Project ID: 731591, Horizon 2020
http://www.reassure.eu

REASSURE

Deliverable D2.3

Shortcut Formulas for Side-Channel Evaluation

Editor:	N. Debande (Idemia)
Deliverable nature:	R
Dissemination level: (Confidentiality)	PU
Delivery date:	February, 2018
Version:	0.1
Total number of pages:	40
Keywords:	shortcuts, side-channel attacks, evaluations

Horizon 2020
European Union funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 731591.

Executive summary
This deliverable surveys a number of approaches to shortcut the effort required to assess implementations and devices with respect to their susceptibility to side channel attacks. Our approach aligns with the divide and conquer nature of most side channel attacks and hence we touch on shortcuts that apply to the distinguisher statistics (the divide step) and the key rank (the conquer step).

We notice that shortcuts make significant assumptions about leakage characteristics (in particular independence of leakages and equal variances) that do not hold in many of the more challenging device evaluation scenarios. In addition, it is not always possible to characterise the signal and noise as required: early on in a design this information is not yet available, whereas later on in an evaluation within a set scheme the evaluator often cannot turn off countermeasures to establish the nature of the "original" signal. When it comes to key rank computations it has been shown that the variance of the key rank is huge in those cases where the key rank is most relevant. The tightness by which the average rank is estimated is thus not so important as long as the estimate also gives some information about the spread of the rank.

Within REASSURE we set ourselves the challenge to assess how such shortcut approaches could nevertheless be leveraged to improve evaluations. This deliverable attempts to highlight at what stage shortcuts are particularly appropriate. Secondly, some shortcuts will be implemented within WP3.2 in the context of the REASSURE simulation tool.

List of authors

Company	Author
Idemia	N. Debande
Bristol	E. Oswald
UCL	R. Poussier
UCL	F.-X. Standaert

Revision history

Revision number	Date	Comment
0.1	February 2018	First Draft

Contents

1 Introduction

Designing devices and corresponding software in such a way that they are resilient against a wide range of leakage based attacks requires to make (many) informed choices regarding the trade-off between using one or several of the multitude of implementation options that are available for cryptographic algorithms. Of course, it is possible to create implementations of many options, and then attack them. The obvious problem with this strategy is that it requires time and expertise to craft (*i.e.* optimise) implementations, and design and carry out appropriate attacks. In order to speed this process up, a number of "shortcuts" have been designed over the years (e.g. estimation of a loose lower bound on the number of required measurements for a specific type of attack based on knowledge of signal and noise). *In this deliverable we define a shortcut as any technique that enables efficient estimation of attack outcomes.*

This deliverable is a white paper that aims to give a brief overview of the state of the art of shortcut formulas and to explain how to use them to make statements about leakage properties of implementations. Thus in this deliverable we aim to give a balanced picture of the state of the art and thus choose references to be representative rather than exhaustive.

1.1 Motivation

There are several key considerations that motivate research into shortcut formulas in general and therefore also this deliverable.

- The decision of choosing a suitable combination of countermeasures must be made at a *very early stage of the design* to avoid costly design iterations. Therefore, investigating the effect of (tuneable) parameters on attack outcomes is helpful for designers.
- The quality of evaluations may be improved by supplying meaningful estimates for attacks as *"supporting evidence"* for observed attack performance.

1.1.1 Supporting Early Decision Making

Designing and implementing cryptographic algorithms (and protocol architectures on top) requires many decisions: the choice of algorithms, what to put in hardware and what to do in software, and eventually implementation specifics (representations of e.g. finite field elements, reliance on look-up tables, degree of parallelism, *etc.*). All such details impact on the "signal" (*i.e.* the part of the device state that an adversary can predict) and the "noise" (*i.e.* those unrelated parts of a device that are statistically independent). Although at an early stage neither signal nor noise can be precisely described, assessing the *relative impact* of an implementation choice could be useful. Such techniques are thus mainly relevant for designers and this links this deliverable with our work in WP1.

1.1.2 Supporting Non-expert Developers

Although most devices that undergo a rigorous security evaluation are designed by experts today, with the advent of areas such as the Internet of Things (IoT) or autonomous vehicles, we can already witness that security (and safety) critical devices are being designed by non-crypto experts. Especially in markets which (at present) require rapid product releases, "light touch" evaluation schemes (see D1.1) are necessary. But even in such light touch schemes, a considerable demand is placed on those non-crypto experts with regards to designing products that are (at least) not trivially breakable by standard side-channel techniques. Methods which could be integrated in leakage simulation tools and are "easy to use" are therefore of interest.

1.1.3 Producing Supporting Evidence

In the domain where thorough security evaluations are standard, another consideration of REASSURE is not only to make evaluations quicker but also more rigorous (this aligns with WP1). One potential avenue for increasing our confidence in the results of evaluations could be if experimental results are accompanied by some additional reasoning: e.g. a designer could estimate the best leakage model for the "best" (e.g. first order) attack and then reason based on the noise that is expected in a typical real world setting that any (e.g. first order) attack in theory requires at least N measurements. Naturally such statements would be desirable for attacks of any order.

1.2 When Does an Evaluator Gain by Using Shortcut Formulas?

Two main aspects are into consideration: constraint on the number of samples (or traces) and constraint on time.
Shortcut formulas aim at reducing the number of traces needed to estimate attack outcomes (and generally do). Indeed, shortcut formulas only need estimations of expectations or variances of some variable on one set of curves whereas true computation of success rate needs several sets of curves (typically at least 20). An evaluator may have a limited number of traces to analyze, e.g. this occurs when the targeted command has a limited number of executions. Thus, the amount of acquired traces may be largely insufficient to empirically compute a success rate but sufficient to use shortcut formulas.
Regarding the time execution, some shortcut formulas (as in [26, Sect.6.4.1] or [12]) are always faster than the computation of true attacks. But some others require an estimation of high dimensional multivariate Gaussian cumulative distribution (e.g. [13] and [22]) and are not always faster than true computations, especially for high SNR and low order attacks. Assuming that an estimated success rate based on 100 evaluations of a 255-dimensional Gaussian Cumulative Density Function (CDF) takes 800 seconds, and that a Correlation Power Analysis (CPA) attack on 10000 traces takes 0.35 seconds, Figure 1 and Figure 2 show time computations comparisons. Figure 1 shows that shortcut formulas are faster only if the Signal-to-Noise Ratio (SNR) is lower than 0.00004. If not, computing true attacks is more efficient. In the case of second order analysis, the time complexity of true attacks increases while it remains constant for shortcut formulas. Shortcut formulas are likely to be more efficient than true attacks (when SNR is lower than 0.02).

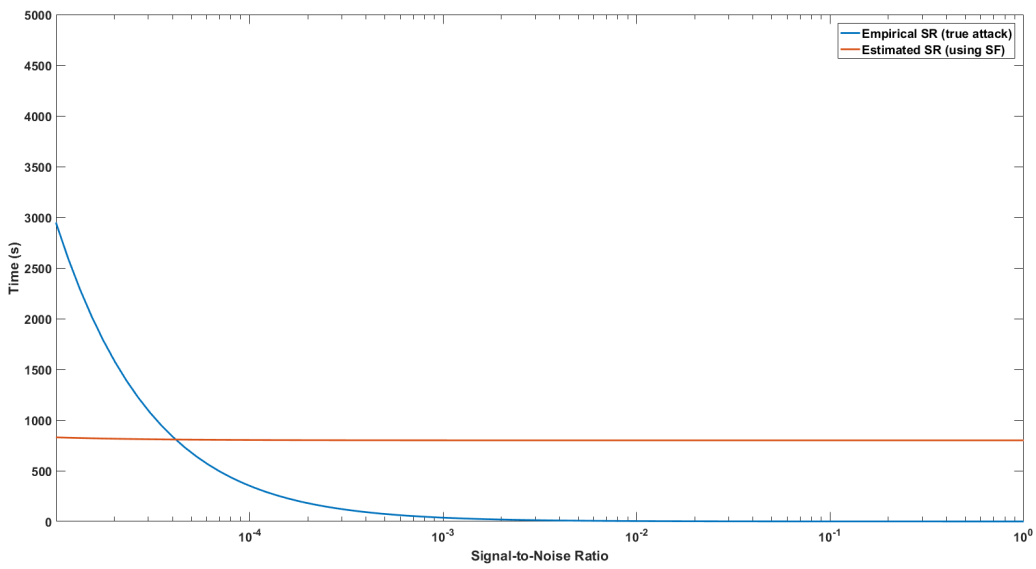

Figure 1: Computation time of 100 true attacks vs one estimated attack outcome in function Signal-to-Noise Ratio (1O-SCA)

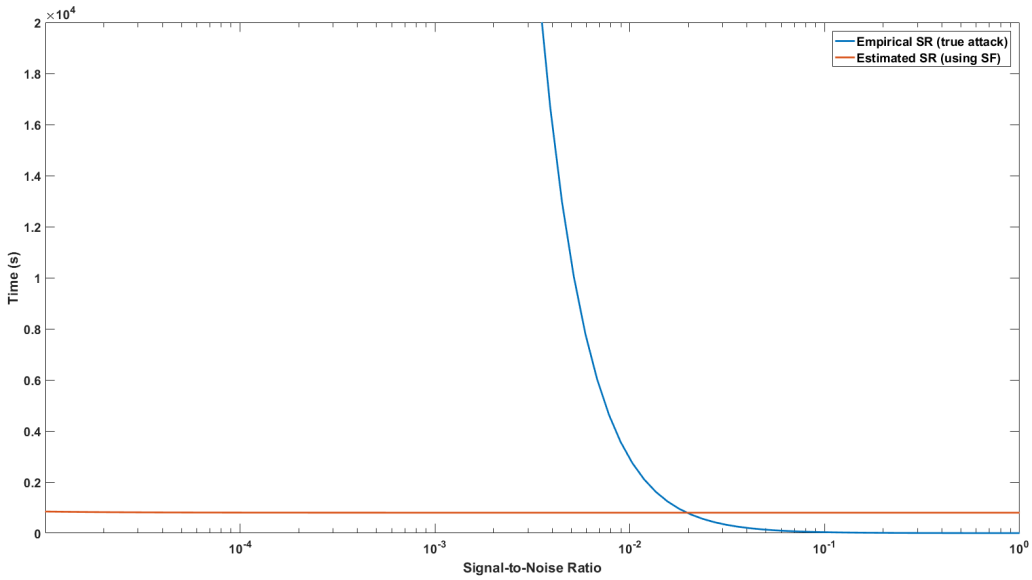

Figure 2: Computation time of 100 true attacks vs one estimated attack outcome in function Signal-to-Noise Ratio (2O-SCA)

1.3 On the Choice of Shortcut Formulas

Among the different shortcut formulas in the literature, one can wonder which one is the most appropriate for a certain evaluation. This section gives some indications to help the non-expert evaluator to choose between the correlation-based rule of thumb [26], the Mutual Information (MI)-based method [12], the confusion matrix-based method [13] and the CHES14 method [22].
The following table summarizes the differences between these four approaches.

	[26]	[12]	[13]	[22]
Time complexity	Fast	Fast	some minutes	some minutes
Accuracy	+	+	++	+++
Assumption	-	high noise level	true leakage function = leakage model	-
Application	1O-SCA only	dO-SCA	1O-SCA only	dO-SCA

First, [26] and [12] are faster than the two others, as explained in 1.2. Second, the difference of accuracy between the four approaches depends on many parameters and cannot be stated in an absolute manner. The user should consider the ranking given in the table as a rough indication based on the following intuitions:

- [26] does not consider interactions between the candidates scores and wrongly consider that an uncorrect guess converge to zero.
- [12] estimates a lower bound of the Success Rate (SR) in the worst case (*i.e.* does not estimate success rate of an attack in particular).
- [13] does not take into account difference between true leakage function and the used leakage model.
- [22] is the most accurate as it uses a profiled model of the target to estimate attacks outcome.

Note that the confusion method ([13]) is less accurate if the true leakage function is different from the leakage model used by the attacker.
Note also that [26] and [13] can be extended to multivariate analysis by estimating SNR from some first order estimations as in Equation 6.

1.4 Approach and Organisation of this Document

The first part is dedicated to the introduction of some shortcut formulas. We organise this section in line with the typical working principle of a (differential) side-channel attack, which follows a divide and conquer principle. The first step is thus the estimation of a distinguisher statistic for each subkey. The second step combines the subkey information to obtain the full key via enumeration.

In this setting we can encounter the following cases:

- The direct estimation of the distinguisher is feasible and enumeration is possible: This is typically the case in an unprotected implementation for which shortcuts are not necessary, but can be useful to gain intuition about the parameters influencing security (supporting early decision making). We deal with this case in Section 2.
- The direct estimation of the statistic is feasible but enumeration is hard: This is typically the case in a state-of-the art protected implementation. In this case, we need a computational shortcut: rank estimation. We deal with this specific task in Section 3.
- The direct estimation of the statistic is impossible (requires too much data) and enumeration is hard: This is typically the case of a masked implementation using a large number of shares. In this case, we additionally need a statistical shortcut: for example the metric-based estimations of the success rate or other security metrics. We deal with this case in Section 4.

The second part shows the relevance of shortcut formulas through some empirical results.
In appendix, a Matlab implementation of [13] can be found.

2 Basic Trace Complexity and Success Rate Estimates

Most security evaluations require performing Differential Power Analysis (DPA) style attacks (or Differential ElectroMagnetic Analysis (DEMA)). Such attacks derive scores for subkey candidates (they are thus divide-and-conquer attacks) by analysing a (small) number of leakage points across a (large) set of leakage traces. Typically this can be done without knowledge of the precise leakage models exhibited by a device and without knowledge of the corresponding time points. The fact that DPA strategies tend to analyse leakage points across a set of traces has led to them sometimes being termed as "vertical" attacks. In contrast "horizontal" attack strategies exploit leakage features within a leakage trace. This typically requires knowledge about what time points to consider, and often also their corresponding leakage models. Thus modern attack strategies can be understood as being part of a continuum of attacks: on one end of the spectrum there are attacks which require very little knowledge about a device and an implementation, and on the other end there are attack which require full knowledge about the device and the implementation.

Within most evaluation schemes there exists the notion of the strength of an attacker. The strength tends to be defined related to the capabilities of an adversary and their knowledge (*i.e.* their attack potential). DPA style attacks require modest attacker expertise and knowledge only, hence they will be the "base line" attack for any evaluation, which is why it is interesting to have some simple means to predict DPA outcomes early in any design. This deliverable thus focuses on simple predictive formulas for DPA style attacks.

2.1 Background and Notation

Most evaluations are based on a "standard DPA attack" scenario as defined in [24]. We briefly explain the underlying idea as well as introduce the necessary terminology here. We assume that the power consumption P of a cryptographic device depends on some internal value (or state) $F_{k^*}(X)$ which we call the *target*: a function $F_{k^*} : \mathcal{X} \to \mathcal{Z}$ of some part of the known plaintext—a random variable $X \overset{R}{\in} \mathcal{X}$—which is dependent on some part of the secret key $k^* \in \mathcal{K}$. Consequently, we have that $P = L \circ F_{k^*}(X) + \varepsilon$, where $L : \mathcal{Z} \to \mathbb{R}$ describes the data-dependent component and ε comprises the remaining power consumption which can be modeled as independent random noise (this simplifying assumption is common in the literature—see, again, [24]). The attacker has N power measurements corresponding to encryptions of N known plaintexts $x_i \in \mathcal{X}, i = 1, \ldots, N$ and wishes to recover the secret key k^*. The attacker can accurately compute the internal values as they would be under each key hypothesis $\{F_k(x_i)\}_{i=1}^N,\ k \in \mathcal{K}$ and uses whatever information he possesses about the true leakage function L to construct a prediction model $M : \mathcal{Z} \to \mathcal{M}$.

DPA is motivated by the intuition that the model predictions under the correct key hypothesis should give more information about the true trace measurements than the model predictions under an incorrect key hypothesis. A distinguisher D is some function which can be applied to the measurements and the hypothesis-dependent predictions in order to quantify the correspondence between them. For a given such comparison statistic D, the *estimated* vector from a practical instantiation of the attack is $\hat{\mathbf{D}}_N = \{\hat{D}_N(L \circ F_{k^*}(\mathbf{x}) + \mathbf{e}, M \circ F_k(\mathbf{x}))\}_{k \in \mathcal{K}}$ (where $\mathbf{x} = \{x_i\}_{i=1}^N$ are the known inputs and $\mathbf{e} = \{e_i\}_{i=1}^N$ is the observed noise). Then the attack is *o-th order successful* if $\#\{k \in \mathcal{K} : \hat{\mathbf{D}}_N[k^*] \leq \hat{\mathbf{D}}_N[k]\} \leq o$.

The *success rate (SR)* of a DPA attack is the probability that the correct key is ranked first by the distinguisher (the *o-th order success rate* is the probability it is ranked among the o first candidates); the *guessing entropy* is the expected number of candidates to test before reaching the correct one [44]. These metrics are often associated with the *subkeys* targeted in the 'divide-and-conquer' paradigm rather than with the global key when the partial outcomes are finally combined; we use the terms accordingly, unless explicitly stated.

2.2 Factors Influencing the Success Rate of an Attack

The SR of an attack potentially depends on several factors implicit in the choice of the underlying DPA distinguisher. These are:

- the magnitude of the signal (*i.e.* the exploitable information) vs. the magnitude of the noise (*i.e.* what is independent of the signal),
- the number of leakage observations,
- the properties of the target function that the adversary base their attack on,
- how well the adversary's power model matches the device's leakage model, and
- potentially the properties of the statistical distinguisher.

With regards to the last two points, it was shown in [24] that the three canonical distinguishers (t-test, correlation, and Gaussian template matching) behave asymptotically equivalently when provided with the same power model. Independently, [18] showed that in specific contexts (e.g. non-Gaussian noise, high signal) it is possible to derive optimal distinguishers, but they confirmed the asymptotic results of [24]. Both papers equally confirm, as well as previous work such as [26] that the most important factor for attack success is the quality of the adversary's power model. Fine tuned distinguishers such as [18] may be less useful when used with power models of lesser quality.

Most existing work then that aimed at predicting attack outcomes thus did this in the context of two popular (asymptotically equivalent) distinguishers: a distance of means test, and the correlation coefficient. We will thus focus on formulas for the correlation for the sake of clarity, and specialise notation such that $\hat{\mathbf{D}}_N = \{\hat{R}_N(L \circ F_{k^*}(\mathbf{x}) + \mathbf{e}, M \circ F_k(\mathbf{x}))\}_{k \in \mathcal{K}}$, where $\hat{R}$ is the estimator for the correlation. We use the shorthand $\hat{R}(k^*)$ to refer to the estimated correlation for the correct subkey candidate.

2.3 Existing Work

A number of early works [25, 26] discussed statistical approaches to assess and predict the success of DPA attacks. They focused on using correlation as the distinguisher and assumed that the point at which the attacker's model prediction coincides with the computation of the target function would be 'easily' discernible (this assumption was silently made by all follow-on works too).

A DPA attack will succeed in revealing the correct subkey k^* if the (absolute) difference between the $\hat{R}(k^*)$ and (any) $R(k)$ is positive. In a hypothesis test we can seek to minimise a Type 1 error (*i.e.* we falsely reject the null hypothesis) by drawing enough samples. This fact was utilised in [25, 26] to lower bound the number of required leakage samples to succeed at a specific confidence level (w.r.t the Type 1 error). Because estimated correlation coefficients can be mapped via the Fisher transform to a variable that exhibits a normal distribution, it is possible to analytically express the sample size as a function of the distance between $\hat{R}(k^*)$ and $R(k)$:

$$N = 3 + 8 \cdot \frac{z_{1-\alpha}^2}{\left(\ln \frac{1+\hat{R}(k^*)}{1-\hat{R}(k^*)} - \ln \frac{1+R(k)}{1-R(k)}\right)^2} \tag{1}$$

Where α is a confidence index and $z_{1-\alpha}$ is the evaluation of CDF$(z)^{-1}$ for the Gaussian law. Using as further simplifying assumption that all incorrect subkey candidates behave statistically indistinguishably and approach zero correlation leads to the "DPA rule of thumb":

As a rule of thumb, the number of traces n that are needed to mount a successful DPA attack can be calculated as follows:

$$n = 3 + 8 \frac{z_{1-\alpha}^2}{\ln^2 \frac{1+\hat{R}(k^*)}{1-\hat{R}(k^*)}}, [26, Sect.6.4.1]. \tag{2}$$

The fact that the rule of thumb does not take the behaviour of the other key candidates into account implies its limited predictive power for the SR (as observed in [42] and partially addressed in [46]), and this was the motivation for [41]) to develop a technique that factored in the behaviour of all subkey candidates and could be used to derive an o-th order SR. Finally, the work by [14] sets the foundations to better understand the impact of the target function on the distinguishing power of an attack. In the context of single-bit attacks their results

are explicit. The multi-bit case under the assumption of a Hamming weight leakage model was resolved in [47] and a complete treatment for linear leakage models was given finally in [13].

Their (more restrictive) device model is that leakage is given as $L = \epsilon F_{k^*}(X) + c + r$, where ϵ and c are unknown constants and r is independent noise from a Gaussian distribution with zero mean and σ^2 variance. In the context of correlation, $F(X)$ can be the Hamming weight or distance (to some known value), which is scaled by a single value of ϵ (thus the earlier mention of restriction to a linear power model). Under these assumptions, [13] express the success rate as $SR = \Phi_\Sigma(\sqrt{N}\mu)$:

$$\begin{aligned} \mu &= \tfrac{1}{2}\left(\tfrac{\epsilon}{\sigma}\right)^2 \\ \Sigma &= \left(\tfrac{\epsilon}{\sigma}\right)^2 \mathbf{K} + \tfrac{1}{4}\left(\tfrac{\epsilon}{\sigma}\right)^4 (\mathbf{K}^* - \kappa\kappa^T). \end{aligned}$$

In these equations $\mathbf{K}$ and $\mathbf{K}^*$ are the so-called confusion matrices, and κ represent confusion vectors. These quantities are only dependent on the target function and can thus be calculated prior to an attack. It is evident that by applying Φ^{-1} we can express N as a function SR and thus have a similar expression as the DPA rule of thumb.

These results all relate to standard DPA attacks and thus do not apply in any straightforward way to scenarios in which pre-processing might violate the Gaussian noise assumption. This might happen in the case of so called higher-order DPA attacks, which are relevant for masked implementations, see Section 4.

2.4 Assessing the Impact of Countermeasures

When considering correlation based distinguishers, it is easy to express the impact of some countermeasures on the correlation coefficient of the correct key [26]. For instance if traces are misaligned, then this misalignment can be expressed as a probability that the time points associated with the target value are in the "right place" p. The adjusted correlation coefficient (for an attack on such traces) is then scaled by this probability: $p \cdot \rho_{k^*}$, assuming variances on trace points are roughly equal (if these variances are considerably different, this fact can be taken into account as well). However, the main variable that a designer can control is the probability for displacement.

If countermeasures directly change the signal-to-noise ratio, then this can equally by expressed in terms of resulting correlation ρ':

$$\rho'(k^*) = \frac{\rho(k^*)}{\sqrt{1 + \frac{1}{SNR}}}.$$

For small correlation coefficients it holds that the correlation can be approximated by the signal-to-noise ratio (ignoring a constant c) [26]:

$$\frac{c}{\rho(k^*)^2} \sim \frac{1}{SNR}. \tag{3}$$

These relationships are well known in theory and it is of interest to challenge their practical use in an internal evaluation or design process.

2.5 Considerations for Practical Use

There has been a considerable evolution in the reasoning about attack success: from [25], where attack success was related to a notion of Type 1 error in hypothesis testing, over [41], where the properties of wrong key candidates got included, to [13], which features explicit formulae that express both the target function properties as well as the signal and noise. The increase in precision and sophistication comes of course at a cost: whereas the approach by Mangard just requires a successful attack to "estimate" the relevant parameters for a simple formula, the approach by Fei requires profiling of the device and is based on a leakage model that is a linear function of (some combination) of (some) bit(s).

In the practical context of security evaluations, we can expect that many devices under evaluation feature hardware implementations. One should note though that such hardware implementations only get evaluated directly in IC evaluations. Otherwise, in the case of composite evaluations, embedded software and hardware are evaluated jointly. Typically in such a scenario, the embedded software severely restricts direct access to the hardware and thus makes profiling extremely challenging. If profiling is possible it is highly likely that such devices might not exhibit a linear leakage model, or even any leakage model that is linear in individual bits (there might be differently weighted Hamming distance leakages between register writings or due to the used bus lines).

In "in-house" evaluations a final version of the device is also only available after tape out. Because internal teams tend to work independently from design teams, the signal and noise characteristics have to be estimated empirically with limited information/resources. However this information does not get passed on to the external evaluator because this would make it easier for them and thus artificially inflate the "attack potential" (which is not in the interest of the product manufacturer).

During an evaluation within a set scheme the evaluator, or maybe even the product designer might want to "prove" that an attack with less than N traces is statistically highly unlikely to succeed. Whilst proving such a statement in a strict mathematical sense is not possible, it would certainly be feasible to statistically argue such a statement, in which case a reasonably exact formula (for the SR or the number of traces N) might be desireable.

Thus it seems pertinent to check the applicability of [13] in the context of challenging devices and as an alternative to consider wether it is possible to improve the predictive power of the rule of thumb.

3 Key Rank Estimation

DPA attacks utilise a divide-and-conquer strategy: they target small portions of a key independently. Until 2012 the academic community ignored the fact that even imperfect attack outcomes (*i.e.* some subkeys might not be uniquely revealed) still give substantial information, which, together with some enumeration effort would enable an adversary to determine the entire cipher key (assuming access to a pair of plaintext and corresponding ciphertext). This changed with the work of Veyrat-Charvillon et al. [48], which introduced a first algorithm to *enumerate* and test the most likely candidate keys (from the most to the least likely using known plaintext and ciphertext pairs) to determine whether a candidate is the correct key.

Informally, the number of candidate keys an adversary must enumerate (and test) after an imperfect side-channel attack before arriving at the correct key sk is termed the *rank* of the key. Recent efforts [4, 6, 15, 33, 51, 49] considered determining the rank of the correct (known) key after the side-channel phase of an attack. The computation of the key rank is thus a natural task in the context of any evaluation.

The rank of a key is a function of the ranks of all subkeys: thus some notion is required that enables the combination of subkey scores. In the case of profiled attacks this can be done via a straightforward argument: template matching directly operates on the normal Probability Density Function (PDF) and thus produces likelihood scores, which via some simple normalisation, can be viewed as conditional probabilities $\Pr(k|L)$. Similarly, linear regression based scores have been argued as "close enough" to probabilities because one can show the direct derivation of typical linear regression distinguishers from a normal pdf. This has been used as motivation to consider them "suitable" as probabilities. In contrast other distinguishers, and notably popular ones such as the distance-of-means or correlation distinguishers, seem less suited to be interpreted as "probabilities".

3.1 Assigning Likelihoods

One can try as work-around to use the Fisher transformation to map estimated correlations to "probabilities". This evidently shows a problematic understanding of what constitutes a "probability": estimated correlations already have a (non-parametric) distribution (and therefore can be associated a probability) thus using the Fisher transform only makes them "easier" to handle but the transform doesn't make them "better" as probabilities. Furthermore for small correlation values the Fisher transform is no more than a linear mapping, and this means that it does no more than preserving the ordering and relative magnitudes of scores. Clearly then any linear mapping has the same properties and thus is equally good for the purposes of key ranking.

The attempt of [48] to estimate "genuine" probabilities on the subkey hypotheses in the non-profiled setting, by using the recovered models derived from linear regression based attacks, is expensive and features another problem: incorrect key hypotheses recover of course invalid models and thus no meaningful statements can be made by using them. Whilst their mapping also preserves the ranking of the keys as they appear in the distinguishing vector produced by a non-profiled linear regression-based DPA, because of the nature of the formula used, it dramatically exaggerates the apparent distance between the high- and low-ranked key candidates. If the implied key is the right one it reinforces this "correct" result. But if it is not the right one it reinforces the misleading result.

Any distinguisher is a function that assigns some "credibility" to a key guess (which is a parameter in the attacker's model) given some leakage observations. The interpretation of an attack outcomes is that larger distinguisher values give a key candidate more "credibility": we consider a key candidate with a higher score to be more likely than a key candidate with a lower score. Thus if distinguisher scores need to be related to probabilities, a simple and conservative mapping is most likely to preserve our interpretation of those scores, e.g. via a simple linear mapping.

3.2 Existing Key Rank Algorithms

A host of advanced key rank *estimation* algorithms returning either an interval containing the actual rank or a point estimate of the rank have been published in the literature. When comparing such algorithms, both the

efficiency and the accuracy are relevant. Accuracy is measured in bits, where b bits of accuracy means that if an algorithm says the key has rank 2^x, the actual rank is in the range $2^{x \pm b}$.

Veyrat-Charvillon et al. [49] developed the first non-trivial key rank algorithm. They consider the distinguishing scores as being in a multi-dimensional space, where each dimension represents an individual (sorted) distinguishing vector. This space can naturally be divided into two parts; one part contains those keys with rank higher than the target key and the other part contains those with a rank lower. Using the property that the 'frontier' between these two halves is convex, they give an algorithm that can estimate the rank of the key within 10 bits (of accuracy) by repeatedly pruning the space.

A big step forward towards efficient key ranking was made in Glowacz et al. [15]. They construct an efficient rank algorithm based on the convolution of histograms. They utilise the property that if H_1 is a histogram of $\mathcal{S}_1$ and H_2 is a histogram of $\mathcal{S}_2$ then the convolution of H_1 and H_2 is a suitable approximation of $\mathcal{S}_1 + \mathcal{S}_2 = \{x_1 + x_2 | x_1 \in \mathcal{S}_1, x_2 \in \mathcal{S}_2\}$. By discretising the distinguishing vectors and utilising histograms they are able to estimate the rank of the key with an arbitrary accuracy.

Simultaneously, Bernstein et al. [4] proposed two key rank algorithms. The first adds a post-processing phase to the algorithm by Veyrat-Charvillon et al. [49], which tightens the accuracy to 5 bits. The second algorithm uses techniques similar to counting all y-smooth numbers less than x, which can be thought of as the convolution trick used in [15].

Duc et al. [12] propose a similar solution to that of Glowacz et al. [15]. They repeatedly "merge" each set of data (similar to the histogram convolution) and then down-sample the resulting data (this could be regarded as the binning step in creating histograms). The additional down-sampling indicates that the accuracy of that algorithm will be worse than that of Glowacz et al. 's algorithm. They are very clear that this approach will only be tight for implementations of masking with a large number of shares, which is confirmed in [31] by comparison to rank predictions for first and second order masking (*i.e.* a setting with not enough shares).

Martin et al. [33] give a key rank algorithm based on an efficient path counting algorithm in a graph. After mapping the distinguishing scores to integer weights (such that larger distinguishing scores give smaller integers), they are able to efficiently count the number of keys with a weight less than the target key which directly corresponds to the rank of the key. Varying the size of the resulting integers allows them to make a trade-off between accuracy and runtime.

Two of the recent key rank algorithms [4, 15] have been known to be similar: both use a convolution approach to combine subkey information. In contrast, [33] appeared to be based on a different graph-based technique. However, it was shown in [30] that it is possible to express the histogram method as a version of the path counting approach (as given in [32]), and thus show mathematical equivalence between the two ranking methods. Their proof is based on the fact that the convolution based approach assumes equally spaced bins, and this implies an equivalence between the "precision" parameter of the path counting approach and the "number of bins" parameter of the convolution based approach. Using this they rewrite the equations that underly the convolution based approach, such that they are equivalent to the equations of the path counting approach. By showing mathematical correspondence between "precision" and "number of bins" they also settle any open questions about the accuracy of those methods (both methods are equally accurate).

Whilst both methods arrive at the same result mathematically (assuming use of the same discretisation parameter, and the same "shift to zero" technique), there is a clear difference in how they are expressed algorithmically, which implies that their practical performance will be different. To achieve a like-for-like comparison, [30] run both on the discretisation parameter for which their underlying mathematical representations are equivalent. Their comparison shows that up to 12 bits of precision (which is equivalent to 2^{12} bins) the convolution based method is faster than path-counting. From 12 bits of precision onwards path-counting wins.

Precision is crucial for the ability to parallelise large enumeration efforts across many cores. Thus for small to medium size search efforts, convolution seems the better choice, whilst for large scale search efforts a path-counting implementation appears to be preferable. However, an in-depth study of both algorithms on a high-performance computing platform is necessary before any firm conclusions can be drawn.

All listed algorithms, and also more heuristic approaches such as [51] have the limitation that their complexity

grows with the number of subkeys that need to be considered. In addition all existing implementations are optimised for typical block cipher parameters, *i.e.* 16-32 subkey bytes. Whilst it is entirely feasible with existing implementations to tackle up to 64 subkey bytes reasonably efficiently, going much further puts current implementations at their limit. This might not be a problem for symmetric primitives, however key ranking is also relevant for public-key primitives, e.g. RSA keys, ECC keys, and of course including applications from post-quantum cryptosystems. Consequently, there is a need to either improve existing ranking algorithms, or to consider other options of predicting key ranks. The first approach of this kind was very recently published [7], where they provide bounds for a Guessing Entropy estimate which is directly derived from the score vectors. They show that their bounds are reasonably tight, when compared to [4, 15]. However, this comparison depends on the number of bins and the impact of this choice has not been further investigated.

3.3 Interpreting Key Rank Outcomes

The key rank $\mathrm{kr}_{\mathbf{sk}}(\mathbf{D})$ of a single side-channel attack is not particularly interesting on its own. To say something meaningful, we need to regard the key rank as a random variable that results from an attack which is based on a randomly selected key, random plaintexts and some random noise. We will denote this random variable KR and a single realisation from an attack as kr.

As specified in [49], one way to exploit key rank outcomes is to compute a security graph, expressing the complete key recovery success rate of an attack according to the number of used traces and the computing power of the attacker. While this picture describes the overall distribution of KR, some shortcut interpretations can also be used.

One can link the key rank to the Guessing Entropy [34] by observing that the key rank is the number of guesses an optimal adversary would take to guess the secret key; [45] first made this connection. The guessing entropy captures the expected number of guesses (with an optimum strategy) to correctly guess the value of a random variable (in our scenario the secret key). The key guessing entropy is defined as:

$$\mathrm{GE} = \mathbb{E}(\mathrm{KR}) \ .$$

A key observation is that the guessing entropy is the *expected value* of the distribution of the key rank. Rivain found that the distribution of a *additive distinguishing vector* tends to a multivariate Gaussian [41], but the general distribution of the subkey rank and also the full key rank have not been thoroughly explored.

The pertinent question in the context of an evaluation is how to best consider the relative strength of two adversaries that have different sized key enumeration budgets. In DPA style attack contexts we assume access to at least one pair of plaintext and ciphertext, and therefore the cost of checking a key is almost zero—a single call to an encryption or decryption. Thus, very much alike as in classical cryptanalysis, it is perhaps more useful to compare enumeration budgets in terms of *orders of magnitude*, *i.e.* consider the logarithm of (a function of) the key rank outcomes. In [31] the authors propose as measure the *ranking* entropy:

$$\mathbb{RE} = \mathbb{E}(\log(\mathrm{KR})) \ .$$

The ranking entropy is hence defined as the expectation of the logarithm of the rank. It is important to be aware of that taking logarithms and expectation do not commute, so in general the ranking entropy will not equal the log of the guessing entropy.

Within [31] several experiments to assess representative key rank distributions are performed. With permission from the authors, we include Figure 3 here. It shows the distribution of key ranks across a range of average rank values from an attack on AES-128. The interesting fact is that whilst rank distributions of "mid" ranks are approximately normal, distributions associated with ranks either closer to zero or to the maximum rank are no longer symmetric and appear to be truncated. This has an implication for approaches such as [7] which rely on approximations that work well when the cumulative distribution "behaves well": for mid range key ranks we would expect that approximations work well, but for ranks in either tail of the distribution it could be that the predictive power of such approaches will be insufficient. Another important observation is that these distributions are very wide: this means that practical key ranks will be widely distributed around the expected

value, and thus the chance for "lucky" adversaries is non-negligeable. A better illustration of this fact is shown in Figure 4 (again included with permission of the authors of [31]). A "lucky" adversary is an adversary who gets a "good" sample set of leakage observations by chance and thus extracts the key with less enumeration effort (or even none).

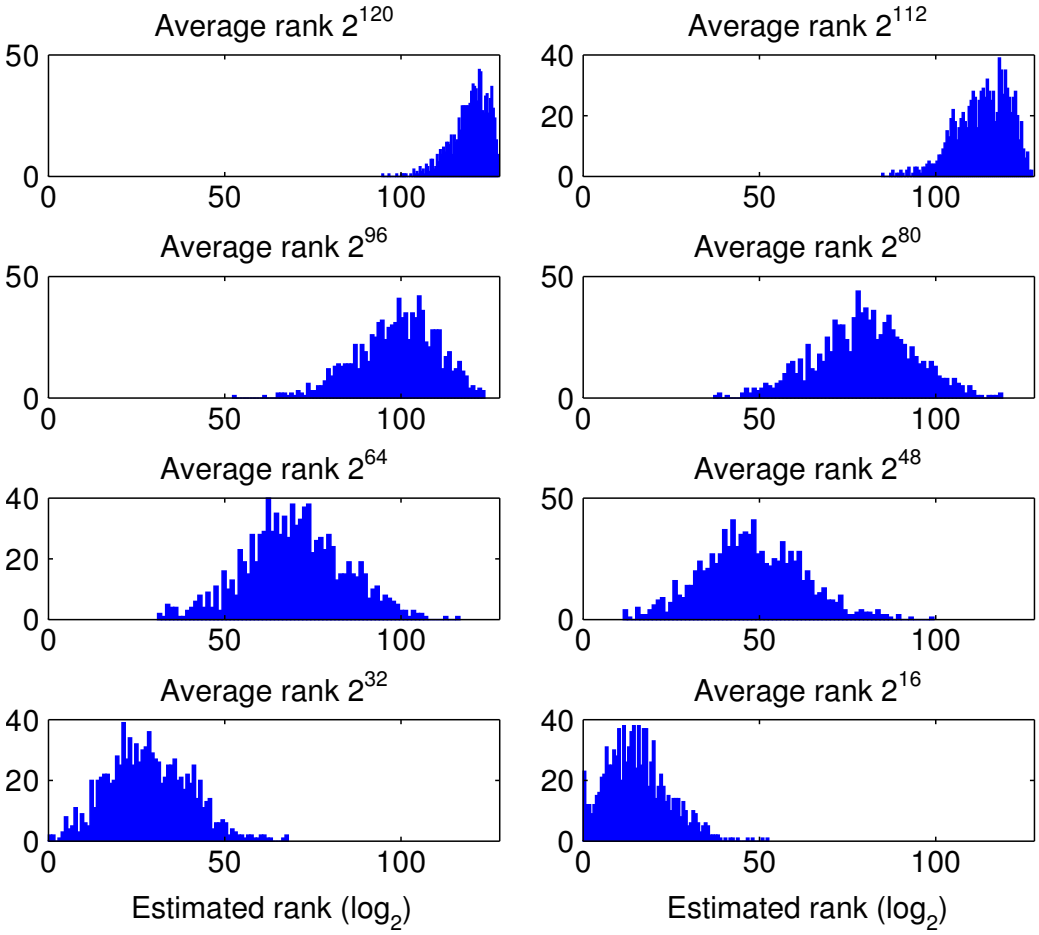

Figure 3: Histograms for attacks with a (geometric) mean rank close to one of several values. Here the leakage is simulated Hamming-weight with Gaussian noise at an SNR of 2^{-7}, with the attacker using CPA as a distinguisher.

Finally, [31] also observe that the variance of the rank distribution seemingly depends on the DPA distinguisher (correlation exhibiting a wider variation). Thus for popular correlation based attacks, again there is a non-negligeable chance of "lucky" adversaries.

3.4 Data Complexity Shortcuts for Key Rank Estimation

Section 2 introduced the basics of a standard DPA attack and showed how a simple-to-estimate metric (such as the correlation coefficient) translates into an estimation of the success rate. This gives a data complexity shortcut formula in the case of standard DPA against a single subkey. However, estimating the individual success rates of the independent subkeys does not give the overall success rate on the full key. In that matter, the aforementioned key rank estimation algorithms aim at giving this global success rate in a sampling-based manner. That is, from the results of several independant attacks on all subkeys, the overall remaining security is calculated by (e.g.) computing the ranking entropy. As a result, they cannot be combined with the method of Section 2 to give information on the full subkey recovery.

Two algorithms have been developed to tackle this issue [38]. As opposed to all the previously described rank estimation algorithms, they are designed to directly work with a metric-based input (such as the success rate) instead of sampling-based one (such as the result of a single actual attack). As a result, when combined with

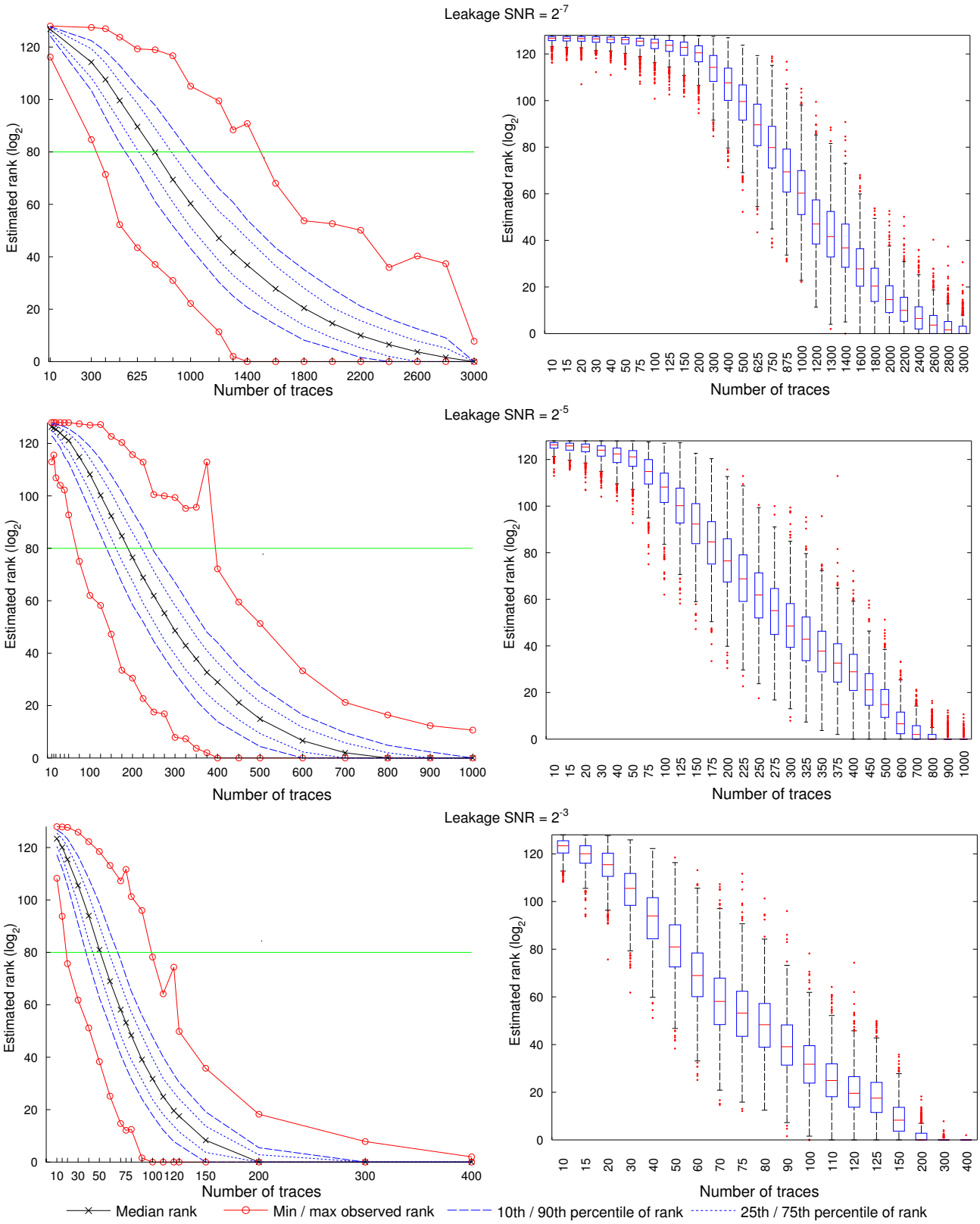

Figure 4: (Left) Estimated ranks after 1,000 DPA attacks at SNRs $2^{-7}, 2^{-5}$ and 2^{-3}, using Hamming-weight, targeting simulated leakage on the AES SubBytes operation. (Right) Equivalent box-plots for using the same data as on the left. The central line in each box is the median, the box defines the inter-quartile range, the whiskers cover all samples not considered to be outlier values, and outliers are plotted individually.

the computation of the subkeys' success rate of Section 2, they allow estimating the guessing entropy of the full key. As a drawback, the resulting guessing entropy of these two algorithms does not correspond to the one given by an optimal sampling-based enumeration. The first algorithm, that we denote by Metric-based Lower Bound

(MLB), corresponds to a suboptimal enumeration, thus resulting in an over estimation of the actual security level. The second one, that we denote by Metric-based Upper Bound (MUB), corresponds to an over-optimal (unfeasible) enumeration, thus resulting in an under estimation of the actual security level. Fortunately, bounds on the actual security can be obtained by computing both the MLB and the MUB. That is, the first one (resp. the second one) provides a higher (resp. lower) bound on the security level. Figure 5 shows the result of a both MLB and ULB compared to an optimal sampling based rank estimation. The security graph [49] computed from the MLB (resp. MUB) are depicted on the top left (resp. top right) part of the picture, where the security level for a given number of attack traces is given by the red line. On the other hand, the bottom part of the figure shows the security graph computed from the sampling over many attacks and using one of the rank estimation algorithm described in the previous subsection. As we can see, the red line given by the MLB (resp. MUB) is above (resp. below) the one given by the sampling method. By computing both, the evaluator is thus provided with bounds on the actual security level.

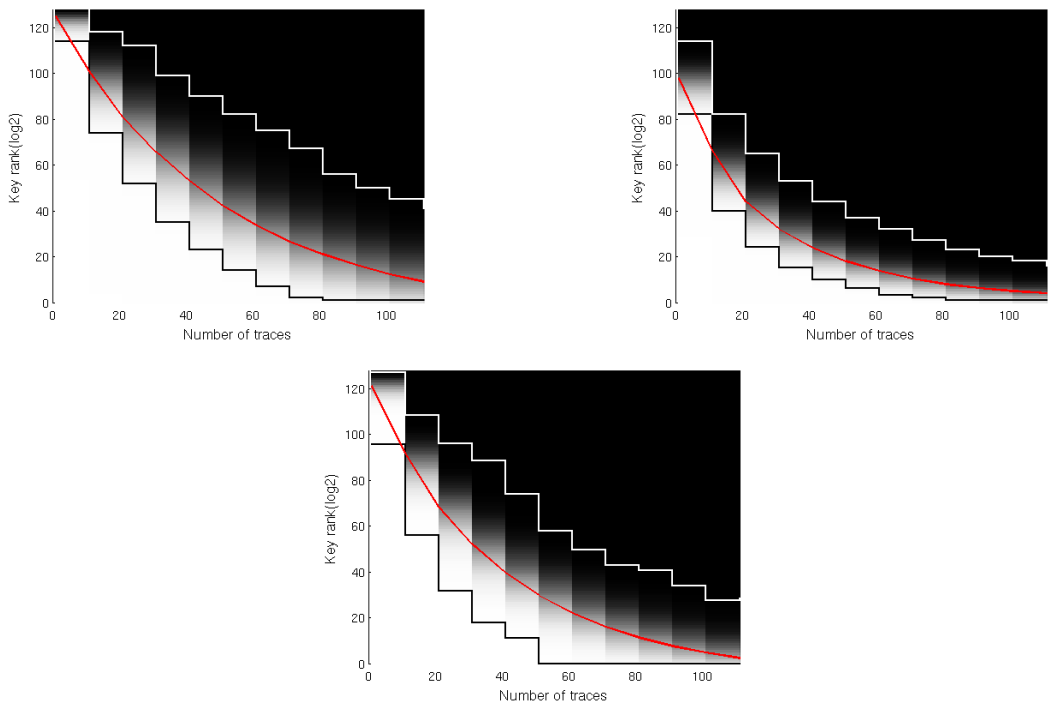

Figure 5: Security graphs. Top Left: MLB method. Top Right: MUB method. Bottom: sampling-based optimal rank estimation.

3.5 Considerations for Practical Use

Approaches such as [7] are interesting for evaluating very long keys. For typical DPA style attacks on state of the art symmetric encryption schemes though, existing tight rank estimation algorithms seem to be a more conservative choice especially for very high or low ranks where the rank distribution is not "well" behaved. For average ranks the tightness of the estimation might not be a huge concern because the large variation of the rank itself.

For practical usage another consideration (besides tightness and speed) is important: because the key rank is a random variable, a single sample of it is entirely meaningless. An evaluator thus has to consider how many samples of the key rank are necessary to be able to make any meaningful statement, and how to deal with (*i.e.* report or not) the percentage of "lucky" adversaries. The authors of [31] suggest to use percentiles and demonstrate how to conduct a meaningful evaluation of the security margin of a "challenging" target (a hardware AES implementation on a Beaglebone, for which just under one million leakage observations are available in total). As an alternative, approaches directly based on estimating a security metric, such as sketched in the previous section, are potentially "tight enough" as well.

4 Masking Attack Complexity

When moving to the important case of (high-order) masked implementation, the evaluation problem naturally becomes more challenging. First, simple metrics for the estimation of the data complexity, based on the SNR or the correlation coefficient (as described in Section 2) are not valid anymore. Indeed, the main effect of a masking scheme is to hide the sensitive information in higher-order statistical moments of the leakage distribution which are not (directly) captured by such metrics. Taking the simple example of a d-share additive encoding $x = x_1 \oplus x_1 \oplus \ldots \oplus x_d$ (e.g. such as proposed in [19]), with corresponding leakages $\bar{l} = [l_1 l_2 \ldots l_d]$ and $l_i = \mathsf{L}(x_i)$, we have that only a mutual information metric such as $\mathrm{MI}(X; \bar{L})$ (or similar metrics capturing the full leakage distribution) can predict the attack complexity [39, 12]. But the direct estimation of this metric becomes exponentially hard as the number of shares in the masking scheme increases (just as the expected security level). In this section, we discuss various approaches that can be used to approximate or bound the worst-case security level of an implementation in this case, together with their limitations. The main motivation for these approaches is that evaluation laboratories are in general very constrained in resources. So it is desirable to be able to state security results for complexities that go significantly beyond the time and data that can be devoted to evaluation in practice.

4.1 Evaluating the MI Metric

One first natural shortcut for avoiding the direct estimation of $\mathrm{MI}(X; \bar{L})$ is to consider the main result of masking security proofs, which is that this metric can roughly be bounded by $\mathrm{MI}(X_i; L_i)^d$. The latter is extensively discussed in [12]. In brief, such an approximation is leading to correct results if the two main assumptions of masking are satisfied (namely that the leakages of the shares are sufficiently independent and noisy - see Section 4.3 for a discussion). Note that more specific approaches can be used to obtained similar bounds based on the SNR of the shares [22], or their correlations [10]. Due to the equivalences in [27], they lead to similar intuitions, though the use of an information theoretic metric allows a cleaner connection with formal security proofs.

Quite naturally, the estimation of the success rate via such a shortcut through the estimation of the mutual information implies that one cannot directly perform key enumeration or rank estimation anymore. By contrast, it can directly be plugged into the metric-based bounds of Section 3.4, leading to extremely fast security assessments (see Section 4.3 of [12]). The latter admittedly leads to very rough (conservative) estimates of the security level of an adversary exploiting one d-tuple of leakage samples. Yet, it gains relevance when security levels increase significantly beyond the concrete capabilities of evaluation laboratories (typically, such an approach becomes relevant if the goal is to claim security with up to $> 2^{30}$ measurements).

4.2 Considering Horizontal Attacks

Given that the MI (or a similar) metric can be bounded thanks to the previous shortcut, it then remains that worst-case side-channel attacks should not only exploit one d-tuple of samples but all the ones available in the implementation.

First sticking with a divide-and-conquer approach (*i.e.* considering attacks that only exploit the leakages corresponding to intermediate computations that only depend on an enumerable part of the key), it already happens that such attacks become increasingly powerful when the number of shares of a masking scheme increases, because the quadratic performance overheads it implies leads to the presence of many d-tuples that can be exploited in this way. Yet, as discussed in [3, 17], the joint and optimal exploitation of all these d-tuples can be computationally intensive (especially when considering the leakage of secure multiplication algorithms). The concrete solution proposed in these papers to circumvent this (computational) difficulty is to exploit decoding algorithms such as the Belief Propagation (BP) or related ones [23]. More theoretically, it is argued in [17] that under some additional (independence) assumptions, it is possible to bound the total amount of information exploitable by divide-and-conquer attacks by simply identifying the leaking operations and summing their leakages (which, for multiplication algorithms, leverages the bound proposed in [39]).

Second, it is worth noting that the previous bounds remain limited to divide-and-conquer attacks, while optimal attacks (such as analyzed in [11]) exploit all intermediate computations in a leaking implementation. The concrete solution to exploit such leakages is to rely on Soft Analytical Side-Channel Attacks (SASCA) [50, 16], which also exploit the BP algorithm.[1] As for the use of the BP algorithm to exploit the leakage of the predictable d-tuples in a masked implementation, the implementation of SASCA is however quite involved. It is therefore an interesting open question to find out whether shortcut approaches can be generalized in order to efficiently bound the information leakage in such a worst-case context.

4.3 Considerations for Practical Use

As aforementioned, the main limitation of the previous shortcut approaches is that they heavily rely on assumptions that may not always be fulfilled by actual implementations. In this respect, the noise condition and the independence assumptions lead to quite different intuitions. On the one hand, the noise condition may be difficult to guarantee in practice, but it is usually easy to test/evaluate (thanks to simple tools such as the ones described in Section 2). So it is mostly a concrete problem to design implementations with sufficient noise, in particular when considering small embedded devices where no “algorithmic noise" can help.

The independence condition is not only difficult to guarantee in practice, due to physical defaults such as transitions [9, 1], glitches [28, 29] or couplings [8]: it is also difficult to test. The standard solution for this purpose is indeed to estimate higher-order statistical moments of the leakage distribution [43]. In the current state-of-the-art, there are two main options to mitigate this issue.

On the one hand, one can design implementations with larger number of shares in order to cope with the potential re-combinations due to physical defaults. This is typically what is proposed in [35] for glitches, or in [1] for transitions.

On the other hand, and given the fact that direct test of the independence condition for large number of shares is intensive, one can test “reduced-share" versions of an implementation (analogically with what is done in block cipher cryptanalysis with reduced-round versions), and extrapolate the results (and the observed recombinations) to larger number of shares, while always considering a risk factor capturing the fact that reduced-share versions may not be fully reflective of full versions. This approach was recently applied in [21]. The further investigation and formalization of such ideas is another interesting scope for further research.

[1] Earlier approaches to analytical side-channel attacks were based on an algebraic representation of the target cipher, which in general seems less effective due to their limited tolerance to noise [40, 36, 37].

5 Accuracy of Shortcut Formulas

This section aims at comparing real attack outcomes to the expected attack outcomes given by shortcut formulas. To that end, we will acquire some sets of traces, apply on them several true attacks (in order to compute an empirical SR) and analyse them to estimate theoretical SR through shortcut formulas. We target two different CPU architectures: an 8-bit AVR (with high SNR) and a 32-bit ARM (with lower SNR). In both cases, the embedded assembly code is public and attached to this deliverable.

The goal of this section is to evidence, with some practical cases, to what extent a shortcut formula is able to estimate the true attack outcome, taking into account the noise level.

5.1 Setup Description

The following table gives a description of each target set.

	Set A	Set B
Device	ATmega8515	-
Architecture	8-bit AVR	32-bit ARM Cortex M0
Algorithm	AES128	AES128
# traces	60k	1M

In order to mount the analyses in the following, Points of Interest (PoI) have been selected based on SNR curves. The SNR has been computed using the NICV metric [5]:

$$\text{SNR} = \frac{\text{Var}\left(\text{E}\left[L_i | X_i = x\right]\right)}{\text{Var}(L_i)} \tag{4}$$

where all the realization of X_i are known from the evaluator. The SNR is then computed on each time sample independently. The variable X_i is the targeted intermediate value. Figures 6 and 7 show SNR results on Set A. Figures 8 and 9 show SNR results on Set B.

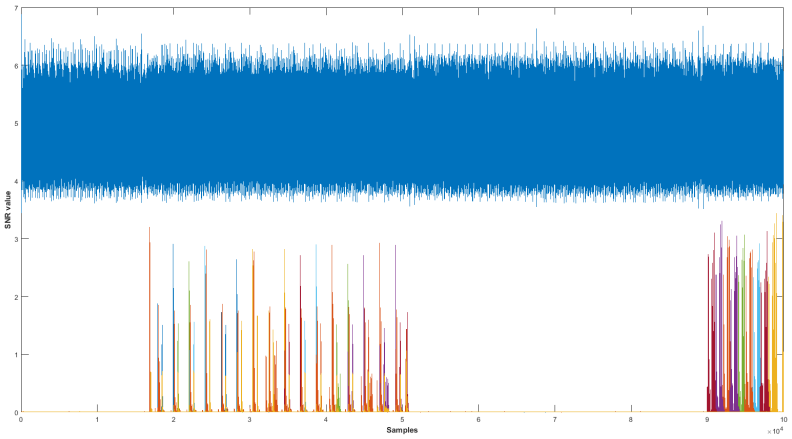

Figure 6: SNR curves when targeting masked S-box output values (with known mask) on Set A

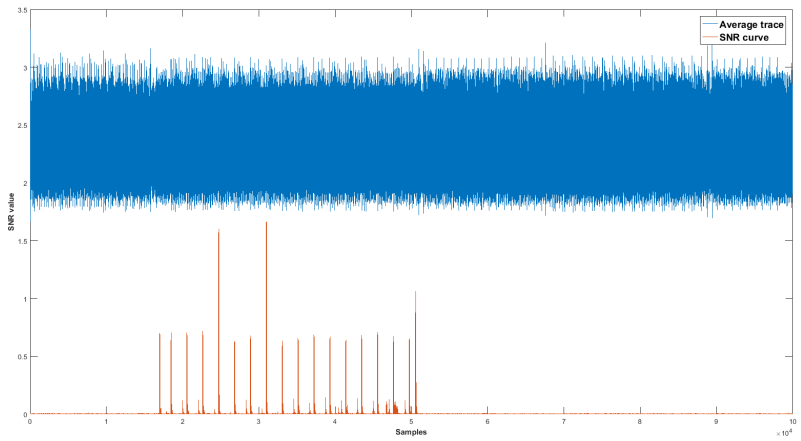

Figure 7: SNR curve when targeting the S-box output mask on Set A

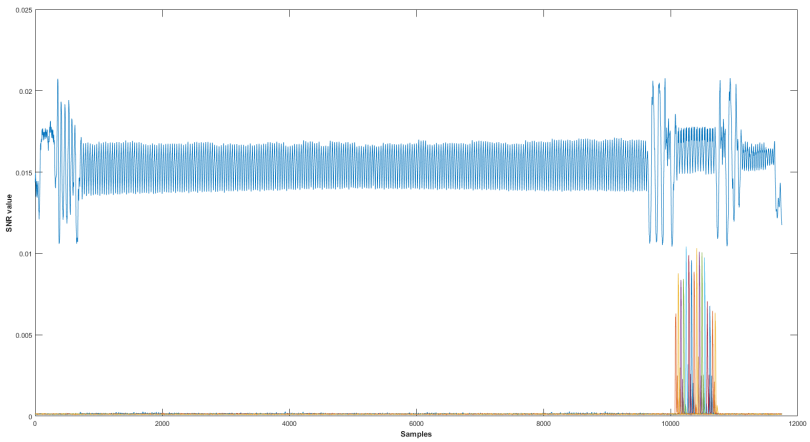

Figure 8: SNR curves when targeting masked S-box output values (with known mask) on Set B

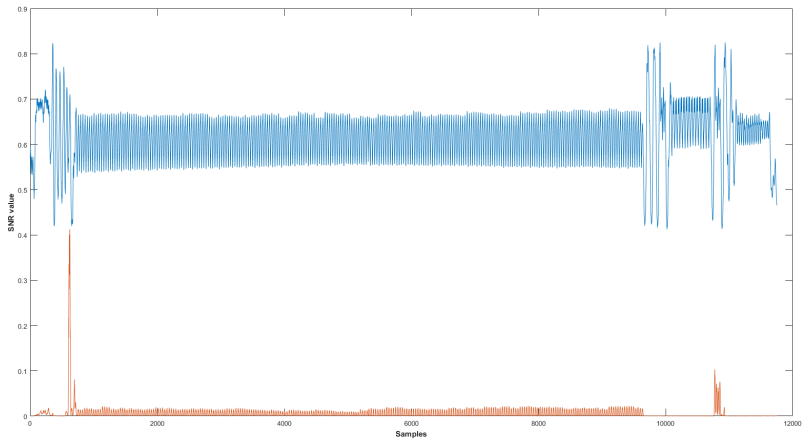

Figure 9: SNR curve when targeting the S-box output mask on Set A

5.2 Case of First-Order Analysis

The following methodology has been applied:

- Mount a 1O-CPA on N traces and estimate the SR thanks to the shortcut formula
- Mount several 1O-CPA on N traces and estimate an empirical SR
- Compare the two SR and evaluate the time saved by using shortcut formula

Three different intermediate values have been targeted: SubBytes input, SubBytes output, and distance between two SubBytes outputs, all with a Hamming weight leakage model. With the assumption that the SNR is equivalent for the three intermediate values, the three attacks should not converge to the same success rate. We expect that the attack on SubBytes input is much harder than the attack on SubBytes output, because of the non-linear property of the SubBytes operation, which ease the discrimination between subkey candidates. On the other hand, we expect the attack on two SubBytes outputs slightly harder than the attack on SubBytes output, as the first one needs to compare 2^{16} subkey candidates, thus favoring ghost peaks.

5.2.1 Experimental results

S-box outputs on Set A

Our first test targets the Set A, on four S-box outputs. Using the SNR curve (see 6), one point of interest (PoI) is selected per target byte: 31317, 25070, 45893, 33399. Figures 10, 11 and 12 show a characterization of the power model on each of these PoI. Figure 10 shows the average power consumption of each manipulated value (as we target the output of the AES SubByte operation, there are 256 possible values). Figure 11 shows the average power consumption of each manipulated Hamming weight. Finally, Figure 12 shows the influence of each manipulated bit on the power consumption. The consistency of the empirical leakage model estimations from a byte to another gives some confidence about the quality of the estimations. In other words, the empirical leakage model is close to the true leakage model, from a first-order point of view.

We observe that the estimated leakage model is very close to the Hamming weight model, which is an assumption of the method in [13]. The ratio between ϵ and σ is computed as following:

$$\frac{\epsilon}{\sigma} = \sqrt{\frac{\text{SNR}}{2}} \tag{5}$$

In our experiments, we found the following values:

Target	SB1	SB2	SB3	SB4
$\frac{\epsilon}{\sigma}$ value	1.0256	1.0955	0.6154	0.5811

If the true leakage model is the same for all targets, these values enable to classify the targets for the weakest (here SB2) to the more robust (here SB4).

The results are shown in Figure 13. We see that, for all bytes, the shortcut formula underestimates the effectiveness of the attack. However, the efficiency order from a byte to another is conserved (SB2 is the weakest, etc.).

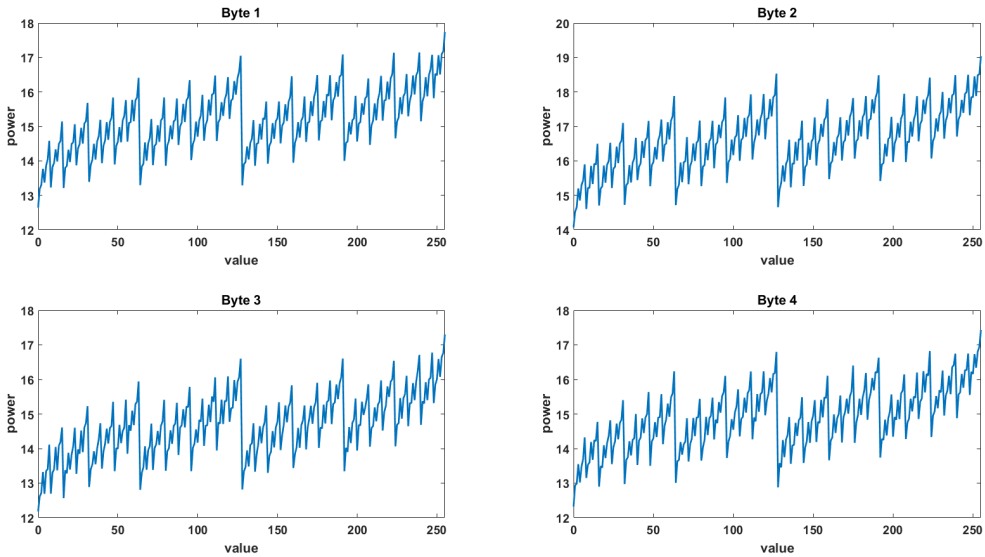

Figure 10: Estimated leakage model (value classification)

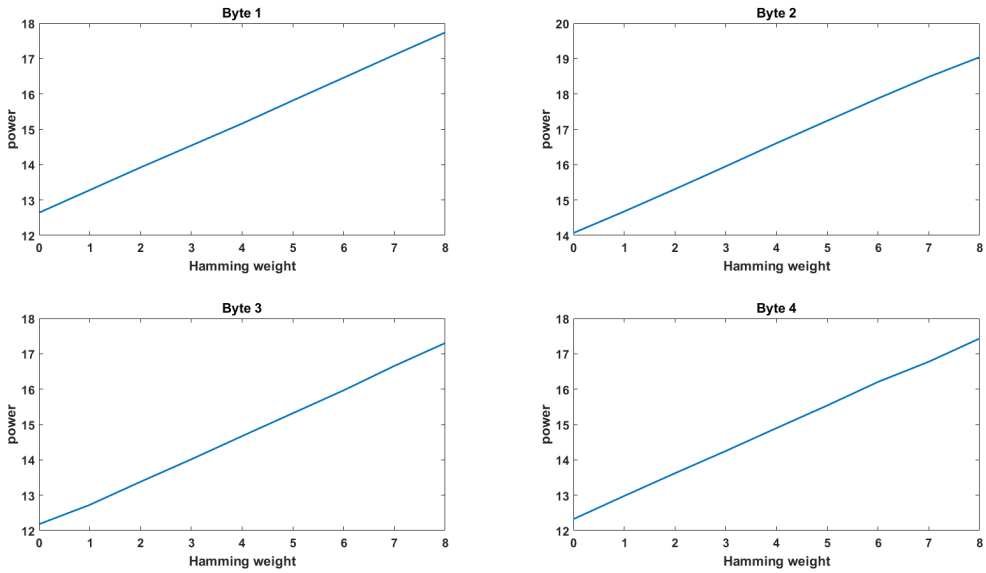

Figure 11: Estimated leakage model (HW classification)

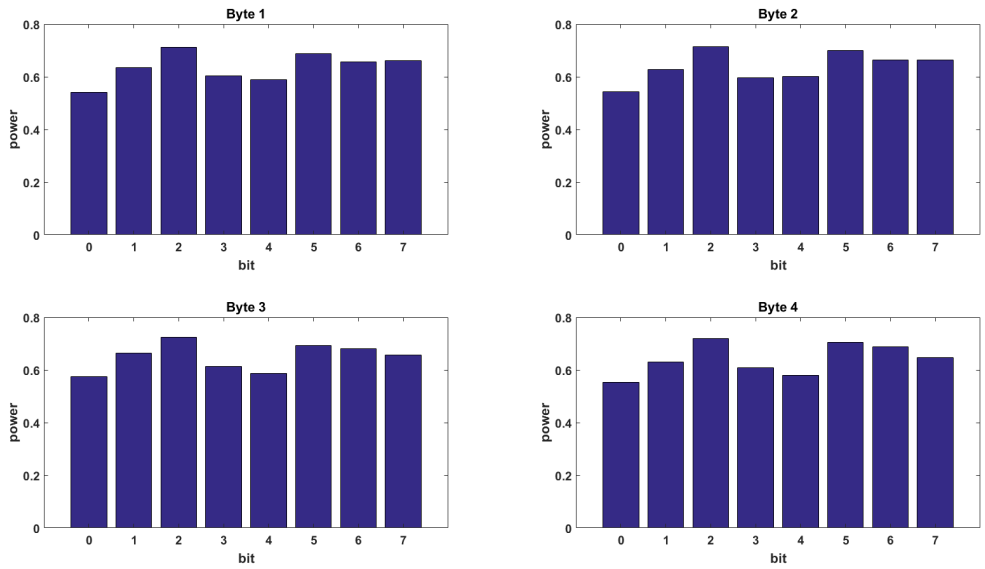

Figure 12: Estimated power influence of each bit

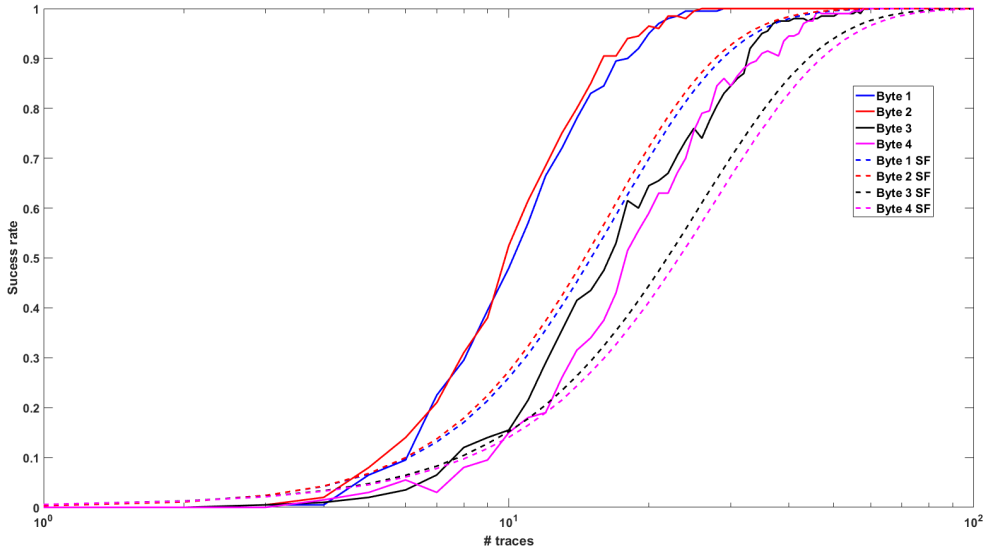

Figure 13: Success rates of real attack (plain curves) compared to expectations using shortcut formula (dashed curves)

S-box outputs on Set B

Our second test targets the Set B, on four S-box outputs. Again, we selected one point of interest per target byte (using Figure 8): 10076, 10119, 10160, 10206.
As previously, Figures 14, 15 and 16 show a characterization of the power model on each PoI.
This time, the true leakage model is further from the Hamming weight leakage model.

Target	SB1	SB2	SB3	SB4
$\frac{\epsilon}{\sigma}$ value	0.0570	0.0674	0.0652	0.0637

The results are shown in Figure 17. We see that, for all bytes, the shortcut formula overestimates the efficiency of the attack.

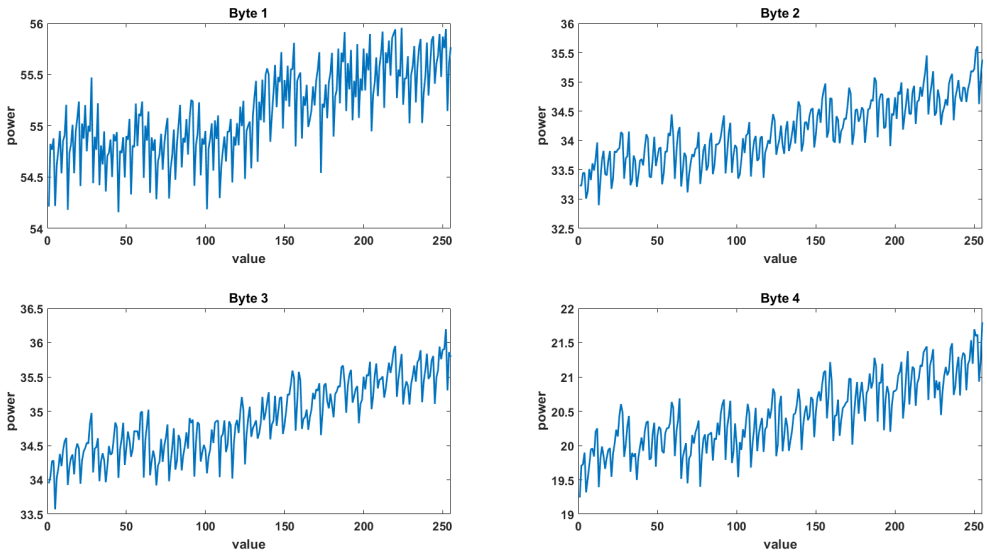

Figure 14: Estimated leakage model (value classification)

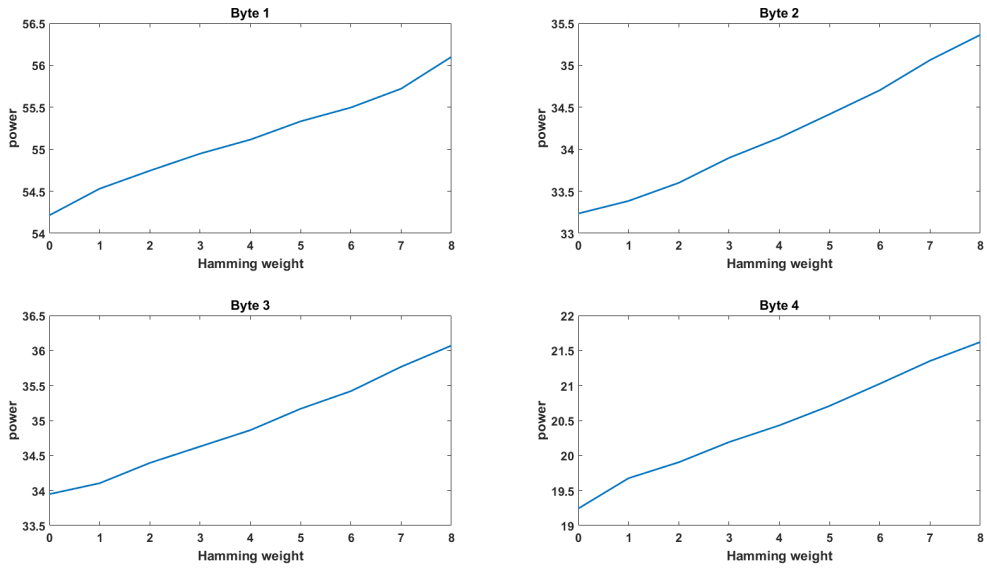

Figure 15: Estimated leakage model (HW classification)

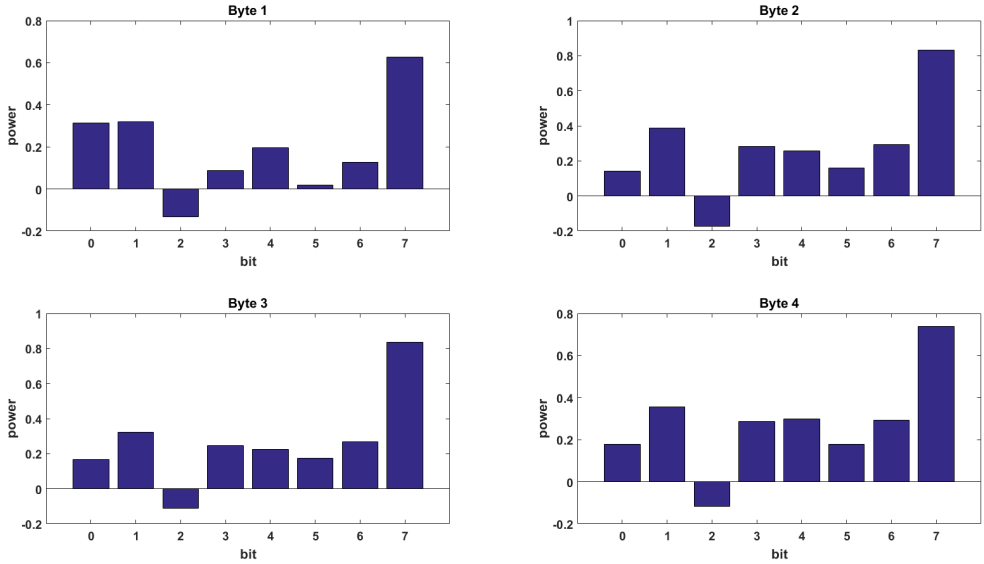

Figure 16: Estimated power influence of each bit

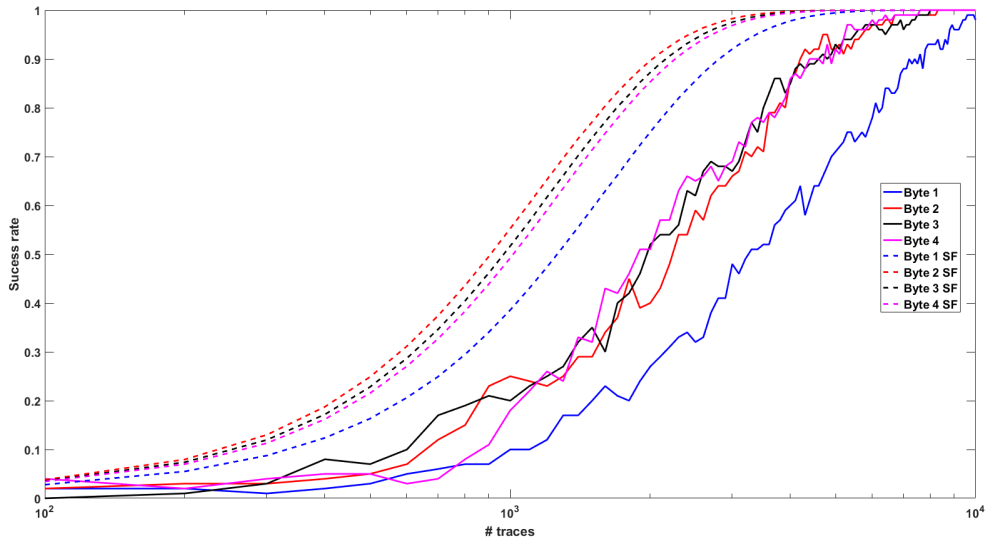

Figure 17: Success rates of real attack (plain curves) compared to expectations using shortcut formula (dashed curves)

5.2.2 Discussion

In this section, we applied shortcut formulas to compute first-order SR on a very low-noise case (leading to very high SNR) and on a higher-noise case.
The main observation here is that the computation of 100 true attacks is faster than evaluating the confusion-based method.
However, the shortcut formula needs a lower number of traces to estimate the SR. Thus, shortcut formula remains interesting if the number of acquisitions is limited, even if the SNR is very high.

5.3 Case of Second-Order Analysis

The following methodology has been applied:

- Estimate MI (by targeting two different intermediate values) on N_1 curves traces and deduce the SR thanks to the shortcut formula

- Mount a 2O-CPA on N_2 traces (N_2 can be deduced from the previous estimated SR, e.g. choose N_2 such that SR=0.9)
- Estimate the consistency of the two results and evaluate the saved time by using shortcut formula

Here, multivariate second-order analysis and univariate on second statistical moment have to be analysed independently. Furthermore, one can differentiate the case where $(X \oplus M, M)$ is targeted from the case where $(X \oplus M, Y \oplus M)$ is targeted, with X and Y sensitive variables, and M a random mask.

5.3.1 Experimental results

S-box outputs on Set A

Using the SNR curve (see 7), we select an additional point of interest, corresponding to the manipulation of the mask applied on S-box output: 50528. In this case, the MI metric gives a very strong overestimation of the true efficiency of the CPA attack. It can be explain by the fact that the noise condition is surely not fulfilled.
On the other hand, one can use the first-order empirical SNR (one for the masked variable, one for the mask) to compute the SNR after central product combination.
Let L_{t_0} and L_{t_1} be the observations of X and Y respectively, and let $L_{t_0,i} = \mathrm{E}[L_{t_0}|X = i]$ and $L_{t_1,i} = \mathrm{E}[L_{t_1}|Y = i]$.
Then, the SNR is computed as follows:

$$\begin{aligned} SNR =& \frac{A}{\mathrm{Var}\left[L_{t_0}\right] \times \mathrm{Var}\left[L_{t_1}\right] - A} \\ \text{with } A =& \mathrm{Var}_i\left[\mathrm{E}_j\left[(L_{t_0,j} - \mathrm{E}[L_{t_0}]) \times (L_{t_1,i\oplus j} - \mathrm{E}[L_{t_1}])\right]\right] \end{aligned} \tag{6}$$

This SNR can be used to estimate the success rate with the confusion matrices method. Note that the Hamming weight assumption is theoretically fulfilled, as the central product of two variables following the Hamming weight model still follows a Hamming weight model. However, it does not follow a Gaussian distribution anymore. Nevertheless, results in Figure 18 shows that this estimation is very close to the true attack outcomes.

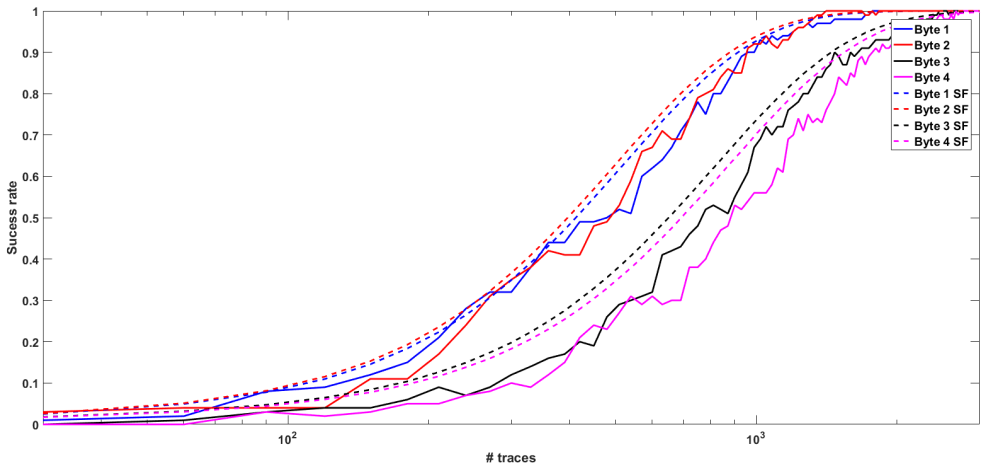

Figure 18: Success rates of real attack (plain curves) compared to expectations using shortcut formula (dashed curves)

S-box outputs on Set B

Using the SNR curve (see 9), we select an additional point of interest, corresponding to the manipulation of the mask applied on S-box output: 624. As the SNR is lower in the Set B, we are not able to compute an empirical success rate. Instead, for each of the first four target bytes, one attack has been mounted.
The results are shown in Figure 19. On the other hand, the expected success rates given by the MI metric and the confusion matrices are given in Figure 20.

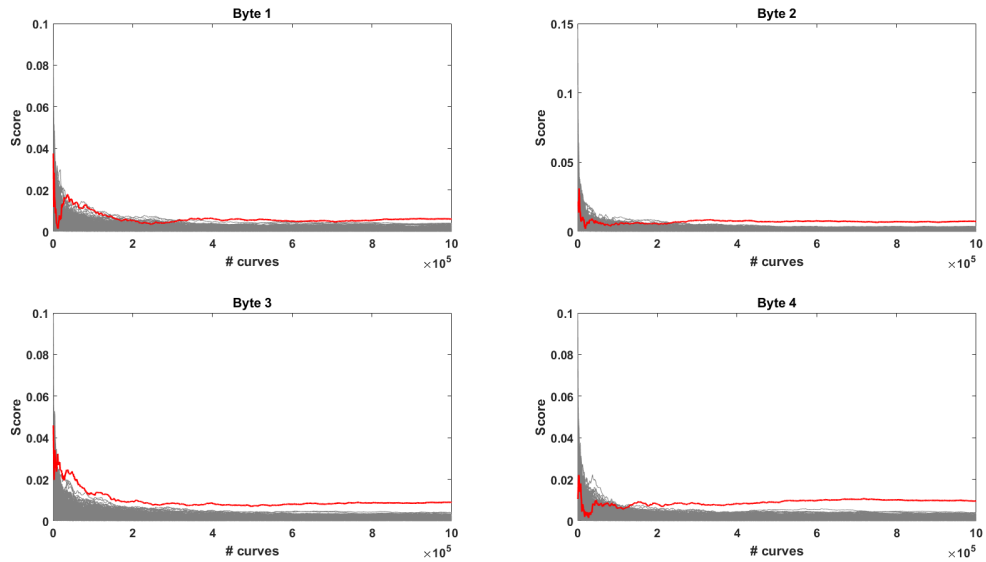

Figure 19: Correlation curves according to the number of processed traces (correct key in red)

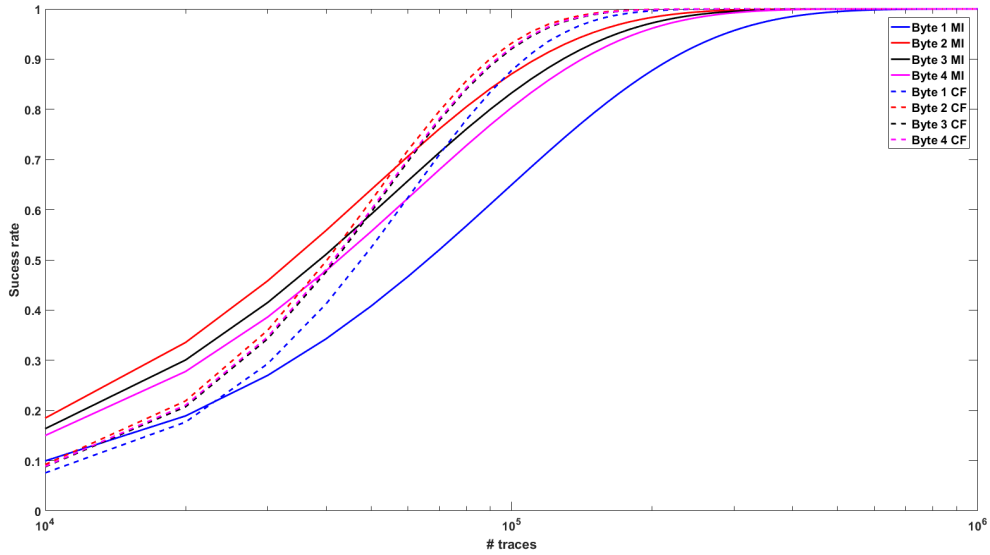

Figure 20: Expectations using MI (plain curves) compared to expectations using confusion matrices (dashed curves)

5.3.2 Discussion

In this section, we applied shortcut formulas to compute second-order SR on the same sets of traces.
The first observation is that such a high SNR does not allow the use of the MI-based method, which has a minimal level-of-noise condition. We saw that it can be replaced by the use of the confusion-based method using an estimated second-order SNR.
Second, we experimentally saw a case where the number of traces is insufficient to compute an empirical SR and where shortcut formulas are the only way to estimate it.

References

[1] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[2] Lejla Batina and Matthew Robshaw, editors. *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*. Springer, 2014.

[3] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2016.

[4] 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.

[5] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: normalized inter-class variance for detection of side-channel leakage. *IACR Cryptology ePrint Archive*, 2013:717, 2013.

[6] 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.

[7] Marios O. Choudary and P. G. Popescu. Back to massey: Impressively fast, scalable and tight security evaluation tools. In *CHES 2017*, pages 367–386, 2017.

[8] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, volume 10348 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2017.

[9] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.

[10] A. Adam Ding, Liwei Zhang, Yunsi Fei, and Pei Luo. A statistical model for higher order DPA on masked devices. In Batina and Robshaw [2], pages 147–169.

[11] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. 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 423–440. Springer, 2014.

[12] 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, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[13] Yunsi Fei, A. Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based success rate model for DPA and CPA. *Journal of Cryptographic Engineering*, 5(4):227–243, Nov 2015.

[14] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In *CHES 2012*, pages 233–250, 2012.

[15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 117–129, 2015.

[16] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with enumeration: Which one beats the other and when? In Iwata and Cheon [20], pages 291–312.

[17] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight (and how to exploit it in security evaluations). *IACR Cryptology ePrint Archive*, 2017:116, 2017.

[18] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In *CHES 2014*, pages 55–74, 2014.

[19] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[20] Tetsu Iwata and Jung Hee Cheon, editors. *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*. Springer, 2015.

[21] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2017.

[22] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In Batina and Robshaw [2], pages 35–54.

[23] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[24] S. Mangard, E. Oswald, and F-X. Standaert. One for All – All for One: Unifying Standard DPA Attacks. *IET Information Security*, 5(2):100–110, 2011.

[25] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *Topics in Cryptology - CT-RSA 2004*, pages 222–235, 2004.

[26] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*.

[27] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.

[28] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[29] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[30] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. Quantum key search with side channel advice. In *CT-RSA 2018*, 2018.

[31] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In *ASIACRYPT 2016 Part I*, pages 548–572, 2016.

[32] Daniel P. Martin, Ashley Montanaro, Elisabeth Oswald, and Dan Shepherd. Quantum key search with side channel advice. In *SAC 2017*, 2017.

[33] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Iwata and Cheon [20], pages 313–337.

[34] James L. Massey. Guessing and Entropy. *IEEE International Symposium on Information Theory*, page 204, 1994.

[35] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[36] 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, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2010.

[37] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. Algebraic side-channel attacks beyond the hamming weight leakage model. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2012.

[38] Romain Poussier, Vincent Grosso, and François-Xavier Standaert. Comparing approaches to rank estimation for side-channel security evaluations. In *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pages 125–142, 2015.

[39] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[40] 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, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.

[41] Matthieu Rivain. On the exact success rate of side channel analysis in the Gaussian model. In *SAC 2008*, pages 165–183, 2008.

[42] Y. Souissi L. Sauvage S. Guilley, H. Maghrebi and J. Danger. Quantifying the quality of side-channel acquisition, 2011.

[43] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016.

[44] F-X Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT '09*, pages 443–461, Berlin, Heidelberg, 2009. Springer-Verlag.

[45] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[46] François-Xavier Standaert, Eric Peeters, Gaël Rouvroy, and Jean-Jacques Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2):383–394, 2006.

[47] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. pages 21–36, 2013.

[48] 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, *Selected Areas in Cryptography*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.

[49] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security Evaluations beyond Computing Power. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.

[50] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014.

[51] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Information Enables Key Recovery. In *CARDIS 2014*, volume 7707 of *LNCS*. Springer, 2014.

A Code of Shortcut Formula

A.1 Confusion Coefficient Method

In this section, we introduce a Matlab implementation of the shortcut formula from [13]. To run an example, just launch the 'main.m' file.

```
%%%%%%%%%%%%%%%%%%
%     main.m     %
%%%%%%%%%%%%%%%%%%

% func -> handle of function, computing S(x \oplus y) with S the AES Sbox
func = @(x, y) AES_R1_sboxout(x,y);

% L -> leakage model
L = genHammingWeightModel(8);

% nbTraces -> vector containing the number of traces for which we want to evaluate the success rate
nbTraces = 5:5:500;

% kc -> correct subkey
kc = 42;

% SNR -> Signal-to-Noise Ratio, has be to estimated using side channel traces
SNR = 1;

% nbInputs -> number of possible inputs for x in func(x,y)
nbInputs = 256;

% nbGuesses -> number of guesses
nbGuesses = 256;


% Launch evaluation of SR
SR = SR_confusion(func, L, kc, SNR, nbTraces, nbInputs, nbGuesses);

figure, plot(nbTraces, SR);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     AES_R1_sboxout.m %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Compute S(M \oplus k) for each element of M and k
%
% M -> column vector (nbInputs x 1)
% k -> row vector      (1 x nbGuesses)
%
function P = AES_R1_sboxout(M, k)

    P = bitxor(M, k);
    P = AES_Sbox(P);

end

function M = AES_Sbox(M)

    S = [99 124 119 123 242 107 111 197 48 1 103 43 254 215 171 118 202 130 201 125 250, ...
         89 71 240 173 212 162 175 156 164 114 192 183 253 147 38 54 63 247 204 52 165, ...
         229 241 113 216 49 21 4 199 35 195 24 150 5 154 7 18 128 226 235 39 178, ...
         117 9 131 44 26 27 110 90 160 82 59 214 179 41 227 47 132 83 209 0 237, ...
         32 252 177 91 106 203 190 57 74 76 88 207 208 239 170 251 67 77 51 133 69, ...
         249 2 127 80 60 159 168 81 163 64 143 146 157 56 245 188 182 218 33 16 255, ...
         243 210 205 12 19 236 95 151 68 23 196 167 126 61 100 93 25 115 96 129 79, ...
         220 34 42 144 136 70 238 184 20 222 94 11 219 224 50 58 10 73 6 36 92, ...
         194 211 172 98 145 149 228 121 231 200 55 109 141 213 78 169 108 86 244 234 101, ...
         122 174 8 186 120 37 46 28 166 180 198 232 221 116 31 75 189 139 138 112 62, ...
         181 102 72 3 246 14 97 53 87 185 134 193 29 158 225 248 152 17 105 217 142, ...
         148 155 30 135 233 206 85 40 223 140 161 137 13 191 230 66 104 65 153 45 15, ...
         176 84 187 22]';

    size_in = size(M);
    M = S(M+1);

    if ~all(size_in==size(M)), M = M'; end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     genHammingWeightModel.m %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Return the Hamming weight of 0:2^size-1
%
function hw = genHammingWeightModel(size)

    hw = sum(dec2bin((0:2^size-1)')-48, 2);

end
```



```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%     SR_confusion.m     %
%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Compute success rate of a CPA targeting @func with the leakage model L
%
% Example of use
%
% func = AES_R1_sboxout();
% L = genHammingWeightModel(8); % leakage model
% kc = 42; % correct key
% SNR = 1;
% nbTraces = 5:5:500;
% SR = SR_confusion(func, L, kc, SNR, nbTraces, 256, 256);
%
function SR = SR_confusion(func, L, kc, snr, iN, nbInput, nbGuess)

    es_ratio = sqrt(snr/var(L,1));

    [CM, CMS, CMSS] = getConfusionMatrice(func, L, kc, nbInput, nbGuess);
    D = size(CM, 1);

    k = diag(CM);
    upp = 0.5*( es_ratio .^2) .*k;
    E   = ( es_ratio .^2)*CM + 0.25*(es_ratio .^4)*(CMSS - k*k');

    N = length(iN);
    SR = zeros(N, 1);
    for n=1:N
        SR(n) = my_mvncdf(repmat(-Inf, D, 1), sqrt(iN(n))*upp/ es_ratio , zeros(D, 1), E/( es_ratio ^2));
    end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     getConfusionMatrice.m        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Example of use:
%
% func = @(x, y) AES_R1_sboxout(x, y);
% L = genHammingWeightModel(8); % leakage model
% kc = 42; % correct key
% [CM, CMS, CMSS] = getConfusionMatrice(func, L, kc, 256, 256);
%
function [CM, CMS, CMSS] = getConfusionMatrice(func, L, kc, nbInput, nbGuess)

    M = (0:nbInput-1)';
    CM = zeros(nbGuess-1);
    CMS = zeros(nbGuess-1);
    CMSS = zeros(nbGuess-1);

    Vc = L(func(M, kc)+1);
    A = 4*(Vc - mean(Vc)).^2;

    gTab = [0:kc-1, kc+1:nbGuess-1];
    for i=1:nbGuess-1
        g1 = gTab(i);
        Vg1 = L(func(M, g1)+1);

        di1 = Vc - Vg1;
        di2 = di1.^2;

        for j=i:nbGuess-1
            if j==i
                CM(i,i) = mean(di2);
                CMS(i,i) = mean(di2.^2);
                CMSS(i,i) = mean(A.*di2);
            else
                g2 = gTab(j);
                Vg2 = L(func(M, g2)+1);

                dj1 = Vc - Vg2;
                dj2 = dj1.^2;

                CM(i,j) = mean(di1.*dj1);
                CM(j,i) = CM(i,j);
                CMS(i,j) = mean(di2.*dj2);
                CMS(j,i) = CMS(i,j);
                CMSS(i,j) = mean(A.*di1.*dj1);
                CMSS(j,i) = CMSS(i,j);
            end

        end
    end

end
```



```
%%%%%%%%%%%%%%%%%%%%%%
%     my_mvncdf.m    %
%%%%%%%%%%%%%%%%%%%%%%
%
% Matlab interface to call the Python function mvncdf.
%
function    est = my_mvncdf(low,upp,mu,Sig)

    mvn = py.importlib.import_module('scipy.stats.mvn');

    if isrow(low), low = low'; end
    if isrow(upp), upp = upp'; end
    if isrow(mu), mu = mu'; end

    low = mat2array(low);
    upp = mat2array(upp);
    mu = mat2array(mu);
    Sig = mat2array(Sig);

    est = cell(mvn.mvnun(low,upp,mu,Sig)){1};

end
```

```
%%%%%%%%%%%%%%%%%%%%%%
%     mat2array.m     %
%%%%%%%%%%%%%%%%%%%%%%
%
% Convert Matlab matrix M in a numpy array X
%
function X = mat2array(M)

    X = py.numpy.array(M(:)');

        dim = size(M);

    if numel(dim)>1
        X.shape = py.tuple(int32(dim));
    else
        X.shape = py.tuple({int32(dim)});
    end

end
```

