simple algorithm from [5] (that exploits a combination of subsampled metrics) naturally applies in a sampling-based setting, leading to the previously mentioned SLB that directly suggests a suboptimal but parallel enumaration strategy. Second, we provide a Metric-based Upper Bound (MUB) on the SR which allows very fast security evaluations and nicely completes the lower bound provided by the metric-based wML approach. Third and eventually, we provide an experimental evaluation of these different solutions, allowing to comprehend their pros and cons. In particular, our experiments include an analysis of the number of cores that would be necessary for the parallel wML enumation to start gaining advantage over the serial approach of [13].

Related works. Two recent works related to key enumeration and rank estimation will appear in the proceedings of SAC 2015 [2] and ASIACRYPT 2015 [10]. More precisely, they primarily contribute to key enumeration algorithms that can be parallelized, based on simple backtracking procedures for the SAC proposal, and by casting the enumeration problem as a knapsack for the ASIACRYPT one. The second paper additionally proposes an alternative solution for rank estimation, which leads to similar outcomes as the FSE 2015 algorithm.

Notations. We next use sans serif font for functions (e.g. F), calligraphic fonts for sets (e.g. $\mathcal{A}$) and denote the ith element of a list L by $L[i-1]$.

2 Errors and bounds

As first discussed in [14], estimating the rank of a key essentially amounts to performing a mix of depth-first and breadth-first searches in a high-dimensional space representing the key probabilities of a side-channel attack, which turns out to be computationally hard as the key rank increases. It implies that with current computational means and knowledge, it is typically impossible to know the exact position of keys ranked beyond $2^{80} - 2^{90}$ in such high-dimensional spaces. As a result, the only solution is to estimate the rank and to bound the

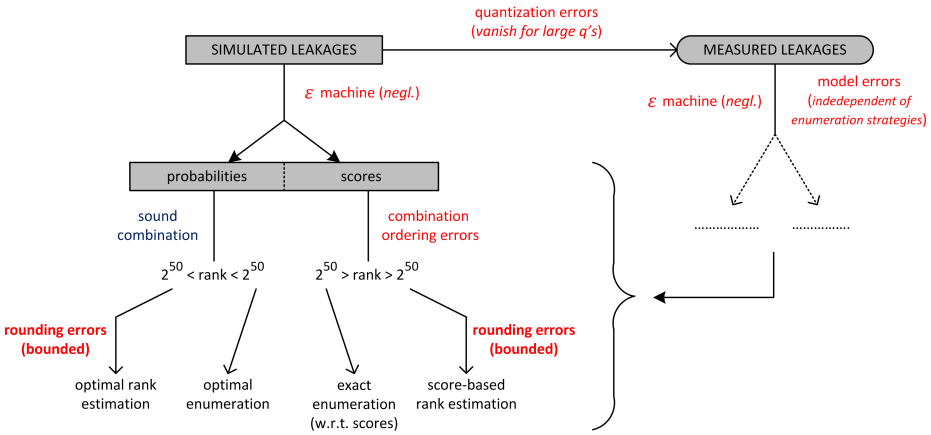

Fig. 1. Errors & bounds in key enumeration and rank estimation.

error on this rank estimation. Since there are in fact several types of errors that can be introduced in the course of a side-channel attack, we start this paper with a brief discussion about which are the important errors in key enumeration and rank estimation, and how they can be efficiently bounded by evaluators.

For this purpose, we summarize the main intuitions in Figure 1, starting with "simulated leakages" (i.e. mathematically-generated ones for which we exactly know the distribution). In this case, the evaluator will start by producing a "distinguishing vector" for which there are only small numerical errors (due to the ϵ machine), that can generally be neglected. More significantly, this distinguishing vector can be made of probabilities (e.g. for TA and LR based attacks) or scores (e.g. for DPA and CPA). This is an important distinction, since in the first case the evaluator will be able to combine the information of independent subkeys on a sound basis, whereas in the second case he will potentially introduce "combination ordering errors". For example, say we have two lists of subkey probabilities $[p_1, p_2]$ and $[p_a, p_b]$ with $p_1 > p_2$ and $p_a > p_b$ (resp. scores $[s_1, s_2]$ and $[s_a, s_b]$ with $s_1 > s_2$ and $s_a > s_b$). Whereas it is clear (using both scores and probabilities) that the best-rated key corresponds to the pair $\{p_1, p_a\}$ or $\{s_1, s_a\}$ and the worst-rated one corresponds to the pair $\{p_2, p_b\}$ or $\{s_2, s_b\}$, only probabilities allow comparing the pairs $\{p_1, p_b\}$ and $\{p_2, p_a\}$ (by multiplying the probabilities). Intuitively, the key property here is that a probability of 1 for a subkey implies a success rate of 1 for this subkey, which therefore allows "guiding" the key enumeration and rank estimation algorithms.[3] Given this dif-

[3] By contrast, one could typically imagine a scenario where the scores obtained from a CPA lead to a correlation 0.2 for a subkey that is known with high confidence and the same correlation of 0.2 for a subkey that is not know at all – because of different Signal-to-Noise Ratios. In this case, the key enumeration and rank estimation algorithms will not be able to list keys optimally. Quite naturally, it is possible to mitigate such issues by outputing the p-values of the CPA distinguisher, but this re-

ference, it is then possible to enumerate optimally based on probabilities, or to enumerate exactly w.r.t. some scores, whenever the key rank is computationally reachable (e.g. when it is smaller than 2^{50} on the figure). By contrast, if the rank is too high for the keys to be enumerated, the only option is key rank estimation, where additional rounding errors will appear. Interestingly, it is shown in [1, 8, 10] that the errors due to this rounding can be kept small comparatively to the key rank. For example, rank estimation algorithms allow claiming that a key is rated among ranks 2^{80} and 2^{81} (or even tighter bounds) which is perfectly relevant in a side-channel evaluation context, since it indicates its remaining computational security. Admittedly, this does not mean that the rank estimation itself is perfectly accurate, since the only thing we can guarantee in our example is that the key rank is in a (large) set of size $2^{81} - 2^{80} = 2^{80}$.

Next, and when moving to the practically relevant case of measured leakages, two additional types of errors can appear. First, the measurements are typically operated with a sampling device, with 8 bits to 12 bits of accuracy, which causes quantization errors. However, the impact of these errors tends to vanish as the number of measured leakages q in the attack increases (since the cardinality of the joint leakages then increases exponentially in q, e.g. 256^q or 4096^q for our 8-bit and 12-bit cases). More importantly, model errors can be introduced, i.e. discrepancies between the true leakage distribution and the one exploited by the adversary. However, these errors are in fact independent of the enumeration strategies. So indeed, when speaking about optimal enumeration, the optimality is relative to the leakage models, which we expect to be sound in worst-case security evaluations (and can be guaranteed with leakage certification tests [6]). But for the rest, the main errors that we consider and bound in this paper are the rounding ones, that are unavoidable when estimating large ranks.

3 Algorithm specification

3.1 Algorithms inputs

Details on how a side-channel attack extracts information from leakage traces are not necessary to understand the following analysis. We only assume that for a n-bit master key k, an attacker recovers information on N_s subkeys $k_0, ..., k_{N_s-1}$ of length $b = \frac{n}{N_s}$ bits (for simplicity, we assume that b divides n). The side-channel adversary uses the leakages corresponding to a set of q inputs $\mathcal{X}_q$ leading to a set of q leakages $\mathcal{L}_q$. As a result of the attack, he obtains N_s lists of 2^b probabilities $P_i = \Pr[k_i^* | \mathcal{X}_q, \mathcal{L}_q]$, where $i \in [0, N_s - 1]$ and k_i^* denotes a subkey candidate among the 2^b possible ones. As just mentioned, TA and LR based attacks directly output such probabilities. For other attacks such as DPA or CPA, one can either go for score-based rank estimation or use Bayesian extensions [13].

quires making additional assumptions on its distribution, and eventually corresponds to a type of profiling which would then allow evaluators to directly estimate probabilities. So in general, we believe it is advisable to directly use probability-based distinguishers for optimal key enumeration and rank estimation algorithms.

Based on these notations, our sampling-based approaches to rank estimation will require two additional inputs. First the lists of probabilities might be turned into lists of log probabilities, denoted as $LP_i = \log(P_i)$. Next, these lists of probabilities will also be be translated into lists of cumulative probabilities as follows. First a sorted list is produced as: $SP_i = \mathsf{sort}(P_i, \text{decreasing})$. Then, the list of cumulative probabilities is derived as CP_i such that $CP_i[j] = \sum_{jj=0}^{j} SP_i[jj]$.

As for the metric-based approaches, they will require the subkey success rates of order d introduced in [12], which simply correspond to the probability that the correct subkey is rated among the first d ones by an attack. In the following, we will denote the lists of 2^b success rates of order d (with $d \in [1, 2^b]$) as SR_i. We will additionally denote the "derivative" of these success rates (i.e. the probabilities that a key is rated exactly at position d) as ΔSR_i such that $\Delta SR_i[d] = SR_i[d] - SR_i[d-1]$ (with $SR_i[0]$ set to 0 by definition).

3.2 Toolbox

For convenience, we now introduce a number of generic tools that will be used to simplify the description of our following algorithms, and can be found (or easily developed) in most mathematical programming languages.

Linear histograms. The function $H = \mathsf{hist\_lin}(LP, \mathsf{bins})$ creates a standard histogram from a list of (e.g.) log probabilities LP and linearly-spaced bins bins.

Logarithmic indexing. Given a list of (e.g.) probabilities P indexed from 1 to 2^b and a width w, the function $LI = \mathsf{index\_log}(P, w)$ groups the elements in the list by the logarithm of their indexes according to a width w. An example of such a logarithmic indexing with $w = 1$ is illustrated in the left part of Figure 2, where we have $LI[0] = P[0]$, $LI[1] = P[1] + P[2]$ and so on. More precisely $LI[k] = \sum_{j \in \mathcal{E}_k} P[j-1]$ with $\mathcal{E}_k = \{j \in \mathbb{N} \cap [2^{k \cdot w}, 2^{(k+1) \cdot w}[\}$ and $k \in [1, \frac{log_2(l)}{w}]$.

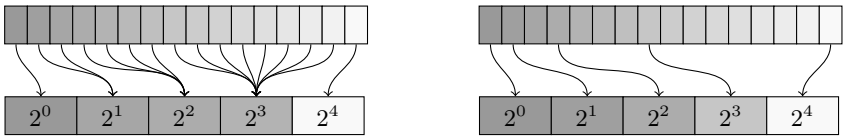

Fig. 2. Logarithmic indexing with $w = 1$ (left) and logarithmic downsampling (right).

Linear downsampling. Given a list of (e.g.) cumulative probabilities CP or success rates SR, the function $\mathsf{downs\_lin}(CP, N_{max})$ (resp. $\mathsf{downs\_lin}(SR, N_{max})$) reduces the size of CP (resp. SR) by selecting N_{max} linearly-spaced samples over the probabilities (resp. success rates). This downsampling is illustrated on the Y axis of Figure 3, where we keep the values sampled by the blue dotted lines.

Logarithmic downsampling. Given a list of (e.g.) cumulative probabilities CP or success rates SR, the function $\mathsf{downs\_log}(CP)$ (resp. $\mathsf{downs\_log}(SR)$) reduces

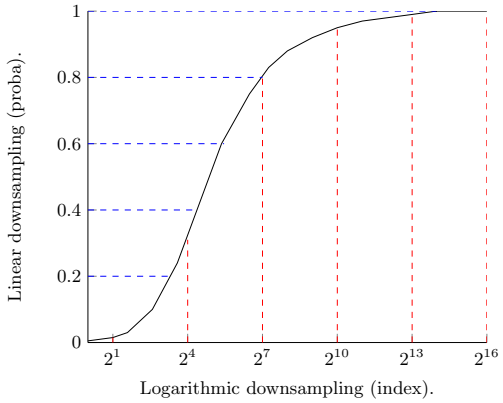

Fig. 3. Illustration of dowsampling possibilities.

the size of CP (resp. SR) by sampling logarithmically over the indexes, and selecting one sample per power of two of the keys (resp. orders). An example of this downsampling is given on the X axis of Figure 3, where we keep the values sampled by the red dotted lines (and in the right part of Figure 2).

Convolution. This is the usual convolution algorithm which from two histograms H_1 and H_2 of sizes n_1 and n_2 computes $H_{1,2} = \mathsf{conv}(H_1, H_2)$ where $H_{1,2}[k] = \sum_{i=0}^{k} H_1[i] \times H_2[k-i]$. It can be implemented efficiently with a FFT.

Linear combination. Given two lists of (e.g.) cumulative probabilities CP_1 and CP_2 of sizes n_1 and n_2, the linear combination $CP_{1,2} = \mathsf{comb\_lin}(CP_1, CP_2)$ can be computed as $CP_{1,2}[k] = max\{CP_1[i] \times CP_2[j] \mid i \cdot j = k\}$ for $k \in [0, n_1 \cdot n_2 - 1]$. Such a combination assumes that the underlying relation between the indexes is multiplicative, which is the case for cumulative probabilities. As for the linear downsampling, this linear combination similarly applies to success rates.

Logarithmic combination. Given two lists of (e.g.) cumulative probabilities CP_1 and CP_2 of sizes n_1 and n_2, the logarithmic combination $CP_{1,2} = \mathsf{comb\_log}(CP_1, CP_2)$ can be computed as $CP_{1,2}[k] = max\{CP_1[i] \times CP_2[j] \mid i + j = k\}$ for $k \in [0, n_1 + n_2 - 1]$. Such a combination assumes that the underlying relation between the indexes is additive, which is the case when a list has been previously processed by $\mathsf{downs\_log}$ (so it again applies to success rates as well).

Effort distributor. In general, the (linear or logarithmic) downsampling and combination procedures, output lists with reduced number of samples. But as mentioned in introduction, for sampling-based wML rank estimation, it may additionally be interesting to output a so-called effort distributor. In this case, these procedures (applied to lists of cumulative probabilities) will also input/output the indices of the retained samples, which indicates how much effort should be devoted (i.e. how many keys should be tested) for each subkey.

	Sampling-based	Metric-based
wML	`Input:` subkey cumulative prob. `Approx:` log. downsampling `Aggreg:` log. combination	`Input:` subkey success rates `Approx:` log. downsampling `Aggreg:` log. combination
ML	`Input:` subkey log. probabilities `Approx:` linear histograms `Aggreg:` convolution	`Input:` subkey SR derivative `Approx:` log. indexing `Aggreg:` convolution

Table 2. Tools used in our rank estimation algorithms.

3.3 Preprocessing

As part of our evaluations, we will also consider the preprocessing which consists of merging m lists of probabilities P_i of size 2^b in order to generate a larger list $P'_i = \mathsf{merge}(P_0, P_1, \ldots, P_{m-1})$, such that P'_i contains the $2^{m \cdot b}$ product of probabilities of these lists. Taking again our notations where the n bits of master key are split in N_s subkeys of b bits, it amounts to split them into a $N'_s = N_s/m$ subkeys of $m \cdot b$ bits. This process is in fact similar to the previously described linear combination one. We just use the term merging and denote it as merge_m when it is used for pre-processing (to be consistent with previous works).

3.4 Overview of the tools used in different approaches

Before describing our four rank estimation algorithms in detail, Table 2 provides a quick overview of the different tools that we exploit in the different approaches. As clearly seen from the table, each of the algorithms is mainly characterized by a type of input together with an approximation and an aggregation procedure. Note that for the sampling-based ML one, we use exactly the FSE 2015 algorithm [8]. This choice is motivated by the fact that it is the fastest one published so far.[4] As for the metric-based wML approach, we use a slight variant of the EUROCRYPT 2015 heuristic [5], where we replace the linear downsampling by a logarithmic downsampling (which turns out to be more efficient).[5]

3.5 Sampling-based rank estimation

We use exactly the solution proposed at FSE 2015, recalled in Algorithm 1. As detailed in [8], it allows reaching tight bounds for the key ranks (for key sizes up to 1024 bits), typically setting the log of the ratio between the upper and lower bounds to less than one bit. In the following, we will consider this Sampled Estimation (SE) as a reference (i.e. our most accurate rank estimation).

[4] Together with its analog in [1] which would yield very similar performances.

[5] The algorithm by Ye et al. could be used as a sightly more accurate alternative. However, as our proposal, it can only provide a lower bound on the success rate because it is based on a wML approach. We focused on the EUROCRYPT 2015 heuristic because of its simplicity and connections with the other solutions of Table 2.

Algorithm 1 Sampling-based rank estimation (SE).

Input: The log proba. of the master key lpk, lists of log proba. LP_i, and bins bins.
Output: An estimation of the master key rank.

$H_{curr} = \mathsf{hist\_lin}(LP_0, \mathsf{bins})$
for $i = 1$ to $N'_s - 1$ **do**
  $H_i \leftarrow \mathsf{hist\_lin}(LP_i, \mathsf{bins})$
  $H_{curr} \leftarrow \mathsf{conv}(H_{curr}, H_i)$
end for
$rank \leftarrow log_2(\sum_{i=\mathsf{bins}(lpk)}^{N'_s \cdot N_{bin} - (N'_s - 1)} H_{curr}[i])$
return $rank$

Note that if the evaluator's goal is to estimate a success rate of a given order for the master key (or more generally to build a security graph such as described in [14]), he will need to repeat Algorithm 1 several times in order to obtain many estimates of the key rank that he will average afterwards, i.e. a quite costly task which may motivate the use of metric-based approaches to rank estimation.

3.6 Sampling-based success lower bound

The ML rank estimation actually corresponds to an optimal enumeration strategy such as described in [13]. In this context, the adversary/evaluator produces a list of (master) key candidates in decreasing order of likelihood that he can test. Quite naturally, such an optimal enumeration implies that (most of the times) when moving from the ith most likely key to the $i+1$th one, the indices of several subkeys will vary. In general, such an approach therefore has significant memory requirements (corresponding to the size of the lists to produce and send to the "key testing" hardware). The wML approach introduced in [15] actually corresponds to a much simpler (greedy) strategy where, when moving from the ith most likely key to the $i+1$th one, the indices of only one subkey are modified. While obviously suboptimal, this strategy has the significant advantage that it has essentialy no memory requirements, i.e. one only needs to store the number of candidates to enumerate per subkey (i.e. the effort distributor), and is straightforward to parallelize. Its simplified version (inspired from [5] but replacing the linear downsampling and combination by logarithmic ones) is described in Algorithm 2. It outputs a SLB on the (single) attack's success together with an effort distributor. Since the combination is done using a logarithmic downsampling, any output of the effort distributor $ED[i]$ (with $i \in [0, n]$) is a N'_s-element list corresponding to an effort of 2^i, where every element of the list is a subkey effort. For simplicity, we will say that a key k is in the effort distributor $ED[i]$ if the rank of all its subkeys is lower than the corresponding subkey effort in the list $ED[i]$. As in the SE approach, estimation of a success rate thanks to this approach requires repeating attacks (and Algorithm 2) several times.

Algorithm 2 Sampling-based success lower bound (SLB).

Input: Lists of cumulative proba. CP_i and actual subkeys $k = \{k_0, ..., k_{N'_s-1}\}$.
Output: An upper bound on the master key rank.

$CP_{curr} \leftarrow \mathsf{downs\_log}(CP_0)$
$ED_{curr} \leftarrow \varnothing$
for $i = 1$ to $N'_s - 1$ **do**
  $CP'_i \leftarrow \mathsf{downs\_log}(CP_i)$
  $CP_{curr}, ED \leftarrow \mathsf{comb\_log}(CP, CP'_i, ED_{curr})$
  $ED_{curr} = ED_{curr} \cup ED$
end for
$log\_rank \leftarrow 0$
while $k \notin ED[log\_rank]$ **do**
  $log\_rank \leftarrow log\_rank + 1$
end while
return 2^{log_rank}

3.7 Metric-based success rate lower bound

As previously mentioned, a possible drawback of the previous sampling-based approaches is that producing success rate curves based on them requires running the rank estimation algorithms several times. One natural solution to avoid this drawback is to consider metric-based approaches, where the evaluator does not directly deal with DPA outcomes (i.e. subkey log probabilities or cumulative probabilities), but with the success rates of every subkey considered independently. Concretely, this approach highly resembles the one in the previous section (only its inputs are different). However, it does not only correspond to a suboptimal wML strategy: it also includes an additional loss of tightness due to a Jensen inequality. In the following, we refer to this strategy, described in Algorithm 3, as the Metric-based success rate Lower Bound MLB approach.

Algorithm 3 Metric-based success rate lower bound (MLB).

Input: Lists of success rates SR_i.
Output: A lower bound on the master key success rate.

$SR_{low} \leftarrow \mathsf{downs\_log}(SR_0)$
for $i = 1$ to $N'_s - 1$ **do**
  $SR^i_{low} \leftarrow \mathsf{downs\_log}(SR_i)$
  $SR_{low} \leftarrow \mathsf{comb\_log}(SR_{low}, SR^i_{low})$
end for
return SR_{low}

3.8 Metric-based success rate upper bound

The metric-based approach in the previous section is very convenient to manipulate (i.e. simple and efficient) but it only provides a lower bound for the success rate. This section finally brings its natural complement, i.e. an easy-to-manipulate metric-based upper bound for this success rate. For this purpose, the main technical ingredient is to replace the logarithmic downsampling of the success rate curves (where only the sample with maximum success rate is kept) by a logarithmic indexing of the derivative success rates with width w (which are then combined thanks to the convolution tool). As a result, we obtain the Metric-based success rate Upper Bound (MUB) described in Algorithm 4.

Algorithm 4 Metric-based success rate upper bound (MUB).

Input: Lists of derivative success rates ΔSR_i and bin log-width w.
Output: An upper bound on the master key success rate.

$LI_{curr} = \mathsf{index\_log}(\Delta SR_0, w)$
for $i = 1$ to $N'_s - 1$ **do**
  $LI_i \leftarrow \mathsf{index\_log}(\Delta SR_i, w)$
  $LI_{curr} \leftarrow \mathsf{conv}(LI_{curr}, LI_i)$
end for
for $i = 0$ to n **do**
  $SR_{up}[i] \leftarrow \sum_{\frac{1}{w} \cdot i}^{\frac{1}{w} \cdot (i+1)-1} LI_{curr}[j]$
end for
return SR_{up}

In practice, the lower w is, the tighter the bound is. For simplicity, we will assume w is at most equal to 1 and that $\frac{1}{w}$ must fall into the integer domain (our following experiments will take $w = 0.001$). An explanation why this algorithm indeed gives an upper bound for the success rate is given in the Appendix A.

4 Experiments

In this section, we provide experiments to illustrate our four methods. For this purpose, we considered the standard case study (also used in previous evaluations of rank estimation algorithms) of a simulated AES-128 implementation, for which every encryption leaks 16 samples denoted as $l_i = \mathsf{L}(\mathsf{S}(x_i, k_i)) + N$ with $i \in [0, 15]$, where L is the leakage function, S is the AES S-box and N is a random noise following a Gaussian distribution. Concretely, we tried different L's (Hamming weight, linear, non-linear) which had no impact on our observations (as previously reported). We also tried having different L's and N's for the different subkeys with the same conclusion. So because of place constraints, we only report experiments for a single leakage function and noise level. Based on these leakages, we implemented a simple univariate TA, in order to produce the lists of probabilities defined in Section 3.1. For all our experiments, our evaluation metrics were obtained by launching 1000 independent attacks.

4.1 Sampling-based evaluations.

A comparison of the (tight) rank estimation bound from FSE 2015 (using 10,000 bins) with the sampling-based lower bound of Algorithm 2 is reported in Figure 4, for different number of measurements (hence security levels). As expected, we see that the wML approach can only provide a lower bound on the success rate. The SLB curves additionally exhibit the (positive) impact of merging the subkey probabilities prior to the evaluation for the quality of the lower bound.

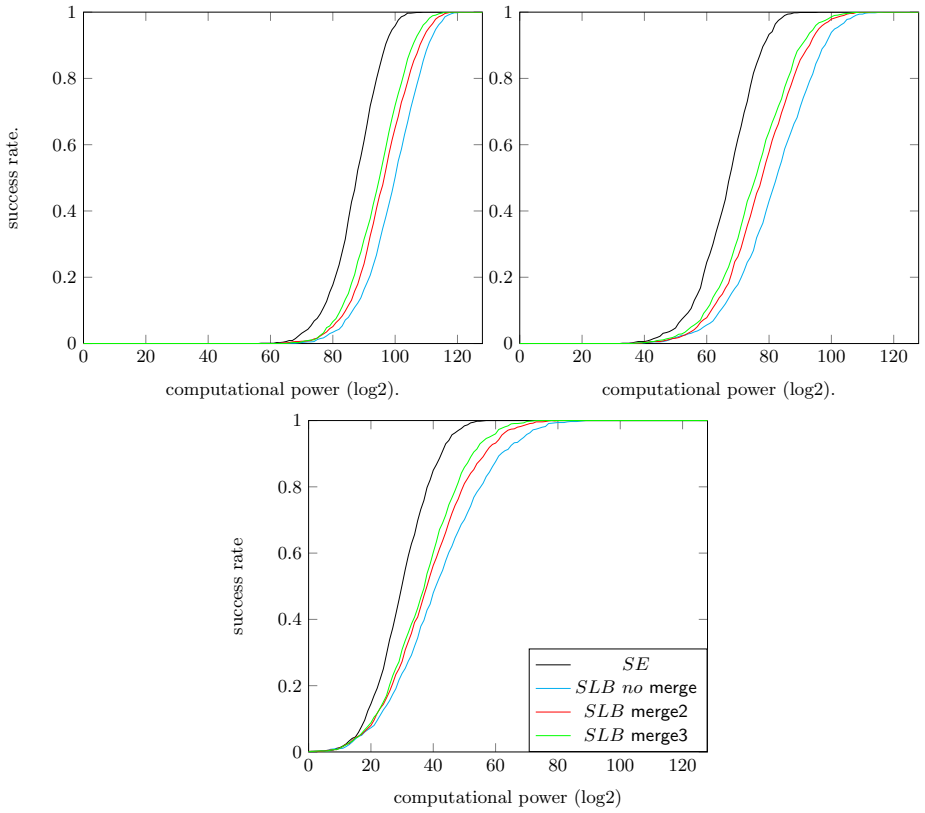

Fig. 4. Sampling-based rank estimations for different security levels. Upper left: 80-bit security level. Upper right: 65-bit security level. Bottom: 30-bit security level.

Besides, and as discussed in Section 3.6, Algorithm 2 can also be used to provide a simple (yet suboptimal) enumeration strategy that can be parallelized. So assuming that the optimal enumeration algorithm in [13] is implemented in a purely serial fashion (which is correct as soon as generating the list of most likely keys becomes more expensive than testing them), another interesting experiment is to measure the gap between these two approaches. For this purpose, Figure 5 shows the number of cores needed to reach the success rate of a (purely serial)

ML enumeration with the suboptimal but parallel wML heuristic of Algorithm 2, for the same experiments as reported in Figure 4. For example, we see that for the $\approx$ 80-bit security level (upper left plot) and assuming a merge_3 preprocessing, we can reach the same 80% success rate with both approaches assuming that the parallel enumeration exploits 2^{10} computing cores. This result is interesting since it suggests that for organized adversaries, the enumeration overheads of a wML enumeration strategy may be compensated by computing capabilities. Note that these results assume that it takes as much time for the enumeration algorithm to output a candidate as it takes to test it, which is a conservative estimate if one assumes that the testing can be performed by some dedicated hardware.

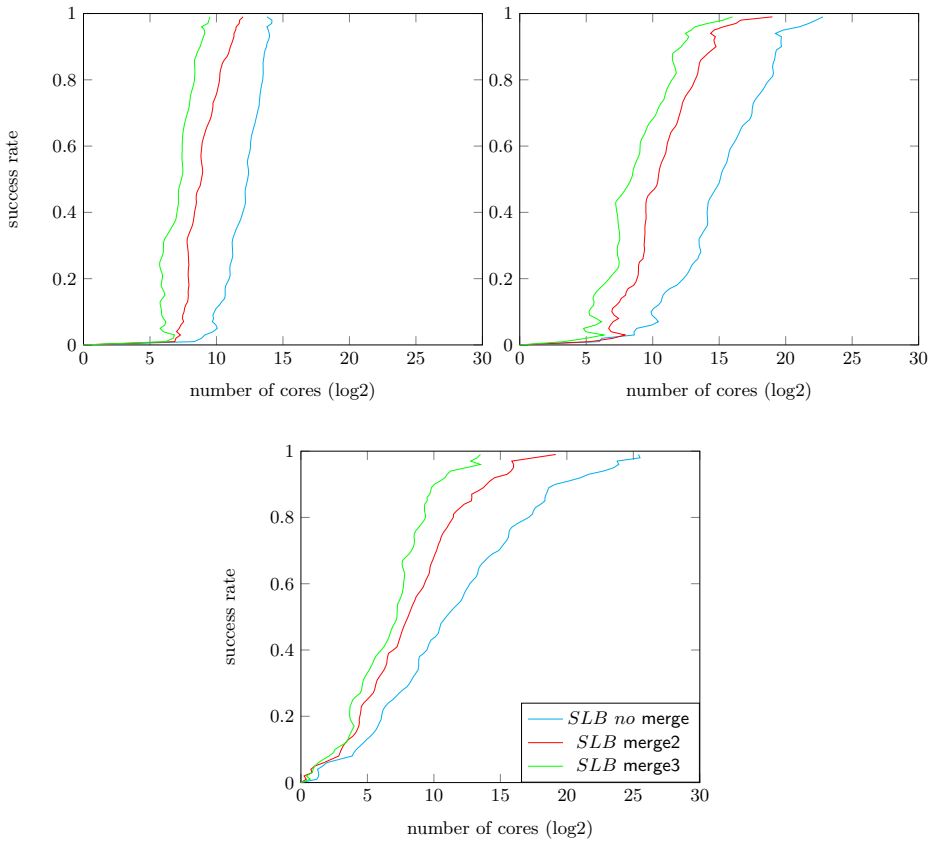

Fig. 5. Number of cores needed to reach the success rate of a (serial) ML enumeration with the suboptimal but parallel wML heuristic of Algorithm 2. Upper left: 80-bit security level. Upper right: 65-bit security level. Bottom: 30-bit security level.

4.2 Metric-based evaluations

We now describe the metric-based counterpart to the previous sampling-based experiments, reported in Figure 6. We again use the estimate from Algorithm 1

as a reference, and this time compare it with Algorithms 3 and 4. Note that for comparison purposes, our subkey success rates were directly sampled (i.e. not bounded as suggested in [5]). This allows us to gauge the impact of the different enumeration strategies for similar inputs. As expected again, Algorithms 3 and 4 indeed provide lower and upper security bounds. However, while those metric-based solutions are especially convenient from an evaluation time point of view (see the following discussion in Section 4.3), it is also clear from the figure that none of them provides tight approximations of the security level. This is somehow expected as well since the MLB curves correspond to an adversary who is weakened both by a wML approach and a Jensen inequality, while the MUB one is based on the rough bound discussed in Appendix A. Note that we used $w = 0.001$: improvements were not noticeable anymore for lower widths.

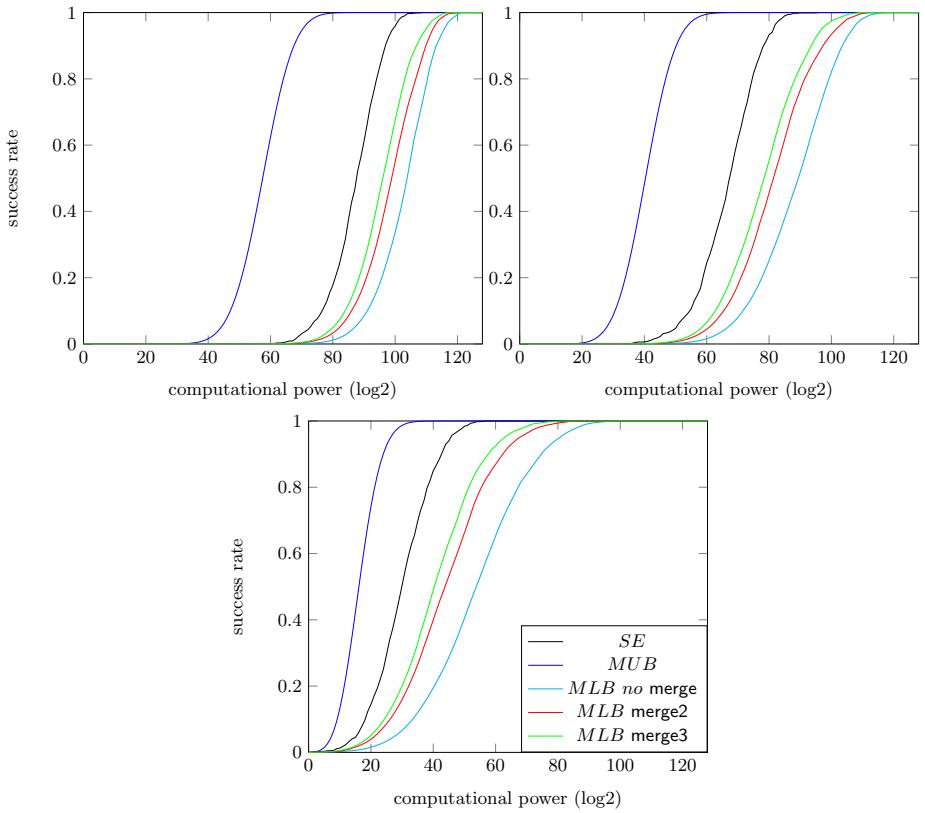

Fig. 6. Metric-based rank estimations for different security levels. Upper left: 80-bit security level. Upper right: 65-bit security level. Bottom: 30-bit security level.

Interestingly though, these conclusions are moderated when looking at the complete security graphs obtained with these different approaches, reported in Figure 7 (i.e. plots of the success rate in function of the number of measure-

ments and time complexity of the attacks [14]). Namely, the upper and lower security bounds that correspond to the MLB and MUB curves are not so optimistic/pessimistic from this point-of-view. This is in fact mainly due to the weaker impact of enumeration (compared to measurements) in side-channel analysis. That is, adding a new measurements reduces the security exponentially, while enumerating only does it polynomially. So overall, despite being based on relatively rough heuristics, Algorithms 3 and 4 can in fact be used to obtain a reasonable view of the security level of leaking implementation. The latter observation is in line with the experiments in [15] where the gap between the ML and wML approach appeared to be limited when looking at similar figures.

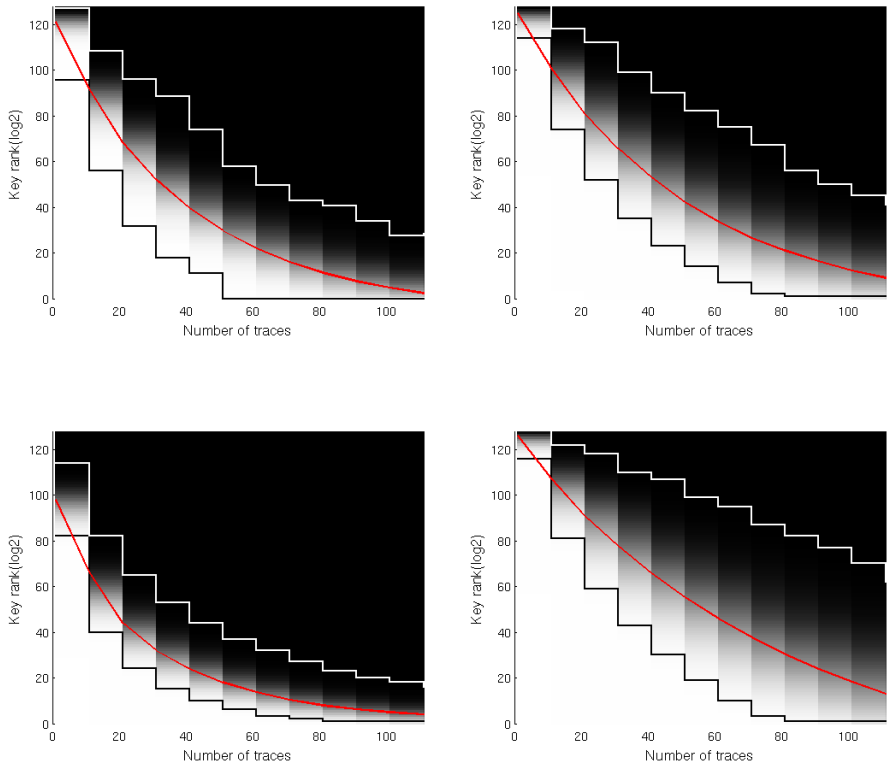

Fig. 7. Security graphs. Upper left: SE methods, upper right: $MLB + \mathsf{merge3}$ preprocessing, lower left: MUB (bottom left), lower right: MLB with no merging.

4.3 Time complexity

In order to provide a complete view of the rank estimation problem, we finally provide a brief discussion of the time complexity of these approaches. Note that this discussion mainly makes sense in the context of an evaluation, where many attacks have to be launched in parallel (while in a real attack context, a single

enumeration is usually enough). Taking our exemplary experiments, we performed 1000 attacks for each security level. In this case, the SE computation required approximatively 5 minutes proceed (per number of traces), with 10,000 initial bins. By contrast, only one second was needed for computing MUB with $w = 0.001$ (so a factor 300 less). In addition, the MLB computations required 1 second, 3 seconds and 5 minutes with the merge1, merge2 and merge3 preprocessings. (Note that the time to compute the subkeys success rates are included in these MUB and MLB times). Eventually, the SLB computation required 3 second, 38 seconds and 2 hours with the merge1, merge2 and merge3 preprocessings. So overall, we can conclude that in the simple context we investigated, the sampling-based approach leads to very accurate results in a sufficiently short time. But in contexts where the evaluation cost increases, simpler metric-based bounds can gain more interest. Such bounds are anyway usful for fast prototyping since they can be combined with the Mutual Information based evaluations in [5]. In this case, we gain the additional advantage that the analyis does not have to be repeated for several number of measurements (i.e. the success rate for any such number is derived from a Mutual Information metric).

5 Conclusion

This paper clarifies the link between various approaches to rank estimation. We divided our investigations in two parts, one sampling-based that is closer to the goals of concrete adversaries, one metric-based that is closer to evaluator's ones. We additionally discussed the interest of wML enumeration strategies in order to make this task easily parallelizable. Our experiments suggest that tight rank estimation can be efficiently implemented thanks to a simple algorithm from FSE 2015. But they also highlight that the wML and metric-based approaches can gain interest (e.g. for adversaries with high parallel computing power and to further speed up the evaluation time, respectively). In view of the efficient solutions existing for rank estimation, an important scope for further research remains to find new algorithms for optimal and parallel enumeration. First proposals in this direction can be found at SAC 2015 [2] and ASIACRYPT 2015 [10].

Acknowledgements. F.-X. Standaert is a research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the European Commission through the ERC project 280141 (CRASH).

References

1. Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptology ePrint Archive*, 2015:221, 2015.
2. Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and memory-efficient key recovery in side-channel attacks. *IACR Cryptology ePrint Archive*, 2015:795, 2015.
3. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004, Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.
4. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.
5. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *LNCS*, pages 401–429. Springer, 2015.
6. François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 459–476. Springer, 2014.
7. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.
8. Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schueth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. *IACR Cryptology ePrint Archive*, 2014:920, 2014.
9. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO '99, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
10. Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. How to enumerate your keys accurately and efficiently after a side channel attack. *IACR Cryptology ePrint Archive*, 2015:689, 2015.
11. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *CHES 2005, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *LNCS*, pages 30–46. Springer, 2005.
12. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT 2009, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.
13. Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012, Wind-*

sor, ON, Canada, August 15-16, 2012, Revised Selected Papers, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.
14. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.
15. Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? how to determine whether limited side channel information enables key recovery. In Marc Joye and Amir Moradi, editors, *CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2014.

A Bound from the metric-based maximum likelihood

We start from the observation that if the lists of probabilities obtained for each S-box are independent, $rank(k_0) = d_0$, ..., $rank(k_{N'_s}) = d_{N'_s}$ and $rank(k) = d$, then $d \geq \prod_i d_i$. When estimating a master key success rate, it implies that:

$$\widehat{SR}[d] \leq \sum_{d_i, \prod_i d_i \leq d} \Delta SR_0[d_0] \times \cdots \times \Delta SR_{N'_s}[d_{N'_s}].$$

Algorithm 4 just computes such a sum of products of derivative success rates.

Note that the independence condition for the S-box probabilities in this bound is not the same as the usual independent leakages considered, e.g. in leakage-resilience cryptography [7], which is a purely physical condition. Here, we only need that the adversary considers the subkeys independently.