The Side-Channel Metric Cheat Sheet

KOSTAS PAPAGIANNOPOULOS[*], UvA, The Netherlands
OGNJEN GLAMOČANIN[*], EPFL, Switzerland
MELISSA AZOUAOUI[*], NXP Semiconductors, Germany
DORIAN ROS[*], EPFL, Switzerland
FRANCESCO REGAZZONI[*], UvA, The Netherlands and USI, Switzerland
MIRJANA STOJILOVIĆ[*], EPFL, Switzerland

Side-channel attacks exploit a physical observable originating from a cryptographic device in order to extract its secrets. Many practically relevant advances in the field of side-channel analysis relate to security evaluations of cryptographic functions and devices. Accordingly, many metrics have been adopted or defined to express and quantify side-channel security. These metrics can relate to one another, but also conflict in terms of effectiveness, assumptions and security goals. In this work, we review the most commonly used metrics in the field of side-channel analysis. We provide a self-contained presentation of each metric, along with a discussion of its limitations. We practically demonstrate the metrics on examples of relevant implementations of the Advanced Encryption Standard (AES), and make the software implementation of the presented metrics available to the community as open source. This work, being beyond a survey of the current status of metrics, will allow researchers and practitioners to produce a well-informed security evaluation through a better understanding of its supporting and summarizing metrics.

Additional Key Words and Phrases: Side-Channel Attacks, Side-Channel Analysis Metrics, Information Theoretic Metrics, Cryptography

1 INTRODUCTION

Embedded and cyber-physical devices pervade our lives in every aspect, including extremely critical ones such as identification, payment, health and safety. Contrary to personal computers and cloud servers, embedded devices are, at least potentially, in the hand of the attackers, which can exploit this physical access to extract secret information. Side-channel attacks, and in particular the attacks exploiting the relation between the power consumed by the device and the data being computed, have attracted significant attention from both the industrial and the academic communities [71]. This is due to their powerful nature (they can easily extract the complete key of a mathematically secure cryptographic algorithm) and their limited cost (the equipment needed is usually in the range of few thousands of Euros).

In this context, protecting from side-channel attacks is of utmost importance. Researchers designed countermeasures to mitigate these attacks as soon as they became known to the general public [43]. In parallel, research also proceeded in finding better and more efficient ways to extract secret information from devices. However, despite more than 20 years of research in the domain of side-channel attacks, the problem of assessing the robustness of certain implementations and the problem of comparing different attacks is still far from being solved. The main reason for this is the lack of a clear, "silver-bullet" metric for the specific problem. This is still true, despite the several attempts that the community has done to date to propose an objective and scalable metric for assessing side channel resistance. Metrics proposed so far may be suitable only in a particular evaluation

[*]All authors contributed equally to this research.

Authors' addresses: Kostas Papagiannopoulos, kostaspap88@gmail.com,k.papagiannopoulos@uva.nl, UvA, The Netherlands; Ognjen Glamočanin, ognjen.glamocanin@epfl.ch, EPFL, Switzerland; Melissa Azouaoui, melissa.azouaoui@nxp.com, NXP Semiconductors, Germany; Dorian Ros, dorian.ros@epfl.ch, EPFL, Switzerland; Francesco Regazzoni, f.regazzoni@uva.nl;regazzoni@alari.ch, UvA, The Netherlands, and USI, Switzerland; Mirjana Stojilović, mirjana.stojilovic@epfl.ch, EPFL, Switzerland.

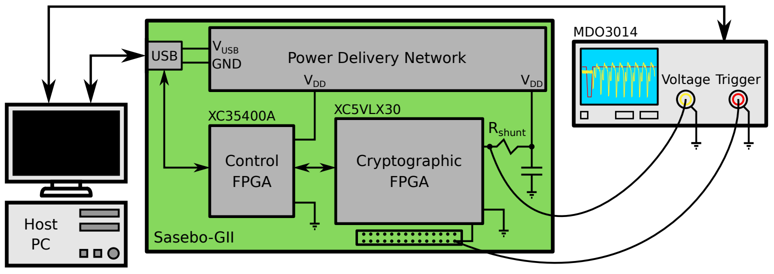


Fig. 1. Measurement setup used for acquiring the power side-channel traces on the Sasebo-GII board.

context, or allow only to draw specific conclusions which can not be generalized. Often, to have a complete picture, evaluators compute more than a single metric.

To extricate from the jungle of metrics is extremely difficult, due to the complexity of the subject (designers should in fact master several domains, including electronics, signal processing and statistics) and because of the high fragmentation of the related literature. Metrics for side channel attacks are in fact either the sole argument of a paper (for instance, when the metric is proposed [83] or analyzed [85]) or are shortly summarized where they are applied [64]. As a result, the current literature on the topic suffers from several limitations. Firstly, it does not provide a complete picture, because papers mostly focus on a single metric. Secondly, it is seldom self-contained; thus, it is hard for the designers to gain a complete understanding of the metrics by simply reading it. Finally, it is not accompanied by an open-source library that can be directly used by designers to evaluate their devices. As a result, despite it being extremely needed, an up to date review that reflects the current state of the art in the domain is missing.

Addressing all these limitations, in this paper we provide a critical review of the most common metrics proposed so far for the evaluation of side channel attacks. Our analysis takes the form of a cheat sheet: For each metric, we present the background, the goals, and the main characteristics (we demonstrate them with examples and experiments). Additionally, we discuss the limitations and report how the community moved forward proposing another metrics.

To obtain the traces used in the experiments for demonstrating the side-channel evaluation metrics, we record traces for three different AES Sbox implementations listed below:

- an unprotected AES Sbox that uses a 256-byte look-up table (*unprotected Sbox*),
- a masked AES Sbox (*masked Sbox 1*), introduced by Regazzoni et al. [72], and
- a masked AES Sbox (*masked Sbox 2*), introduced by Canright and Batina [13].

For the chosen Sbox implementations, we record *simulated* and *real* power side-channel traces. The simulated traces were produced with the Synopsys Nanosim simulator, on the post place and route circuit realized using the NanGate 45nm library. The simulation and the current resolution were set to 1 ps. The real traces were obtained using the Sasebo-GII side-channel evaluation board [30], as illustrated in Figure 1. All three Sbox implementations were deployed on the cryptographic FPGA (Xilinx Virtex-5), at 24 MHz clock frequency. The Tektronix MDO3104 oscilloscope was used to record side-channel traces, with 1 GHz sampling frequency and 8-bit encoding. To ensure the same initial conditions, the Sbox was reset before every experiment. In the case of masked implementations, fresh masks were generated for every encryption. Randomly generated plaintexts and masks were sent from the host PC to the cryptographic FPGA through the control FPGA (Xilinx Spartan-3A) via the USB interface. The trigger signal, raised by the cryptographic FPGA, enabled the coordination between the device under test and the oscilloscope.

To compute the side-channel analysis metrics from the traces, we used an in-house software library, which we make openly available [76].

The rest of the paper is organized as follows. We discuss attack-driven metrics in Sections 2, 4, 5, and 8. Metrics that capture the leakage of cryptographic implementations are covered in Sections 3 and 7. In Section 6, we explain how to utilize metrics in order to make theoretical security projections from the available experiments. Finally, in Section 9, we discuss qualitative metrics based on statistical tests.

2 NUMBER OF TRACES

As mentioned, side-channel attacks start from the physical observables of a device. Their goal is to identify in these quantities any bias about the cipher's intermediate values, for instance, finding whether a register bit is more likely to be zero than one. Subsequently, they exploit such bias to recover the secret key, using techniques that bear a close resemblance to linear and differential cryptanalysis [9, 59]. The bias observed in physical quantities of unprotected devices is often much higher compared to the bias observed when cryptanalyzing a cipher without monitoring such quantities. Therefore, side-channel attacks are colloquially referred to as *cryptanalysis with a large bias*, vividly portraying their strong potential and their relationship with standard cryptography. Given that side-channel attacks are in fact a branch of cryptanalysis, it comes naturally to adjust the existing security metrics from that field.

2.1 Description

The common metrics that reflect the security of a cipher are the time, memory and data complexity required for an attack [37]; that is, the number of encryptions, the needed storage size and the number or type of plaintexts and ciphertexts. Translating the time complexity into the side-channel field results in the metric of *measurement complexity*. This metric tells us how many times we need to perform encryption (or decryption)—while measuring a certain side-channel—so as to hack the device and recover its secret key [84]. The literature refers to side-channel measurements as *traces*, therefore the measurement complexity is typically stated as *number of traces*, one of the most common metrics of side-channel resistance.

When considering symmetric ciphers like AES-128, the starting attack point in cryptanalysis (*without* side-channels) is a brute-force attempt, with time complexity of 2^{128} encryptions (worst-case), or equivalently 128 bits of security. From this point on, specialized cryptanalytic attacks aim to reduce the time complexity, usually with a varying degree of success: Attacks can range from total breaks to small improvements that can just distinguish the cipher from a truly random permutation [42]. Such cryptanalytic attacks (*without* side-channels) can be carried out on any device since they only require the attacker to implement and execute the cipher. Multiple encryptions can be carried out in parallel, thus computer clusters constitute prime candidates for the cryptanalytic computational effort. Note also that some attacks use random plaintext while others require it to be constant or known.

In the same spirit, cryptanalysis *with* side-channels aims to reduce the time complexity of AES-128 from 2^{128} encryptions to a substantially smaller number of encryptions, typically by using random plaintexts. Unlike standard cryptanalysis though, a side-channel attack requires physical measurements that can only be performed on the device under test (DUT), aided by a measurement apparatus (e.g., resistors, EM probes, oscilloscopes, etc.). This close relation between side-channel attacks and devices means that speeding up or parallelizing the cipher encryption is typically not an option and the attacker is essentially limited by the encryption throughput of the particular device. Therefore, the *number of traces* metric contains an implicit single-device limitation and thus a *hidden* measurement cost. Analytically, the processing speed of the device, the input/output communication delays and any overhead due to the measurement apparatus may slow down the side-channel cryptanalysis and result in

Table 1. Resistance of the *unprotected AES Sbox* in number of traces and security bits, taking into account the measurement cost.

Implementation	Number of traces	Security (in bits)
Unprotected AES Sbox (**fast** measurements)	15,000	24
Unprotected AES Sbox (**slow** measurements)	15,000	44

a bottleneck[1]. Estimating precisely this cost is case-dependent with optimistic side-channel attackers placing it at 2^{10}, i.e., carrying out an encryption on a specific device (while measuring the side-channel) is approximately 1000 times slower than simply carrying out the same encryption (without measuring the side-channel) on a high-end processor. More pessimistic attackers place this cost up to 2^{30} [34]. In general, we can conclude that a slow device-under-test or delays during measurement or simply the attacker's inability to parallelize the encryption can give the defender 10 to 30 bits of additional security.

Despite the hidden measurement cost in the *number of traces* metric, it is the most common approach to measure side-channel security. Secure chip manufacturers, evaluation labs and researchers apply a wide variety of techniques that eventually boil down to the statement: *"We are able to recover the secret key using X side-channel traces."* Several certification bodies and governmental agencies have integrated this metric into their guidelines, for instance, the current ISO/IEC 17825:2016 [2] [1, 97] proposes the usage of specific side-channel analysis techniques and requires that no security flaws are found after measuring 10,000 or 100,000 traces, depending on the desired security level. Following a different school of thought, methodologies like JHAS [91] try to account for the hidden measurement cost (among other things) by specifying the time (in days or months) at the attacker's disposal, instead of the number of traces.

In Table 1 we show the usage of the metric. We see that breaking the unprotected AES Sbox and recovering its secret requires approximately 15,000 traces (i.e., 15,000 encryptions). Phrasing it in bits, this implies a measurement of complexity of $\log_2(15,000) = 2^{13.87}$, i.e., almost 14 bits of security. To this we can add the hidden measurement cost that we place optimistically at 10 bits (fast measurements), totalling $14 + 10 = 24$ bits of security. Performing 2^{24} encryptions is computationally easy, thus we can safely conclude that the unprotected device does not offer adequate security. Assuming (pessimistically) a hidden measurement cost of 30 bits (extremely slow measurements due to, e.g., intentional delays) gives us $14 + 30 = 44$ bits of security, which implies stronger security, albeit at a huge time overhead.

2.2 Limitations

Number of traces and attack dependency. Even such a popular metric is by no means a standalone panacea and has several limitations. The obvious deficit of the *number of traces* (and most side-channel metrics) is the fact that it is linked to a specific side-channel attack or technique. Therefore, a more truthful statement is the following: *"Deploying attack A, we are able to recover the secret key using X side-channel traces."* Unfortunately, the deployed attack A may be suboptimal, i.e., it may not effectively exploit the side-channel leakage and the device may appear more resistant than it actually is[3]. Even worse, certain attacks may fail when side-channel

[1]Notably, designers have even introduced encryption delays on purpose, in order to slow down the side-channel adversary. This option is particularly popular when encryption speed is not critical to the overall performance of the device. E.g., the User Identifier encryption in iOS 9 was designed to take 80 milliseconds to delay adversaries [48].

[2]ISO/IEC 17825:2016 [ISO/IEC 17825:2016] Information technology - Security techniques - Testing methods for the mitigation of non-invasive attack classes against cryptographic modules.

[3]In fact, side-channel research has identified cases where certification schemes were found lacking, since they propose suboptimal analysis techniques [97].

countermeasures are present, while different lines of attack may achieve key recovery. Motivated by such pitfalls, certification schemes like EMVCo [4] recommend that side-channel evaluations stay up-to-date with recent attack methods, encouraging a holistic analysis effort[5].

Note also that attacks are often linked to the attacker's computational capabilities, thus it is often hard to distinguish between the inherent leakage of the device and the attacker's ability to exploit it effectively. The inherent leakage of the device is particular to the electronic and electrical layer: It refers to the actual physical phenomena that cause leakage. In contrast to this, the attacker's computational capabilities may enable her to obtain a very large amount of traces, to train a computationally intense model and to search the key space quickly. This coalescence between the device leakage and the attacker's capabilities makes it harder to obtain a clear picture of the vulnerability. As a result, it becomes harder to identify countermeasures that perform well, irrespective of the attacker exploiting them. In light of such deficits, we stress that every side-channel metric should always be considered in the context of 1) the device-under-test and 2) the attack that produced it. Forgetting this may lure the evaluator into a false sense of security, since most metrics cannot fully convey the attack strength on their own. We postpone the discussion about attacker-independent side-channel metrics until Section 7, where we will introduce metrics that alleviate this problem.

Number of traces and key recovery. Another crucial deficit of the *number of traces* metric is its vagueness regarding key recovery. Going back to the security statement, we observe that the phrase *"We are able to recover the secret key"* is lacking a clear context. Attaching a precise meaning to it necessitates the construction of new metrics, namely the *score* and *rank* of key candidates after an attack. This topic is examined in Sections 4 and 5.

3 SIGNAL TO NOISE RATIO

To a large extent, the discussion about side-channel metrics is motivated and influenced by the available attack methods and their limitations. Thus, in most sections we examine metrics in conjunction with the attack that produced them. Still, before delving into that approach, we here digresses to a more intuitive and standalone side-channel metric, the signal-to-noise ratio (SNR). The *SNR* is not computed as part of an attack and it requires only a few modeling assumptions.

3.1 Description

The *SNR* metric can be obtained directly from the traces of any side-channel experiment, without the need to launch an attack. The evaluator has typically measured a dataset with numerous side-channel traces, each containing multiple time samples. Side-channel analysis is interested in a subset of these samples, specifically in the samples that leak information about the intermediate values of a cipher. These intermediate values are in turn linked to the secret key, thus the information contained in them is critical for security[6]. In order to define the *SNR*, we must first model the leakage L of an interesting time sample. We will use the following standard modeling approach:

$$L = L_d + L_n = g(V) + L_n, \text{ where } L_n \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

[4]https://www.emvco.com/

[5]Such recommendations have been triggered by the fairly recent interest into deep-learning based approaches for side-channel attacks [7, 55].

[6]The relationship between the cipher's intermediate values and the key can be more or less straightforward, depending on the cipher under attack. E.g., assume that we have recovered any 8-bit Sbox output of the first round of AES-128 through side-channel analysis. This implies that we can directly recover the respective 8-bit Sbox input, since the Sbox is a bijective function. Continuing, we can directly recover the respective 8-bit part of the secret key during the AddRoundKey operation, since the plaintext is known and we just have to XOR it with the Sbox input to reach the key. On the contrary, the 4-bit Sbox output of the DES cipher does not directly map to the respective part of the secret key, since DES uses a 6 to 4 bit Sbox. Recovering the Sbox output results only in partial information about the key.

In the formula above, the random variable L represents the leakage at a certain time sample and it is assumed to be the sum of two components. The first component, L_d, is known as the deterministic part (or the data part) and is linked to the cipher's intermediate value V, which leaks at this point in time. The leakage behavior of V is specified by a function $g(\cdot)$, referred to as the *leakage function*. When modeling leakage in theory, common choices for $g(\cdot)$ are the identity function, the Hamming weight and linear models. The second component, L_n is known as the non-deterministic part (or the noise) and it encapsulates various effects in the DUT and the measurement apparatus that produce variation. It is often modelled as a normal distribution with zero mean and variance σ^2. We stress that such assumptions about L, L_d and L_n are by no means the only way to model the leakage. Still, using this simple model, we can proceed to define the *SNR* metric:

$$SNR = \frac{Var(L_d)}{Var(L_n)} \tag{2}$$

The *SNR* is defined as the ratio between the variance of the data (deterministic part) and the variance of the noise (non-deterministic part). The metric shows intuitively the ratio between the useful and the useless part of the signal. The goal of the attacker is to improve their setup and increase the *SNR*, while the goal of the defender is to decrease it, hiding the useful signal in noise. Below we explain how to compute the *SNR* in two scenarios: simulated leakage and real measurements. We also discuss NED and NSD, two variants of the *SNR* metric.

***SNR* with simulated leakage.** In this theoretical scenario, the evaluator is simulating herself the leakage L by specifying precisely L_d and L_n. For example, she simulates the leakage of an 8-bit value V (uniformly random). To that end, she chooses a leakage function $g(V) = \alpha\ V + \beta$, where α, β are known constants. That is, $L_d = \alpha\ V + \beta$, a scaled version of identity leakage (by factor α), that is offset by factor β. In addition, she chooses $L_n \sim \mathcal{N}(0, \sigma^2)$ and specifies the numerical value of the parameter σ^2. In this simulation, we can proceed to compute the *SNR* directly:

$$E(L_d) = E(g(V)) = \sum_{v=0}^{255} g(v)\ Pr(v) = \frac{1}{256}\sum_{v=0}^{255}(\alpha v + \beta) \qquad E(L_d^2) = E(g(V)^2) = \frac{1}{256}\sum_{v=0}^{255}(\alpha v + \beta)^2 \tag{3}$$

$$SNR = \frac{Var(L_d)}{Var(L_n)} = \frac{E(L_d^2) - E(L_d)^2}{\sigma^2} \tag{4}$$

***SNR* with real measurements.** In this practical scenario, the evaluator has no a priori knowledge (or control) of L_d and L_n. Since she can only observe their sum L, she cannot compute directly the variance of L_d and L_n. For example, she has carried out an experiment observing the leakage L of an 8-bit value V, obtaining n traces $l_1, \ldots, l_n$, each trace l_i containing a single time sample that leaks information about V. By assuming that the deterministic part of the leakage can be described by a function g which depends only on the value of the intermediate V, we have that:

$$L = g(V) + L_n, \text{ that is } (L|V = v) \sim \mathcal{N}(g(v), \sigma^2) \tag{5}$$

We have no knowledge of L_d, L_n and thus we don't know the leakage function $g(\cdot)$ and variance σ^2 of the noise. Still, we see that setting $V = v$ in Equation 5 will result in normally distributed leakage with mean $g(v)$ and variance σ^2. Using this observation, our new goal is to estimate this mean and variance using the n available traces. In particular:

(1) We partition the n traces into 256 groups, based on the value of V and we obtain the sets $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_{255}$. The set $\mathcal{L}_v$ contains all traces l that are observed when $V = v$.

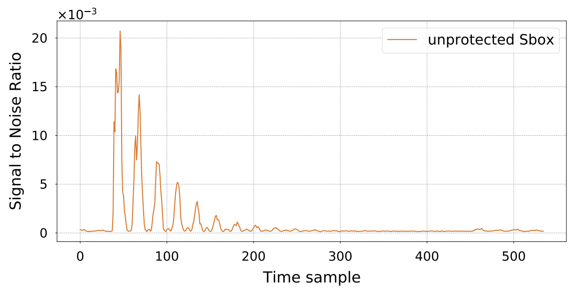


Fig. 2. *SNR* for the *unprotected AES Sbox*.

(2) We estimate the statistical mean $\hat{\mu}$ (the hat notation over μ denotes estimation using the dataset) of every set $\mathcal{L}_v$ and the overall mean. This results in $\hat{\mu}_0, \hat{\mu}_1, \ldots, \hat{\mu}_{255}$ and $\hat{\mu}$:

$$\hat{\mu}_v = \frac{1}{|\mathcal{L}_v|} \sum_{l \in \mathcal{L}_v} l \qquad \hat{\mu} = \frac{1}{256} \sum_{v=0}^{255} \hat{\mu}_v, \tag{6}$$

where $|\mathcal{L}_v|$ is the size of set $\mathcal{L}_v$, i.e., the number of traces s.t. $V = v$.

(3) We estimate the statistical variance of every set $\mathcal{L}_v$, resulting in $\hat{\sigma}_0^2, \hat{\sigma}_1^2, \ldots, \hat{\sigma}_{255}^2$ and compute their average, $\hat{\sigma}^2$

$$\hat{\sigma}_v^2 = \frac{1}{|\mathcal{L}_v| - 1} \sum_{l \in \mathcal{L}_v} (l - \hat{\mu}_v)^2 \qquad \hat{\sigma}^2 = \frac{1}{256} \sum_{v=0}^{255} \hat{\sigma}_v^2 \tag{7}$$

After these estimation steps, we can proceed with the *SNR* computation

$$SNR = \frac{Var(L_d)}{Var(L_n)} = \frac{\sum_{v=0}^{255} (\hat{\mu}_v - \hat{\mu})^2}{\hat{\sigma}^2} \tag{8}$$

Using the real dataset of the *unprotected Sbox* and the estimation steps, we compute the *SNR* for the whole trace length (Figure 2). We observe that several time samples exhibit peaks (*SNR*≈ 0.02), indicating the presence of leakage, while other time samples exhibit negligible *SNR*.

NED and NSD metrics. There are two additional metrics that bear a close resemblance to *SNR*, namely *normalized energy deviation* (NED) and *normalized standard deviation* (NSD). They are defined as follows:

$$NED = \frac{max(L) - min(L)}{max(L)} \tag{9}$$

$$NSD = \frac{Var(L)}{E(L)} \tag{10}$$

The functions $max(\cdot)$, $min(\cdot)$ denote the maximum and minimum leakage during a single time sample. Both NED and NSD have been utilized when analyzing specific CMOS styles [52, 88] that aim to resist side-channel attacks. If such an analysis is based on a noiseless leakage simulation, the leakage L is equal to its useful part L_d. In this case, both metrics intuitively convey the variation of L_d, i.e., high NED/NSD implies strong leakage and low NED/NSD implies weak leakage. If however, the leakage L contains noise, then NED/NSD become less useful, since its unclear whether the leakage variation originates from L_n or L_d.

3.2 Limitations

***SNR* and side-channel attacks.** The inherent strength and weakness of *SNR* is its relationship with side-channel attacks. The metric is not linked by default[7] to a specific side-channel attack, thus it can maintain some level of generality. On the downside, the lack of an actual attack means that *SNR* cannot directly convey the actual security level.

***SNR* and the leakage model.** Although the leakage assumptions required to compute the *SNR* are fairly reasonable, we still rely implicitly on a certain leakage model. Equation 1 makes assumptions about the components L_d, L_n, i.e., the leakage function $g(\cdot)$ and the noise distribution. Thus, in a simulated leakage scenario, where $g(\cdot)$ and the noise are under the evaluator's control, the *SNR* metric is accurate by construction. On the contrary, in a real measurement scenario, the leakage may diverge from our model's assumptions. Therefore, we would need to test if the model is a good statistical fit and adjust it, if needed. Moreover, even if the leakage fits the simple model precisely, the statistical parameters (mean and variance) of the distributions $\mathcal{N}(g(v), \sigma^2)$ need to be adequately estimated from the traces. A poor estimation (due to a limited number of traces) may, in turn, produce misleading results. A detailed discussion about such practical has been carried out by Lomné et al. [50]. We return to the issue of model choice and parameter estimation in Section 7. An additional modeling limitation of the *SNR* metric is its univariate definition: The metric focuses on a single time sample at a time, ignoring joint multivariate leakage. Unfortunately, the adversary is not as limited and various attacks are able to effectively exploit multiple samples [15]. To adapt, the *SNR* metric has been extended in the multivariate context [89], which in turn results in additional assumptions and constraints.

4 SCORE & RANK

We now return to the question posed in the end of Section 2 and clarify the process of *"recovering the secret key."* To this end, we describe a standard side-channel attack and develop the score and rank metrics.

4.1 Description

The majority of side-channel attacks have the same modus operandi. First, they use a divide-and-conquer strategy [11, 43], and they partition the full cipher key in small parts that are easy to attack. For example, instead of recovering directly the full 16-byte key of AES-128, the attack divides it in 16 parts and tries to recover every one of the bytes individually. For every 8-bit key part, the attack will produce scores for every key candidate, while ranking the candidates from best to worst. Throughout this overview, we denote key candidates using the variable K and its variable instances using k. Below, we go through the steps of an attack that recovers a single key byte K of AES-128.

(1) We focus on an 8-bit value, thus our set of key candidates k is limited to $\mathcal{K} = \{0, 1, \ldots, 255\}$.
(2) The attack on the 8-bit key value produces 256 scores $[score_0, score_1, \ldots, score_{255}]$, where $score_k$ is the attack score of the key candidate k. For example, $score_{42}$ is the attack score achieved by the key candidate $k = 42$.
(3) Subsequently, the attack uses the scores to guess which key candidates are the best, producing 256 guesses $[guess_1, guess_2, \ldots, guess_{256}]$, starting from guess 1. The quantity $guess_1$ is equal to the key candidate that scored the best during the attack and is thus the prime choice for the true key. For instance, if the best score during the attack came from candidate $k = 42$, then $guess_1 = 42$.

[7]It is worth noting that the *SNR* metric can be tweaked and used as a side-channel distinguisher of its own accord (https://ileanabuhan.github.io/leakage/detection/2021/05/27/SNR-distinguisher.html)

Algorithm 1: The workflow of a standard side-channel attack.

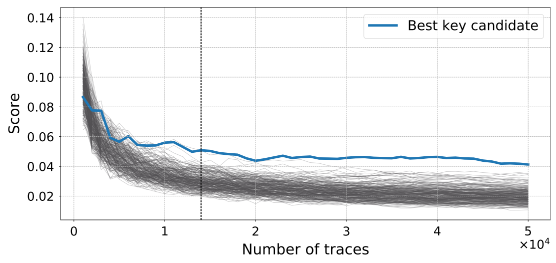
```
Input: Attack score function $f(\cdot)$
Input: Key candidates $\mathcal{K}$, with set size $|\mathcal{K}|$
Data: Simulated or real side-channel traces
Output: $score_k, guess_k, rank_k$, for all $k \in \mathcal{K}$ and all key partitions
▷ Partition the full cipher key and attack all key parts
$partitions$ = divide-and-conquer($full\ key$)
for $all\ partitions$ do
    ▷ Compute the attack score for every key candidate
    for $all\ k \in \mathcal{K}$ do
        $score_k = f(traces, k)$
    ▷ Sort the scores and compute the key guesses
    $[score_i, score_j, \dots, score_m]$ = sort($[score_0, score_1, \dots, score_{|\mathcal{K}|-1}]$)
    $[guess_1, guess_2, \dots, guess_{|\mathcal{K}|}] = [i, j, \dots, m]$
    ▷ Compute the rank of every key candidate using the key guesses
    for $all\ guess_i$, $i \in \{1, 2, \dots, |\mathcal{K}|\}$ do
        $rank_{guess_i} = i$
```

(4) Finally, we can produce a vector $[rank_0, rank_1, \dots, rank_{255}]$, where $rank_k$ is the rank of key candidate k and the best possible rank equals 1. For example, if the best score came from $k = 42$ (and thus $guess_1 = 42$), then $rank_{42} = 1$. By convention, rank equal to 1 indicates the best key candidate.

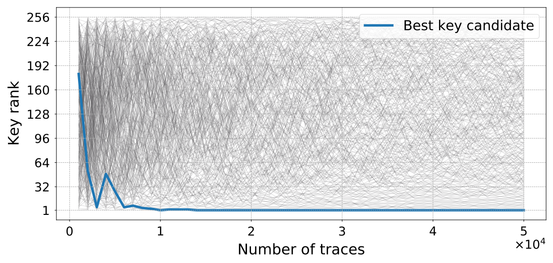
This process of scoring and ranking is typically repeated for all bytes of the key; that is, in our AES-128 example, it will be performed 16 times (partitioning the full key to 16 key byte-sized parts). Algorithm 1 describes this attack process.

Using the *score* and *rank* of a key candidate can clarify the phrase *"We recover the secret key with X traces."* Figures' 3a and 3b y-axes display respectively the score and rank metrics after performing the common correlation power analysis attack (CPA) [11] on a single key byte of the AES-128 cipher. The x-axis displays the number of traces used in the current side-channel attack. What we typically look for in such graphs is a key candidate that stands out compared to the rest. That is, we search for a candidate that achieves the highest score (or the lowest rank, namely rank 1) and maintains this top score (or rank 1) as we keep increasing the number of traces. Such convergent behavior typically implies that our attack has managed to distinguish a key candidate from the rest, thus it concluded successfully. In Figures 3a and 3b, the key candidate $K = 203$ is reaching the highest score and rank 1 after an attack that uses approximately 15,000 traces. Increasing the number of attack traces (up to 50,000) does not alter the situation, i.e., we observe convergent behavior. With the aid of these graphs we can state that *"We recover a byte of the secret key with 15,000 traces"* and thus we attach a concrete meaning to our security statement. From this point we can compute the security level in bits as $\log_2$(number of traces to converge) and create a table similar to Table 1.

Keep in mind that the best key candidate $K = 203$ according to Figures 3a, 3b is still a guess. In our current situation, the attacker cannot be certain about the correctness of the guess, unless she performs a full key recovery (as demonstrated in Algorithm 1) that recovers all 16 key bytes that constitute the full key. Using the full key, the attacker should be able to verify whether her guesses can correctly decrypt the ciphertext and thus, reach certainty. Still, it is a very common practice in the literature to attack a single key partition (typically a byte) and subsequently extrapolate the security of the remaining partitions. Such an extrapolation is motivated by



(a) Scores of 256 key candidates. The candidate $K = 203$ (blue line) reaches the maximum score around 15,000 traces.



(b) Ranks of 256 key candidates. The candidate $K = 203$ (blue line) reaches rank 1 around 15,000 traces.

Fig. 3. Scores and ranks of all key candidates for the *unprotected Sbox*, under correlation power analysis.

time limitations as well as by the strong leakage similarities between intermediate cipher values which are typically processed and protected in similar ways. E.g., in the case of an unprotected AES-128, we have no strong indications that another Sbox of the implementation is more (or less) resilient to side-channel analysis. Thus, we could limit the scope of our analysis to a single key byte recovery and avoid the repetitive full key recovery.

4.2 Limitations

Score, rank and statistical stability. Despite their widespread usage, the score and rank metrics do not come without pitfalls and there exist cases where they do not convey the full security picture. The first issue encountered is the statistical stability of the metrics. Any score or rank graph is typically produced using a side-channel traceset under a fixed cipher key. It is not unlikely that other key values are more (or less) easy to recover. Thus, a good experimental practice is to repeat the side-channel measurements using different keys and/or datasets and subsequently repeat the attack, computing the average rank/score together with its standard deviation [28]. This statistical stability guideline is recommended in other side-channel metrics as well.

Score, rank and the unknown key analysis. The second issue with the score and rank metrics becomes apparent when the performed side-channel attack is suboptimal. Assume the following case: Our attack does not exploit the traces in the best way possible or perhaps it just does not have enough traces to do so. Therefore, the attack may fail to place the correct key at rank 1. Still, if the correct key is placed at a fairly low rank (say rank 2 or 3), one could conclude that the implementation is vulnerable, since a brute-force attack is feasible. The underlying issue is that in such closed evaluations the full device key is unknown. The only way to find the key is to launch an attack, acquire the rank 1 key candidates of all 16 key bytes, use them to construct the full key and finally verify the correctness of the full key using, e.g., cipher decryption[8]. If this fails, one may repeat the full key verification process after including key candidates that ranked 2, 3, etc., as long as they can manage the computational cost of trying many different combinations of key candidates. By construction, the rank and score metrics are focused on a key partition (e.g., a single key byte), therefore they cannot be linked directly to the security of the full cipher key. For example, even after acquiring the ranks of all key candidates for all AES-128 key bytes $\big[[rank_0^{byte_0}, \ldots, rank_{255}^{byte_0}], \ldots, [rank_0^{byte_{15}}, \ldots, rank_{255}^{byte_{15}}]\big]$, it is not straightforward to derive the rank of the full 16-byte key. Finding the security of the full key when the key is unknown requires a

[8]The topic of key verification is not heavily detailed in the literature. Most works assume that the attacker can use mechanisms like cipher decryption, a cryptographic protocol or a communication layer that, once executed with the correct full key, can easily confirm whether the full key is correct indeed.

process of trial and error that is known as *key enumeration* [93] and is often time-consuming, as discussed in Section 8.

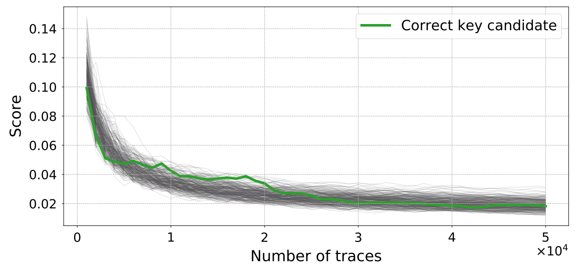
To reduce the workload of side-channel evaluations and to avoid the lengthy key enumeration, analysts try to avoid (if possible) the unknown key scenario. What they do instead is to perform a more open evaluation where they know the full cipher key beforehand. After performing any side-channel attack on a key byte, they can find the distance between the key candidate that achieved rank 1 and the correct key candidate. This evaluation is often referred to as *known-key analysis*. Although it appears as slightly counter-intuitive, it helps us find a better answer to the question *"Can we recover the secret key?"*. Instead of replying with simple *"yes/no"* we are also able to show how far we are from key recovery, enabling a more precise risk assessment. As expected, the concept of *known-key analysis* necessitates new side-channel metrics, such as the *success rate* and the *guessing entropy* that we examine in the next section (Section 5).

5 SUCCESS RATE & GUESSING ENTROPY

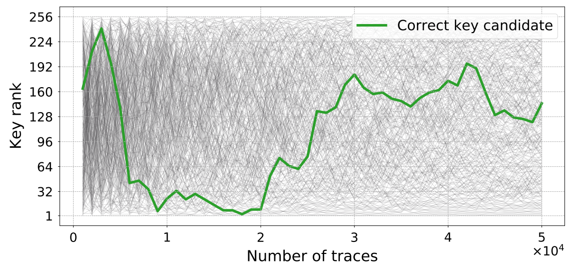
As discussed, in a *known-key analysis*, the evaluator has a priori access to the key she is trying to recover. At first it appears paradoxical for the evaluator to demand the key of a device that she is tasked to break, since an attacker would not (and should not) have access to it anyway. The notions of *evaluator* and *attacker* are largely synonymous, yet here we observe the first divergence between them. The goal of the attacker is to recover the key, while the goal of the evaluator is to assess how hard it is to find the key, even if she is not able to do so. This divergence motivates the *known-key analysis* and its respective metrics, namely the *known-key score*, the *known-key rank*, the *success rate* (*SR*) and the *guessing entropy* (*GE*). Similar considerations will resurface in Sections 6 and 7, where we consider cases in which the available leakage is not enough to perform key recovery, prompting new metrics that highlight again the difference between attacker and evaluator.

5.1 Description

The known-key versions of the score and rank metrics are often used when the analyst wants to show the security of a cryptographic implementation. Figures 4a and 4b, highlight the score and rank of the correct key, after performing CPA on the protected *masked Sbox 1* [9]. The figures show that the correct key is unable to achieve a maximum score (or equivalently rank 1) consistently, even with 50,000 attack traces. Since convergence is not achieved, we can conclude that this protected Sbox implementation remains secure against the particular CPA attack with 50,000 traces.



(a) Score of the correct key (green line). We observe that it does not achieve the maximum score.



(b) Rank of the correct key (green line). We observe that it is unable to converge to rank 1.

Fig. 4. Score and rank of the correct key for the *masked Sbox 1*, under correlation power analysis.

[9]Highlighting the correct key is possible because we now perform a known-key analysis.

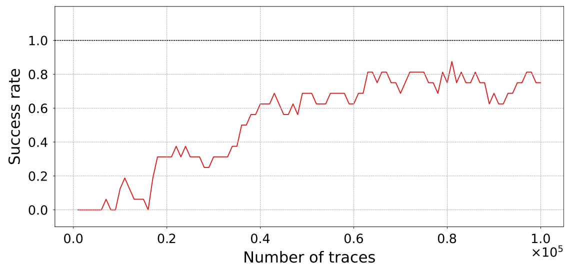


Fig. 5. Success rate (*SR*) of the correlation power analysis for 16 parallel unprotected Sboxes.

Like the known-key score and rank, the *SR* metric requires the correct key to be known and is computed in the following way. Assume a side-channel attack trying to recover a key byte of AES-128. Recall that that the output of a standard side-channel attack (Algorithm 1) is a guess vector $[guess_1, guess_2, \ldots, guess_{|\mathcal{K}|}]$ and let us assume that the correct key is equal to k_c. In its simplest form (Equation 5.1) the success rate of the side channel experiment no. i (i.e., SR^i) is equal to 1 if the best guess is equal to the correct key, that is if $guess_1 = k_c$. Otherwise, the success rate is equal to 0. One can easily derive an equivalent expression for SR^i using the rank of the correct key (that is $rank_{k_c}$) as seen in Equation 11. To ensure statistical stability, it is common practice to repeat the SR^i computation using multiple experiments and various keys. The final success rate metric (*SR*) is estimated with Equation 12, where p denotes the total number of experiments that are averaged.

$$SR^i = \begin{cases} 1, & \text{if } k_c = guess_1 \\ 0, & \text{otherwise} \end{cases} \qquad SR^i = \begin{cases} 1, & \text{if } rank_{k_c} = 1 \\ 0, & \text{otherwise} \end{cases} \qquad (11) \qquad SR = \frac{1}{p}\sum_{i=1}^{p} SR^i \qquad (12)$$

Figure 5 shows the success rate of a CPA attack on the unprotected AES implementation that uses 16 parallel Sboxes. The metric is computed by attacking all its 16 unprotected AES Sboxes under fixed key and averaging the respective SR^is. We observe that increasing the number of attack traces, improves the success rate gradually, reaching $SR \approx 0.8$ with 100,000 traces. Getting closer to $SR = 1$ implies a high probability of correct key recovery .

Strictly speaking, key recovery implies that the side-channel adversary is able to recover the exact key, i.e., $SR = 1$ and the correct key k_c converges to rank 1 after increasing the number of traces used in the attack. However, we want the *SR* metric to reflect also the cases where the correct key is ranked fairly high, yet less than rank 1. Figure 5 for instance shows that *SR* (with 100,000 traces) across multiple Sbox instances equals to 1 only 80% of the time. Thus, the correct key is ranked first often, but not always. To solve this issue, the *SR* metric was extended with the notion of *order*. Given a guess vector $[guess_1, guess_2, \ldots, guess_{|\mathcal{K}|}]$, we can define the *SR* of order o in the experiment no. i in the following manner.

$$SR_o^i = \begin{cases} 1, & \text{if } k_c \in [guess_1, guess_2, \ldots, guess_o] \\ 0, & \text{otherwise} \end{cases} \qquad SR_o^i = \begin{cases} 1, & \text{if } rank_{k_c} \leq o \\ 0, & \text{otherwise} \end{cases} \qquad (13) \qquad SR_o = \frac{1}{p}\sum_{i=1}^{p} SR_o^i \qquad (14)$$

In other words, if the correct key is found within the top o key guesses, then SR_o considers the attack successful. If we set the order o to 1, then the attack is successful only if the correct key comes first, i.e., we revert to our first definition of *SR* (Equation 11).

There exists an interesting link between an evaluator with *SR* of order o and an actual attacker. If the evaluator (using a known key) finds that the success rate of order o is equal to 1, then the attacker (with unknown key) has a maximum of o key candidates to verify (i.e., enumerate), after applying the same side-channel attack.

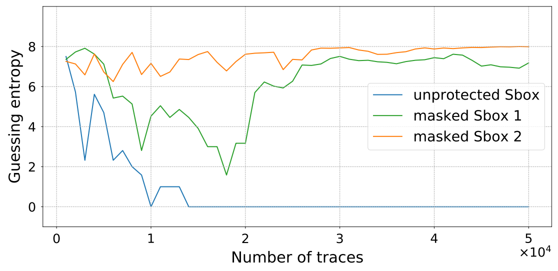


Fig. 6. The *guessing entropy* of the 8-bit secret key, after correlation power analysis on an *unprotected Sbox* (blue line) and two protected AES Sbox implementations: *masked Sbox 1* (green) and *masked Sbox 2* (orange).

That is, if an evaluator attacks a single byte of AES and achieves $SR_5 = 1$, then the attacker must try at most 5 candidates in order to find the correct key byte. However, such a key enumeration process must be repeated for all divide-and-conquer partitions of the full key and all possible key combinations must be verified until the correct combination is found. For example, if the evaluator achieves $SR_5 = 1$ for all 16 key bytes of AES-128, then an attacker that uses the same attack has to verify at most 5^{16} full keys to succeed. Searching among that many key candidates is an inherently heuristic task that becomes computationally hard as the order o increases. Although efficient key enumeration strategies exist, we stress that selecting an appropriate value for the order o reflects the attacker's computational ability to enumerate keys and it can be particularly challenging to decide in advance.

To steer clear from the heuristic process of enumeration, evaluators bypass the order of SR by using the more flexible metric of *guessing entropy* (GE) [45, 56]. In particular, given a rank vector $[rank_0, rank_1, \ldots, rank_{|\mathcal{K}|-1}]$ and the correct key k_c, the evaluator can always find the rank of k_c ($rank_{k_c}$) and plot it against the number of traces used in the attack, as in Figure 4b. Most evaluations focus only on the rank of the correct key and ignore the rank of other (incorrect) key candidates, therefore the term *rank plot* typically signifies the *known-key rank plot* and thus, a known-key analysis. In addition, evaluators repeat the rank computation using multiple experiments with various keys and ensure statistical stability by computing the average rank. The *guessing entropy* metric can be directly derived from the rank and average rank of the correct key ($rank_{k_c}$). For a certain experiment i, GE^i is equal to $\log_2(rank_{k_c})$ and it is typically expressed in bits (Equation 15).

$$GE^i = \log_2(rank_{k_c}) \qquad (15) \qquad\qquad GE = \frac{1}{p}\sum_{i=1}^{p} GE^i \qquad (16)$$

In the case of the GE metric (Equation 16), p is the number of different experiments and GE^i the rank of the correct key during experiment i. It is recommended to change the correct key k_c between experiments.

The $rank_{k_c}$ and GE metrics indicate the average remaining workload needed after the side-channel attack. Figure 6 depicts the GE against the number of traces, when attacking three different AES Sboxes. In the case of the *unprotected Sbox*, GE reaches 0 bits after 15,000 traces, indicating that the side-channel attack has removed all uncertainty about the secret key. On the contrary, the GE of the masked Sboxes does not reduce substantially below 8-bits (even after 50,000 traces) and indicates resistance against the attack.

5.2 Limitations

***SR*, *GE* and the full key.** The first pitfall of the SR and GE metrics is evident should you ask what is their relation to the security of the full cipher key. In other words, given the *guessing entropy* of all AES-128 key bytes $[GE^{byte_0}, GE^{byte_0}, \ldots, GE^{byte_{15}}]$ after an attack, is it possible to compute the GE of the full key? Unfortunately, like the rank and score metrics introduced in Section 4, *SR* and *GE* do not directly convey the effort of a full key recovery. Reaching conclusions for the full key requires the process of *key rank estimation*, which can be considered as the known-key equivalent of *key enumeration*. In the same way that we use the known-key analysis to move from rank and score metrics to *SR* and *GE* metrics, we can use our knowledge of the full cipher key in order to perform the relatively fast process of rank estimation instead of the slow key enumeration. Additional details are provided in Section 8.

***SR*, *GE* and security projections.** The second pitfall of *SR* and *G*E is more subtle compared to other metrics. Consider the following scenario. First the evaluator performs a side-channel attack on AES-128 trying to recover the value of a single key byte. If the attack is partially successful she will observe an increase in *SR* (and decrease of *GE*), leading to a statement in the lines of: *"Deploying attack A, using X traces, we achieve Z % success rate in recovering the secret key"*. However, if the *SR* percentage is close to that of a random key guess (i.e., 1/256), the evaluator can solely conclude that *"Deploying attack A, using X side-channel traces, we were unable to recover the secret key"*. In other words, she cannot estimate if one would be able to recover the key using a larger amount of traces. Note, however, that cryptanalysis favors the idea of *projected* security bounds, i.e., even if one is not close to breaking a cipher, one should still be able to project approximately the remaining effort. Although *GE* is an expression of the remaining effort, like *SR*, it is derived from a specific number of measured traces. For both metrics, the underlying issue is that they are always linked to the number of attack traces at hand and they cannot be directly used for security projections. The necessity for such security bounds led to the development of new metrics and to stronger theoretical ties between the attack scores and the number of traces. Such links are examined in Sections 6 and 7.

6 CORRELATION AS A METRIC

A desirable property of security metrics is flexibility. That is, a good side-channel metric should be able to encompass various side-channel attacks, without involving their particular distinguishers and methods. It is easy to see that *rank*, *score*, *success rate* and *guessing entropy* fulfill this criterion, since they are agnostic to the underlying attack process[10]. However, Section 5.2 laid out a strong limitation on these metrics: In the case of a failed attack, they cannot convey any information other than their inability to recover the key. Thus, they cannot assist us to project the security level and estimate how many traces would the adversary need (theoretically) to break the device.

To fill in this gap, researchers have investigated the links between attack techniques and side-channel metrics, trying to derive theoretical security bounds [53]. This work has already made use of CPA (Section 4), one of the most popular side-channel attacks that relies on the *correlation statistic* to extract the correct key candidate. Expanding, this section will demonstrate useful theoretical links between the *correlation statistic* and attack metrics such as the *success rate* and the *number of traces* needed to recover the key. The advantage of using the *correlation statistic* directly (instead of performing a full CPA attack) is that this statistic can be estimated, irrespective of the success or failure of the CPA attack. Thus, it can be used to provide a security projection, even when the attack itself is not successful.

6.1 Description

Correlation and number of traces. The work of Mangard [53] focuses on predicting the number of traces for the success of CPA attacks using the Pearson's correlation coefficient ρ. During a CPA attack, the evaluator estimates the correlation between two datasets. The first dataset is the observed device leakage $l_1, l_2, \ldots, l_n$ for

[10]Note that agnostic here does not mean independent. The metrics are still directly linked to the attack, yet exchanging one attack for another can still produce such metrics.

n traces that relate to the leakage of an intermediate value V. The second dataset is the modeled leakage $g(v_i)$ of n instances of the intermediate value V, using the leakage function $g(\cdot)$ and the correct key candidate k_c. The intermediate value under attack is typically a function of a known input $x \in \mathcal{X}$ and the key candidate, i.e., $v_i = f(x_i, k)$. We denote the correlation under the correct key candidate as ρ_{k_c}.

$$\rho_{k_c} = \frac{\sum_{i=1}^{n}\left(l_i - \frac{1}{n}\sum_{i=1}^{n} l_i\right)\left(g(f(x_i,k_c)) - \frac{1}{n}\sum_{i=1}^{n} g(f(x_i,k_c))\right)}{\sqrt{\sum_{i=1}^{n}\left(l_i - \frac{1}{n}\sum_{i=1}^{n} l_i\right)^2}\sqrt{\left(g(f(x_i,k_c)) - \frac{1}{n}\sum_{i=1}^{n} g(f(x_i,k_c))\right)^2}} \tag{17}$$

Any CPA attack will succeed in recovering the key if the absolute value[11] of ρ_{k_c} is higher than the correlation of other candidates. A simplifying assumption used in [53] is that the number of traces that is needed to recover the secret key is mainly determined by the distance between the estimated ρ_{k_c} for the correct key and $\rho = 0$, since the correlation of incorrect key candidates should be negligible and thus very close to zero. Assuming that n traces were used to estimate ρ_{k_c}, we can phrase this as a statistical test for the ability to distinguish the correct key among all key candidates using a set of n' traces and reach the shortcut formula in Equation 18. The quantity n' is the projected number of traces that an attacker would need to extract the key using CPA[12].

$$n' = 3 + 8\left(\frac{z_\alpha}{\ln\left(\frac{1+\rho_{k_c}}{1-\rho_{k_c}}\right)}\right)^2 \quad \text{where } Z \sim \mathcal{N}\left(\mu = \frac{1}{2}\ln\left(\frac{1+\rho_{k_c}}{1-\rho_{k_c}}\right),\ \sigma^2 = \frac{1}{n-3}\right) \tag{18}$$

The quantity z_α is the α% quantile[13] and the type I error α is set (via consensus) between 10^{-1} and 10^{-4}. Note also that the correlation of the correct key (ρ_{k_c}) is the only score that must be estimated, thus the evaluator avoids the lengthy process of computing the scores of all key candidates.

Correlation and Success Rate. Similarly to [53], the work of Rivain [74] focuses on predicting the success rate SR of a CPA attack using the absolute correlation scores ρ of all key candidates. Like before, assuming uniformly distributed inputs to the cipher, an intermediate value $v = f(x, k)$, leakage model $g(\cdot)$ and inputs $x \in \mathcal{X}$, the evaluator can acquire experimentally a set $\mathcal{L}$ with n traces $l_1, l_2, \ldots, l_n$. Carrying on with a normal attack, the evaluator can use this dataset to estimate a custom correlation coefficient $\ddot{\rho}_k$ for all key candidates $k \in \mathcal{K}$.

$$\ddot{\rho}_k = \frac{1}{n}\sum_{i=1}^{n} g(f(x_i, k))\ l_i \tag{19}$$

The key k that maximizes the coefficient $\ddot{\rho}_k$ is then chosen by CPA as the best candidate. The work in [74] shows that the vector $\ddot{\boldsymbol{\rho}} = [\ddot{\rho}_0, \ddot{\rho}_1, \ldots, \ddot{\rho}_{(|\mathcal{K}|-1)}]$ follows a multivariate normal distribution with mean vector

[11]Positive or negative correlation are equally useful for a side-channel attack, thus this section considers only the absolute value of the statistic.
[12]We use the notation n' instead of *number of traces* to indicate that the two metrics are not equivalent, since the first is projected and the second is not.
[13]$Pr(Z > z_\alpha) = \alpha$

$\boldsymbol{\mu}_{\ddot{\rho}} = [\mu_0, \mu_1, \ldots, \mu_{(|\mathcal{K}|-1)}]$ and covariance matrix $\Sigma_{\ddot{\rho}} = (\Sigma_{ij})_{0\le i,j\le|\mathcal{K}|-1}$, defined in Equations 20 and 21.

$$\mu_k = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} g(f(x,k))\ \hat{\mu}_v \quad \text{where } v = f(x,k_c) \ \text{ and } \ \hat{\mu}_v = \frac{1}{|\mathcal{L}_v|}\sum_{l\in\mathcal{L}_v} l \tag{20}$$

$$\Sigma_{ij} = \frac{1}{n'|\mathcal{X}|}\sum_{x\in\mathcal{X}} g(f(x,i))\ g(f(x,j))\ \hat{\sigma}_v^2 \quad \text{where } v = f(x,k_c) \ \text{ and } \ \hat{\sigma}_v^2 = \frac{1}{|\mathcal{L}_v|-1}\sum_{l\in\mathcal{L}_v}(l-\hat{\mu}_v)^2 \tag{21}$$

Note that the quantities $\hat{\mu}_v$ and $\hat{\sigma}_v^2$ are both estimated using the n experimentally acquired traces $\mathcal{L}$, and $\mathcal{L}_v$ is the subset of these traces that are observed when $V = v$. Observe also that Equation 21 includes the quantity n', i.e., the projected number of traces that could be used by an adversary to recover the key. After setting n' to a specific theorized value that reflects the strength of the adversary, we can link $\ddot{\boldsymbol{\rho}}$ to the (projected) *SR* metric. To do so, we take multiple random samples from the distribution $\mathcal{N}(\boldsymbol{\mu}_{\ddot{\rho}}, \Sigma_{\ddot{\rho}})$. Every sampled vector can be subsequently sorted from highest to lowest value and then the projected rank of the correct key ($rank_{k_c}$) can be derived, as described in Algorithm 1. Using $rank_{k_c}$ and Equation 11 or 13, we can approximate the projected *SR* or SR_o of a CPA attack that utilizes n' traces.

6.2 Limitations

Shortcut formulas and assumptions. Naturally, the bounds described in this section are not always accurate due to assumption errors. That is, the underlying simplifications may not always hold on a real device due to various electrical/electronic effects. In addition, the statistical estimation of the correlation coefficient (or any statistic used as a score) is always imperfect, with these errors reducing the reliability of the shortcut formulas. Note that several approaches for shortcut formulas have been proposed in the literature. These include analytical formulas [74], shortcuts for template attacks [74], extensions that account for the cipher's structure [29] and for masking [29, 49]. A collection of similar links and shortcuts formulas can be found in [90]. Finally, one should keep in mind that these bounds only describe the *projected* number of traces needed to distinguish the correct key with a high probability and are meant to be used either before an experiment or when the experimental traces are not able to guarantee key recovery.

7 INFORMATION-THEORETIC METRICS

Before defining the information-theoretic metrics for side-channel analysis, it is worth mentioning the two main reasons behind their development and adoption. The first was the evolution of side-channel analysis, which managed to produce strong multivariate attacks like templates [15, 18] and linear regression analysis [23, 78], often assisted by dimensionality reduction techniques like PCA, LDA and SSA [2, 69]. Such attacks were often able to outperform the univariate CPA attack and can recover the secret key using a substantially smaller number of traces. Naturally, before reaching a conclusion about side-channel security, evaluators want to apply the strongest attack in their arsenal. Extending metrics such as correlation and the *SNR* to a multivariate setting is not always straightforward; therefore, it necessitated new metrics that can capture multivariate attacks and provide a fair picture of the adversarial capabilities.

The second (and probably more ambitious) goal behind information-theoretic metrics was the desire to distinguish between A) an implementation that leaks side-channel information and B) the adversary that exploits such information. So far in this text we have deliberately avoided distinguishing between these two components, typically putting them both under the umbrella term *side-channel attack*. Very often though, the evaluator is interested in comparing between cryptographic implementations and countermeasures (A), irrespectively of the adversary's strength (B), a task that cannot be easily carried out by existing metrics. In other words, information-theoretic metrics were developed so as to answer the question *"How secure is a certain implementation against* *side-channel attacks?"* Such formulation helps us to shift away from metrics like *SR* and *GE* that are linked to the specific attack at hand and to the processing/storage/enumeration capabilities of the adversary. Instead, the new metrics will assist us to draw generic conclusions about the implementation that can be applied in various adversarial cases. Such a separation between implementation and adversary can be viewed as yet another divergence between the attacker (whose main concern is full key recovery in a certain device) and the evaluator, who must often consider the security of various devices and countermeasures so as to propose a generic security solution.

7.1 Description: Mutual Information - Part 1

The introduction of information-theoretic metrics in the context of side-channel analysis came with the adoption of the Mutual Information (MI) metric [83]. As discussed, we would like the new information-theoretic metric to capture multivariate leakages, i.e., attacks which utilize multiple time samples. Thus, we switch from the leakage variable L to the leakage vector $\mathbf{L}$ and its instances $\mathbf{l}$ that typically take values in $\mathbb{R}^m$, where m is the number of time samples used in the side-channel analysis. We now proceed with the definition of the mutual information metric, which quantifies how much information (in bits) we can learn about the secret key K by observing the multivariate side-channel leakage $\mathbf{L}$.

Using the definitions of mutual information and conditional entropy [20, 46]:

$$MI(\mathbf{L};K) = H(K) - H(K|\mathbf{L}) = E(-\log_2 Pr(K)) - E(-\log_2 Pr(K|\mathbf{L})) \tag{22}$$

$$= \left(-\sum_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k)\right) - \left(-\sum_{k\in\mathcal{K},\mathbf{l}\in\mathcal{L}} Pr(k,\mathbf{l})\log_2 Pr(k|\mathbf{l})\right) \tag{23}$$

Applying the Bayes Rule $Pr(k,\mathbf{l}) = Pr(\mathbf{l}|k)Pr(k)$ and converting the sum to an $m-$dimensional integral[14] we get:

$$MI(\mathbf{L};K) = -\sum_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k) + \sum_{k\in\mathcal{K}} Pr(k)\int_{\mathbf{l}\in\mathbb{R}^m} Pr(\mathbf{l}|k)\log_2 Pr(k|\mathbf{l})\mathrm{d}\mathbf{l} \tag{24}$$

Observing Equation 24, we see that the evaluator needs to compute several quantities. The entropy and conditional entropy $H(K)$, $H(K|\mathbf{L})$ require the knowledge of $Pr(k)$, for all $k \in \mathcal{K}$. For example, in the case of a single-byte key attack, it holds that $\mathcal{K} = \{0,1,\ldots,255\}$ and if key is uniformly random, then $Pr(k) = 1/256$, for all $k \in \mathcal{K}$. In general, in the very common case of uniformly random key, it holds that $H(K) = \log_2 |\mathcal{K}|$ and $Pr(k) = 1/|\mathcal{K}|$. Note that sometimes the evaluator is interested in recovering the Hamming weight of the secret key, say secret variable W, where $W = \text{HammingWeight}(K)$. In this case, she needs to adjust the probability as $Pr(w) = \binom{\log_2|\mathcal{K}|}{w}/|\mathcal{K}|$. For instance, the probability that the Hamming weight of a single byte is w is equal to $Pr(w) = \binom{8}{w}/256$. Returning to Equation 24, the evaluator must also compute the quantities related to the conditional entropy $H(K|\mathbf{L})$. This requires the knowledge of $Pr(\mathbf{l}|k)$ and $Pr(k|\mathbf{l})$. The quantity $Pr(\mathbf{l}|k)$ shows how likely the leakage vector $\mathbf{l}$ is when the key is k. The quantity $Pr(k|\mathbf{l})$ can be expressed as a function of $\Pr(\mathbf{l}|k)$ using the Bayes rule and the marginal probability definition as follows:

$$Pr(k|\mathbf{l}) = \frac{Pr(k,\mathbf{l})}{Pr(\mathbf{l})} = \frac{Pr(\mathbf{l}|k)Pr(k)}{Pr(\mathbf{l})} = \frac{Pr(\mathbf{l}|k)Pr(k)}{\sum\limits_{k^*\in\mathcal{K}} Pr(\mathbf{l},k^*)} = \frac{Pr(\mathbf{l}|k)Pr(k)}{\sum\limits_{k^*\in\mathcal{K}} Pr(\mathbf{l}|k^*)Pr(k^*)} \tag{25}$$

[14]Typically we define the leakage space $\mathcal{L}$ as continuous, therefore we use integration. In cases of simulated leakage or oscilloscope-quantized leakage values, the leakage $\mathbf{l}$ may take discrete values over a finite set $\mathcal{L}$. In such cases we keep the summation and Equation 22 becomes: $MI(\mathbf{L};K) = -\sum\limits_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k) + \sum\limits_{k\in\mathcal{K}} Pr(k)\sum\limits_{\mathbf{l}\in\mathcal{L}} Pr(\mathbf{l}|k)\log_2 Pr(k|\mathbf{l})$.

If the key K is uniformly random, then $Pr(k) = Pr(k^*) = 1/|\mathcal{K}|$. Thus Equation 25 can be simplified to $Pr(k|\mathbf{l}) = \frac{Pr(\mathbf{l}|k)}{\sum\limits_{k^* \in \mathcal{K}} Pr(\mathbf{l}|k^*)}$ and the *MI* formula can be restated as

$$MI(\mathbf{L}; K) = \log_2 |\mathcal{K}| + \sum_{k \in \mathcal{K}} \frac{1}{|\mathcal{K}|} \int\limits_{\mathbf{l} \in \mathbb{R}^m} Pr[\mathbf{l}|k] \log_2 \frac{Pr(\mathbf{l}|k)}{\sum\limits_{k^* \in \mathcal{K}} Pr(\mathbf{l}|k^*)} \mathrm{d}\mathbf{l} \tag{26}$$

In Equation 26, the only remaining quantity for the evaluator to compute is $Pr(\mathbf{l}|k)$, that is the likelihood of a side-channel leakage, given a certain key. Such computation is directly linked to the concepts of leakage profiling, leakage modeling and parameter estimation that we introduce briefly in the following Section (7.2).

7.2 Profiled Side-Channel Attacks

Many side-channel attacks are referred to as *profiled* attacks because of their different modus operandi, compared to standard attacks described in Section 4. Like the attacks introduced in Algorithm 1, they use a divide-and-conquer strategy to partition the full cipher key in parts that are easy to guess. However, instead of attempting to recover the key directly, they choose to profile the leakage of the device first, trying to learn as much information as possible before launching the actual attack. Such a preparation phase requires an *open* profiling device that is *identical* to the DUT. Both terms (open, identical) carry a specific meaning that needs specific highlighting in the context of such attacks.

The term *open* implies that the evaluator has full control over the device and can set the secret key at will. Note that an *open-device analysis* is slightly different to the *known-key analysis*: The evaluator can set any key on the profiling device but not necessarily on the device she plans to attack. In fact, the open device is only needed to learn the leakage behavior of different keys and such behavior will be subsequently used in recovering the secret key on the DUT, regardless of whether the key there is known or not. Contrarily, the known-key analysis has knowledge of the key on the DUT because the evaluator needs to assess how far she is from key recovery, typically using *SR* or *GE* metrics. Still, a very common profiled attack configuration is to use the same open device for profiling and attacking, i.e., the open-device analysis inherently implies a known-key analysis. We shall refer to this scenario as *same-device analysis*.

Regarding the term *identical*, it typically suggests that the two devices (open profiling device, closed device-under-attack) are technologically identical, i.e., they possess the same design/processor/peripherals and probably similar side-channel behavior. Identical also implies that the two devices have the same cryptographic implementation (in software or hardware), that is, they compute the same cipher in the exact same manner. Such open hardware/software implementations encourage security over obscurity; therefore, the identical device scenario is supported by certification standards like Common Criteria [40]. Still, when considering the degree of similarity between two separate physical entities (like the profiling and the attack device), it is plausible to question the limits of such an assumption. The natural variance of chip manufacturing processes, age differences between the profiling and the attack device and many other factors can insert cracks in the identical device assumption [73]. This problem is studied by the field of knowledge transferability and has led to techniques that alleviate it [16, 87]; yet, side-channel evaluations still face confusion when it comes to assessing the effectiveness of attacks that require two separate devices. An easy way to tackle this issue is to keep in mind that cryptanalysis tends to favor strict security bounds. That is, an evaluation should consider the worst possible scenario and assume an adversary capable of perfect transferability, which essentially is a same-device analysis that builds and tests profiles on a single device. This worst-case approach is often used to speed up evaluations since it uses a single device and avoids multiple measurement campaigns and setups. Interestingly, there exist cases where such a same-device analysis is possible, even when an open device is not available [77].

Having discussed the intricacies of open and identical devices, we proceed with profiled attacks in Algorithm 2:

(1) The first step (after divide-and-conquer partitioning of the full key) is the *leakage profiling*. In this step, the evaluator makes use of the profiling device, sets the key K to a specific value and then measures a large number of side-channel traces corresponding to that value. We denote the leakage when profiling key k as $\mathbf{L}_k^{prof}$ and we denote the measured profiling traces as the set $\mathcal{L}_k^{prof}$, for all $k \in \mathcal{K}$. In addition, we describe this profiling phase as $\mathcal{L}_k^{prof} \leftarrow \mathbf{L}_k^{prof}$, to indicate that the traces can be acquired by measuring when the cipher encrypts using key k. The process results in the following sets of traces: $[\mathcal{L}_0^{prof}, \mathcal{L}_1^{prof}, \ldots, \mathcal{L}_{|\mathcal{K}|-1}^{prof}]$. Note that sometimes it is experimentally hard or very slow to re-key the cipher, therefore the evaluator may keep a fixed key and profile a key-dependent value instead[15]. Since the evaluator possesses an open profiling device, the amount of traces at this step can be arbitrarily large. We note that certification schemes (and the metrics of Sections 2, 4, 5) typically consider the number of traces needed for *attacking* and not necessarily for *profiling*, which leads to some confusion on whether the profiling traces must be included or excluded in a certification's guidelines. This confusion originates from the early side-channel attacks like DPA and CPA, both *unprofiled*[16] attacks, therefore such a distinction between number of traces used for profiling and number of traces used for attacking was not a concern.

(2) After measuring traces for every key $k \in \mathcal{K}$, the second (and probably most crucial step) of a profiled attack is the *leakage modeling*. In this step the evaluator starts by assuming a leakage model that she believes to be a simplification or an approximation of the measured traces. There exists a plethora of options regarding leakage models: Statistical modeling options include parametric distributions [15], linear models [78], machine learning provides models based on support vector machines [38, 47] and more recently deep learning has used multi-layer perceptrons and various neural networks to the same end [7, 55, 66]. Evaluators have used multiple models with varying degrees of success, reminding us of the anecdotal statement in statistics that *"All models are wrong, but some are useful"*, and prompting us to closely examine the relationship between metrics and models. In all cases, after performing an appropriate model assumption, the evaluator will use the the measured traces $\mathcal{L}_k^{prof}$ to train models for every key candidate k, a process that parametric statistical modeling commonly refers to as *parameter estimation*. We denote this process as $model_k \leftarrow \mathcal{L}_k^{prof}$ and it results in the model list $[model_0, model_1, \ldots, model_{|\mathcal{K}|-1}]$.

(3) The third (and final) step in a profiled attack is the *leakage exploitation*, where the evaluator proceeds with the actual attack. During this step the evaluator first measures traces from the attack device, which uses the correct key k_c that she is trying to recover. We will refer to these observable as $\mathbf{L}_{k_c}^{test}$ and to the *test traces* as $\mathcal{L}_{k_c}^{test}$. We use the expression $\mathcal{L}_{k_c}^{test} \leftarrow \mathbf{L}_{k_c}^{test}$ to show that the test traces can be acquired by measuring traces when the cipher uses the correct key candidate k_c. These test traces are compared to the leakage models of all keys $[model_0, model_1, \ldots, model_{|\mathcal{K}|-1}]$, then a score is computed ($score_k$) for every model and finally the models are ranked from best to worst, based on how well they fit the test traces.

7.3 Description: Mutual Information - Part 2

Having described the workflow of profiled attacks, we return to our original goal, which is to compute the *MI* metric and in particular the quantity $Pr(\mathbf{l}|k)$ in Equation 26. The *MI* metric requires that Step 1 (leakage profiling) and Step 2 (leakage modeling) of Algorithm 2 have already been carried out. For instance, assuming a profiled

[15]In the case of AES-128, it is common to observe the leakage $\mathbf{L}^{prof}$ of the Sbox output, since learning the Sbox output is equivalent to learning the key, when the plaintext is also known. This shortcut is ignoring the direct key leakage (since the key is always the same), yet it becomes useful if re-keying AES is impossible (e.g., due to a hard-wired key) or if re-keying creates measurement bottlenecks (e.g., due to slow PUF processes).

[16]Side-channel attacks described in Algorithm 1 do not use an open device to learn the leakage behavior and they analyze directly the device-under-attack. Thus they are often called unprofiled, yet this term appeared after the spread of profiled attacks.

Algorithm 2: The workflow of a profiled side-channel attack.

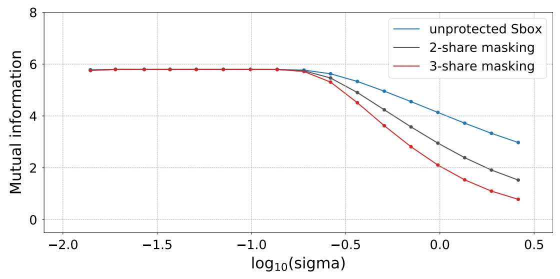
```
Input: Model score function $g(\cdot)$
Input: Key candidates $\mathcal{K}$, with set size $|\mathcal{K}|$
Data: Profiling side-channel traces $[\mathcal{L}_0^{prof}, \mathcal{L}_1^{prof}, \ldots, \mathcal{L}_{|\mathcal{K}|-1}^{prof}]$
Data: Test side-channel traces $\mathcal{L}_{k_c}^{prof}$
Output: $score_k, guess_k, rank_k$, for all $k \in \mathcal{K}$
▷ Partition the full cipher key and attack all partitions
$partitions$ = divide-and-conquer($full\ key$)
for all partitions do
    ▷ Step 1: leakage profiling
    for all $k \in \mathcal{K}$ do
        $\mathcal{L}_k^{prof} \leftarrow \mathbf{L}_k^{prof}$
    ▷ Step 2: leakage modeling
    for all $k \in \mathcal{K}$ do
        $model_k \leftarrow \mathcal{L}_k^{prof}$
    ▷ Step 3: leakage exploitation
    $\mathcal{L}_{k_c}^{test} \leftarrow \mathcal{L}_{k_c}^{test}$
    for all $k \in \mathcal{K}$ do
        $score_k = g(model_k, \mathcal{L}_{k_c}^{test})$
    ▷ Sort the scores and compute the key guesses
    $[score_i, score_j, \ldots, score_m] = \text{sort}([score_0, score_1, \ldots, score_{|\mathcal{K}|-1})$
    $[guess_1, guess_2, \ldots, guess_{|\mathcal{K}|}] = [i, j, \ldots, m]$
    ▷ Compute the rank of every key candidate using the key guesses
    for all $guess_i$, $i = 1, 2, \ldots, |\mathcal{K}|$ do
        $rank_{guess_i} = i$
```

attack on a key byte of AES-128, the evaluator can use an open device to measure traces $[\mathcal{L}_0^{prof}, \mathcal{L}_1^{prof}, \ldots, \mathcal{L}_{255}^{prof}]$. Subsequently, she can assume that a fitting model is a univariate (single-sample) normal distribution. For normal distributions, the step of leakage modeling is equivalent to estimating appropriate values for the parameters μ and σ and producing the model list $[\mathcal{N}(\mu_0, \sigma_0), \mathcal{N}(\mu_1, \sigma_1), \ldots, \mathcal{N}(\mu_{255}, \sigma_{255})]$. That is, $Pr(\mathbf{l}|k) \sim \mathcal{N}(\mu_k, \sigma_k)$. The statistical parameters μ, σ are typically estimated using the sample mean and variance (as in Section 3, Equations 6, 7), yet we note that different models may imply vastly different estimation/training procedures[17]. We now return to Equation 26 and show how to compute the *MI* metric when the leakage is modeled with univariate normal distributions, a model assumption that is commonly referred to as univariate template [15].

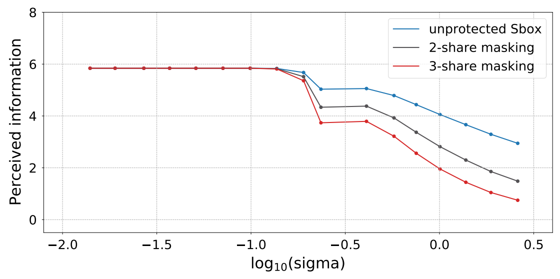
$$Pr(l|k) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{l-\mu_k}{\sigma_k}\right)^2}, \qquad MI(L;K) = \log_2|\mathcal{K}| + \sum_{k\in\mathcal{K}} \frac{1}{|\mathcal{K}|} \int_{-\infty}^{+\infty} Pr[l|k] \log_2 \frac{Pr(l|k)}{\sum\limits_{k^*\in\mathcal{K}} Pr(l|k^*)} \mathrm{d}l \qquad (27)$$

Like the correlation metric in Section 6, the *MI* metric can be used directly for deriving security bounds, i.e., even if we are not able to achieve key recovery, we can still project the security level. Equation 28 links

[17]Neural networks estimate the weights of connected nodes, support vector machines estimate hyperplanes etc.



(a) Mutual Information.



(b) Perceived information.

Fig. 7. Mutual and perceived information vs. noise σ for the unprotected Sbox (blue line), 2-share masking projection (gray line) and 3-share masking projection (red line).

MI to the number of traces required for key recovery [24] and Equation 29 does the same when the masking countermeasure [14] is present in the device. The masking countermeasure splits the key-dependent values of the cipher into d shares, forcing the attacker to recombine them. E.g., the value Y will be split to $Y_1, Y_2, \dots Y_d$ shares, leaking $\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_d$, respectively. Increasing the number d of shares results in noise amplification[18]. Equations 28, 29 offer an adequate approximation when the noise is high and assume that all intermediate values (or shares) leak the same amount of information. Such shortcut formulas can speed up an analysis, yet the evaluator must keep in mind that they provide bounds that may overestimate the attacker's capabilities. Still, we emphasize that MI can capture directly multivariate attacks, so unlike correlation-based shortcuts, these bounds can provide a more fair picture.

$$\text{number of traces} \geq \frac{H(K)}{MI(\mathbf{L};K)} \qquad (28) \qquad\qquad \text{number of traces} \geq \frac{H(Y_i)}{MI(\mathbf{L}_i;Y_i)^d} \qquad (29)$$

Figure 7a shows the effect of noise and masking on the MI metric. We evaluate the MI metric using the simulated traces of the unprotected AES Sbox. Low noise levels result in MI close to six bits, showing low security. Increasing the noise results in lower MI, showing that the device is more secure. Note that MI conveys the security of the device directly, i.e,. it does not rely on a specific adversary that produces scores and ranks. Hence, it can evaluate the security level, regardless of the adversary. The effect of masking on the device security (computed using Equation 29) is also visible in Figure 7a. As the number of shares increases, the slope of MI curve changes, showing the noise amplification.

7.4 Hypothetical Information

It is worth noting that the MI metric is inherently flexible regarding the leakage model, that is it offers a framework that can work with various types of leakage functions. Depending on our analysis' goals, it may not be necessary to estimate the $model_k$ from experimental profiling traces $\mathcal{L}_k^{prof}$. For example, if our goal is to evaluate the behavior and effectiveness of different theorized leakage models (independently of the adversary), then the evaluator can skip the leakage profiling (Step 1 of Algorithm 2). Subsequently, she can replace the leakage model in Step 2 with her theoretical choice. For instance, if she wants to examine the side-channel security when the leakage is a univariate normal distribution, she can simply replace the parameter estimation process (mean and standard deviation estimators of a traceset) with chosen parameters μ and σ. If the model assumption is any probability distribution (like $\mathcal{N}(\mu, \sigma)$ in this case), then we refer to parameters μ, σ as the *true parameters* in

[18]For more details w.r.t. masking see also [39, 60] (early masking techniques), [41, 75] (recent masking techniques) and [4, 64] (practical issues in masking).

order to distinguish them from the *estimated parameters* $\hat{\mu}, \hat{\sigma}$, which are derived from experimental traces. A version of the *MI* metric that uses the true parameters is useful when theorized leakage models are tested against countermeasures in order to identify potential weaknesses and to perform adversary-independent assessments. Masking schemes constitute a prime example for such cases. Although masking has been proven secure under certain leakage models [14], they often present vulnerabilities when the model diverges from the underlying assumption. For example standard, as well as low-entropy masking schemes [4, 5, 35] have been tested against non-linear or distance-based leakage models, in order to assess their effectiveness.

Since *MI* can be encountered with either true or estimated model parameters, later works [41] distinguish between these two versions by using the term *MI* with true parameters and the term *HI* (standing for Hypothetical Information) with estimated parameters[19]. The following equation defines *HI*, using the quantity $\hat{Pr}(\mathbf{l}|k)$ that denotes a model with estimated parameters.

$$HI(\mathbf{L};K) = -\sum_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k) + \sum_{k\in\mathcal{K}} Pr(k) \int_{\mathbf{l}\in\mathbb{R}^m} \hat{Pr}[\mathbf{l}|k] \log_2 \frac{\hat{Pr}(\mathbf{l}|k)}{\sum\limits_{k^*\in\mathcal{K}} \hat{Pr}(\mathbf{l}|k^*)} d\mathbf{l} \tag{30}$$

7.5 Perceived Information

The goal of *MI* is to provide a separation between implementation and adversary. When considering leakage models with true parameters, such separation is largely achieved, i.e., the *MI* metric can lead to attacker-independent conclusions and answer directly *"How secure is this implementation?"* Still, in the estimated parameter version of the metric, that is *HI*, such separation can easily come under scrutiny. If the evaluator's analysis is not limited to theoretical leakage models, then she immediately faces the task of capturing a large training dataset $\mathcal{L}_k^{prof}$ and subsequently choosing an adequate leakage model that reflects the device leakage. In other words, moving from *MI* to *HI* reintroduces adversary-related aspects such as the ability to capture and store a large number of traces, as well as the ability to identify and train effective models. Unfortunately, an adversary's inability to capture many training traces or a poor modelling choice on her side is *not* fully reflected in the *HI* metric. We elaborate on this by examining two cases.

Estimation errors. In the first case, assume an evaluator that has chosen an adequate leakage model $model_k^A$, i.e., this model (after training) can describe sufficiently well the actual chip leakage. Then, the evaluator trains two instances of $model_k^A$: one instance by using training dataset $\mathcal{L}_k^{prof1}$ with a large number of traces (we call this instance $model_k^{A1}$) and one instance using a training dataset $\mathcal{L}_k^{prof2}$ with a small number of traces (we call this instance $model_k^{A2}$). Such a weak leakage profiling step could occur due to, e.g., an intentional delay mechanism that limits the number of captured traces in the profiling device or simply due to the limited storage and computational power of the adversary. Because of the difference between the two profiling steps, it is very likely that the $model_k^{A1}$ will translate into an increased *HI* compared to $model_k^{A2}$ and this in turn will result in less traces needed for key recovery using the shortcut Equation 29. In other words, the weak profiling step is reflected in *HI*. However, if $\mathcal{L}_k^{prof2}$ is *too* small, then the leakage profile becomes inadequate and it may be impossible to recover the key using the trained $model_k^{A2}$. Despite this, the *HI* metric stays positive (by definition) and the shortcut Equation 29 suggests that key recovery is still possible! The underlying problem in this first case lies in the *estimation errors* that occur during leakage modelling [26] and it showcases that positive *HI* does not necessarily imply key recovery.

[19]Interestingly, the first definition of *MI* as a side-channel metric used the true statistical parameters [83].

Assumption errors. In the second case, assume an evaluator that has chosen an inadequate leakage model ($model_k^B$), i.e., her model assumption does not reflect the actual device leakage. Experimentally, such an incorrect assumption may arise in various scenarios. For instance, when leakage profiling is carried out on a different device (i.e., we are not performing a same-device analysis), we face the problem of profile transferability (Section 7.2) which may severely alter the model. Likewise, uncommon leakage behavior on the chip may not be captured by the adversary's model assumption (for instance [67] attempts to gradually adjust a linear model with quadratic terms). In such cases, even when using well-estimated model parameters (typically with a large training dataset $\mathcal{L}_k^{prof}$), the key recovery may be severely hindered if not impossible. Once again though, the *HI* will be positive, suggesting the contrary. The underlying problem in this case is the *assumption error* during leakage modelling [26] and demonstrates once again that *HI* can be misleading.

Note that the *MI* metric (using true parameters and a perfectly matching model) faces neither estimation nor assumption errors. *HI*, however, is *"the amount of information that would be leaked from an implementation of which the leakages would be exactly predicted"* [41]. The pitfall of *HI* is that it takes into account solely the leakage profiling (Step 1) and leakage modeling (Step 2) of Algorithm 2 and does not consider the test dataset during the leakage exploitation (Step 3). Rephrasing, *HI* does not test its estimated model against the actual device leakage. This gap is to be filled with yet another information-theoretic metric, known as Perceived Information (*PI*) [73].

The definition of *PI* requires the notion of cross-entropy $H_{p,q}(K|\mathbf{L})$. This quantity measures the information of the variable $(K|\mathbf{L})$, when the variable is modeled using distribution q, while the true distribution of the variable is p. In our case, the leakage profiling and modeling steps of Algorithm 2 constitute the model distribution q, denoted with probability $\hat{Pr}_{model}(k|\mathbf{l})$. The true distribution p is denoted with probability $Pr_{true}(k|\mathbf{l})$.

$$PI(\mathbf{L};K) = H(K) - H_{true,model}(K|\mathbf{L}) = \left(-\sum_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k)\right) - \left(-\sum_{k\in\mathcal{K},\mathbf{l}\in\mathcal{L}} Pr_{true}(k|\mathbf{l})\log_2 \hat{Pr}_{model}(k|\mathbf{l})\right) \quad (31)$$

Assuming uniformly random keys and using the Bayes rule, we reach the following equation:

$$PI(\mathbf{L};K) = -\sum_{k\in\mathcal{K}} Pr(k)\log_2 Pr(k) + \sum_{k\in\mathcal{K}} Pr(k) \sum_{\mathbf{l}\in\mathcal{L}} Pr_{true}(\mathbf{l}|k)\log_2 \frac{\hat{Pr}_{model}(\mathbf{l}|k)}{\sum\limits_{k^*\in\mathcal{K}} \hat{Pr}_{model}(\mathbf{l}|k^*)} \quad (32)$$

Computing the quantity $\hat{Pr}_{model}(\mathbf{l}|k)$ is identical to the computation of $\hat{Pr}(\mathbf{l}|k)$ in the *HI* metric and it naturally relates to the underlying model assumption and training dataset $\mathcal{L}_k^{prof}$. To compute the quantity $Pr_{true}(k|\mathbf{l})$, we distinguish between the following two cases: simulated leakage and real traces.

***PI* with simulated leakage.** In the theoretical case, the evaluator can replace $Pr_{true}(k|\mathbf{l})$ with any theorized probability distribution that is of interest to her analysis. Such a *PI* metric quantifies how well the adversary can model a certain distribution. That is, it answers the questions *"Can a certain training process capture the theorized leakage well enough?"* Unlike *MI* and *HI*, *PI* is not positive by definition and if the model is under-trained it can result in negative values that indicate estimation errors (but not assumption errors). This version of *PI* can also be helpful when comparing different training models that are all trying to model the same known leakage distribution. It helps us assess how many traces are needed to profile the distribution adequately. Figure 7b shows the effect of noise and masking on the PI metric. We evaluate the PI metric using the simulated traces of the unprotected AES Sbox, while adding artificial white noise. We observe that the computed *PI* value is positive and close to the *MI* of Figure 7a, indicating that the model training captures adequately the simulated leakage.

***PI* with real traces.** In the practical application of *PI* the *true* distribution is unknown and the evaluator can only sample the true leakage distribution via the test dataset $\mathcal{L}_k^{test}$. Computing the *PI* metric requires a process similar to Step 3 of Algorithm 2 that captures a test dataset for all possible key candidates k (note that typically Step 3 captures only $\mathbf{L}_{k_c}$). That is, the evaluator must observe $\mathbf{L}_k^{test}$ and produce test datasets $\mathcal{L}_k^{test}$ for all $k \in \mathcal{K}$. Every test dataset $\mathcal{L}_k^{test}$ contains a finite number of n_k test traces, i.e., $\mathcal{L}_k^{test} = \{\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_{n_k}\}$. Sampling the test datasets implies equiprobable leakages, i.e., $Pr_{true}(\mathbf{l}|k) = \frac{1}{n_k}$ and Equation 32 becomes:

$$PI(\mathbf{L}; K) = -\sum_{k \in \mathcal{K}} Pr(k) \log_2 Pr(k) + \sum_{k \in \mathcal{K}} Pr(k) \sum_{\mathbf{l} \in \mathcal{L}_k^{test}} \frac{1}{n_k} \log_2 \frac{\hat{Pr}_{model}(\mathbf{l}|k)}{\sum\limits_{k^* \in \mathcal{K}} \hat{Pr}_{model}(\mathbf{l}|k^*)} \tag{33}$$

Analogously to the theoretical case, the practical *PI* metric can answer the question *"Can a certain training process capture the actual device leakage well enough?"* and can detect the presence of both estimation and assumption errors. Negative *PI* values indicate such errors (albeit without telling them apart) and show that key recovery is not possible.

We note that the security bounds in Equations 28, 29 can be computed with either *MI* or *HI* or *PI* (if *PI* is positive). In addition, the following inequality holds between the metrics [12]:

$$PI(L; K) \leq MI(L; K) \leq HI(L; K) \tag{34}$$

7.6 Limitations

IT metrics and probabilistic generative models. A shortcoming of the *MI* and *HI* metrics is their inherent reliance on probabilistic generative leakage models. That is, to compute them, we need a modeling step (Step 2 of Algorithm 2) that trains a model capable to describe the leakage and generate new instances for it via a probability distribution. This can limit our modeling options, since it excludes discriminative models such as neural network techniques.

IT metrics in multiple dimensions. The *MI*/*HI*/*PI* metrics are naturally multivariate and can capture multiple samples of a leaky intermediate value. However, computing them can become hard as the sample dimension m increases, due to the numerical issues stemming from integration. This evaluation complexity increases even further when considering m-dimensional integration for all d shares of a masked implementation [33]. Similar issues arise when using *MI* to evaluate the shuffling countermeasure [95]. To resolve this issue, evaluators can opt for Monte-Carlo integration techniques, dimensionality reduction or approximations [24].

IT metrics and horizontal attacks. Ideally the evaluator would like to include many leaky intermediate values of the cipher in the IT metric computation. For instance, observing both the Sbox input and the Sbox output of AES can improve our chances of recovering the key. However, this will, again, increase the integral dimension and result in computational problems. To resolve this issue evaluators have developed approximations that combine the information of multiple intermediates in feasible time [36].

8 KEY ENUMERATION & RANK ESTIMATION

Section 4 and Algorithm 1 demonstrated that side-channel attacks are usually performed in a divide-and-conquer fashion, where each part of the key is attacked independently from all others. Most metrics described thus far focus on the security of these key parts and do not generalize directly to the security of the full key. E.g., in the case of AES-128, if all 16 correct key bytes are ranked first among their candidate set, the full key is trivially recovered. However, when this is not the case, the adversary has to verify different full keys through trial and error (key enumeration). The evaluator must also quantify the remaining effort for full key recovery (rank estimation). The goal of both enumeration and estimation processes is to convert the security metrics for the 16 8-bit parts of the key to a security metric for the full 128-bit key.

8.1 Description

Key enumeration. This process is preceded by an unknown key analysis that results in key scores for all partitions of the secret key, e.g., 256×16 scores for a byte-oriented attack on AES-128. Subsequently, key enumeration will utilize this information in order to list full keys for the cipher. Thus, the input to enumeration is the scores of all key candidates for all key partitions. The output of the process is binary: It will either find the correct full key or it will reach an upper limit of verifications, say n_{en}, and stop. This upper limit reflects the computational power of the adversary, typically her limit on encryptions. Hence, although the computational effort of testing, e.g., all 2^{128} AES-128 full keys is unreachable, a side-channel attack in conjunction with a key enumeration process could recover the full key within a realistic effort n_{en}. The first key enumeration algorithm was proposed by Veyrat-Charvillon et al. [93]. Other proposals followed in the works of Bogdanov et al. [10], Longo et al. [51], Poussier et al. [68] and David et al. [21], the last trading the effectiveness of the key enumeration for memory efficiency. In all cases, if the full key is attacked in p parts (each part with m key candidates) and the upper limit of the enumeration is n_{en}, then:

$$\text{security} = KeyEnumeration(\Big[[score_0^{part_1}, \ldots, score_{m-1}^{part_1}], \ldots, [score_0^{part_p}, \ldots, score_{m-1}^{part_p}]\Big]) \tag{35}$$

$$\text{security} = log_2(n_e) \text{ bits, if the full key is found in } n \leq n_{en} \text{ verifications}$$
$$\text{security} > log_2(n_e) \text{ bits, if the full key is not found after } n_{en} \text{ verifications}$$

Rank estimation. If key enumeration fails to recover the full AES-128 key after n_{en} verifications, we remain unaware of the actual security level, because it could range anywhere from n_{en} to 2^{128}. The actual level can make a significant difference, since any improvement of the experimental setup, signal processing methods and attacks could quickly alter the security picture. To overcome this limitation, the process of rank estimation was proposed. This process is typically preceded by a known key analysis that results in the *score*/*rank*/*GE* of the correct key for all partitions of the secret key (e.g., 16 scores for a byte-oriented attack on AES-128). Subsequently, rank estimation will utilize this information to quickly estimate the remaining brute force effort to recover the full key, without necessarily listing all possible keys. Veyrat-Charvillon et al. defined the first rank estimation algorithm [94]. Following, Ye et al. [100] suggested a method to find the best enumeration effort distributor across key bytes. Other key ranking methods include the knapsack-problem-based approach of Martin et al. [54], the histogram-convolution-based algorithm of Glowacz et al. [31] and the polynomial-multiplication-based algorithm of Bernstein et al. [8]. Notably, rank estimation can also work in the unknown key setting, since heuristics have been proposed to approximate the rank of an unknown secret key without performing key enumeration [3]. In the known key case, if the full key is attacked in p parts each with its own correct key candidate $k_c^i, i = 1, \ldots, p$, then:

$$\text{security} = RankEstimation([score_{k_c^1}^{part_1}, \ldots, score_{k_c^p}^{part_p}]) \tag{36}$$

To highlight the process of the rank estimation we use the histogram-convolution-based algorithm of Glowacz et al. [31]. Preceding the rank estimation, a side-channel attack has observed the side-channel leakage and obtained a list of probabilities, for each part and each key candidate. These probabilities can be used to derive the probability of any full key and can also be phrased as sum of log probabilities.

$$score_k^i = \log Pr(k^i), \text{ for } i = 1, \ldots, p \text{ and } k = 0, \ldots, m-1 \quad (37) \qquad fullkey = [k^1, k^2, \ldots, k^p] \quad (38)$$

$$Pr(fullkey) = \prod_{i=1}^{p} Pr(k^i) \iff \log Pr(fullkey) = \sum_{i=1}^{p} \log Pr(k^i) \tag{39}$$

The goal is to estimate how many full key candidates have a higher log probability than the log probability of the correct full key $[k_c^1, k_c^2, \ldots, k_c^p]$. This estimation defines how many full keys must be verified before reaching the correct one, i.e., the overall enumeration effort. In the algorithm of Glowacz et al.[31]:

(1) For every key part $i = 1, \ldots, p$, we compute the histogram of the m log probabilities (scores) using a fixed number of bins n_b. This results in p histograms $H^1, \ldots, H^p$ with the same n_b.
(2) The p histograms are convoluted iteratively, producing a histogram H corresponding to the distribution of the full key log probabilities. This provides a compressed version of the full key distribution that is subject to the histograms quantization error which depends on n_b.
(3) We compute the log probability for the correct full key $[k_c^1, k_c^2, \ldots, k_c^p]$, i.e., $lp_c = \sum_{i=1}^{p} \log Pr(k_c^i)$. Subsequently, we find in which bin of the histogram H it is positioned (we apply the function $bin(\cdot)$ on lp_c which returns the bin index of a value in a histogram). Finally, the estimated rank of the correct full key amounts to summing the number of full keys with log probability higher than lp_c.

Algorithm 3 illustrates histogram-based rank estimation. The estimated rank is then lower and upper bounded by tracking the quantization error of the histograms and the number of key parts as follows:

$$\text{rank lower bound} = \sum_{j=bin(lp_c)+p}^{p\cdot n_b-(p-1)} H(j) \ \text{ and } \ \text{rank upper bound} = \sum_{j=bin(lp_c)-p}^{p\cdot n_b-(p-1)} H(j) \tag{40}$$

Algorithm 3: Histogram-based rank estimation

```
Input: $score_k^i$, for $i = 1, \ldots, p$ and $k = 0, \ldots, m$
Output: estimated rank, security (in bits)
▷ Compute the histogram for each key part
for $i = 1$ to $p$ do
    $H^i = hist([score_0^i, \ldots, score_m^i])$
▷ Iterative convolution to produce the full key distribution
$H = H^1$
for $i = 2$ to $p$ do
    $H = conv(H, H^i)$,    i.e., the $k$th element of $H$ after the convolution is $H(k) = \sum_j H(j) \cdot H^i(k - j + 1)$
▷ Compute the sum of the log probability for the correct full key
$lp_c = \sum_{i=1}^{p} \log Pr(k_c^i)$
▷ Count the number of full keys with a higher log probability than the correct full key
estimated rank $\approx \sum_{j=bin(lp_c)}^{p\cdot n_b-(p-1)} H(j)$   and  security = $\log_2$(estimated rank)
```

A tempting shortcut is to use easy-to-compute metrics to replace the rank estimation. At first glance, the product of the individual key part ranks, or equivalently the sum of the log ranks, might seem to provide a good indication of the remaining brute force effort to recover the full key. However, summing the log ranks does not correspond to an enumeration strategy and additionally assumes an unrealistic adversary that would somehow have knowledge of the key bytes ranks. As a result, the sum of log ranks provides a pessimistic lower bound on the rank of the full key. We illustrate this with the following simple example. We consider a key consisting of

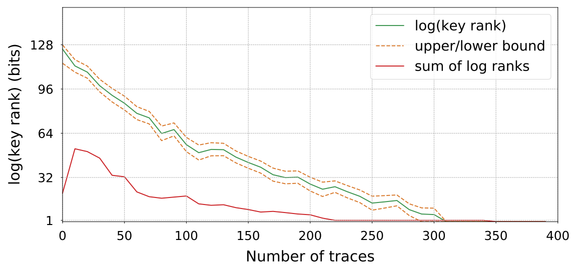


Fig. 8. Unprotected AES Sbox, simulated leakage ($\sigma = 1$).

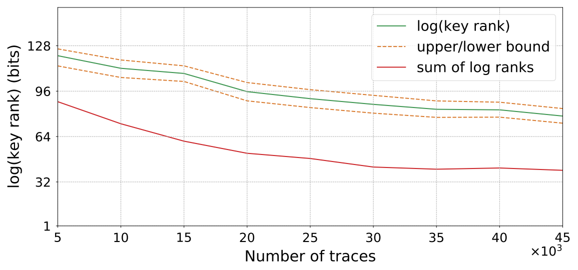


Fig. 9. Unprotected AES Sbox, device leakage.

four bits, where the first two bits and the last two bits are targeted independently. After a side-channel attack we obtain the following probability vector for each two-bit part that we denote by K^1 and K^2:

$$Pr(K^1) = [00 : 0.8,\ \mathbf{01 : 0.1},\ 10 : 0.08,\ 11 : 0.02]$$

$$Pr(K^2) = [\mathbf{00 : 0.09},\ 01 : 0.5,\ 10 : 0.01,\ 11 : 0.3]$$

The correct full key value is $[k_c^0, k_c^1] = [01, 00]$. To find the rank of the correct full key for this simple example in an optimal way, we can simply list the keys in decreasing order of likelihood until we reach the correct key. This leads to the following probability vector, where the rank of the correct full key is equal to 9 while following an optimal enumeration strategy.

$$\begin{aligned}\Pr\left[K^1, K^2\right] = [&0001 : 0.4,\ 0011 : 0.24,\ 0000 : 0.072,\ 0101 : 0.05,\ 1001 : 0.04,\\ &0111 : 0.03,\ 1011 : 0.024,\ 1101 : 0.01,\ \mathbf{0100 : 0.009},\ \ldots]\end{aligned}$$

Notice, however, that if we try to approximate the full key rank using, e.g., the product of the key part ranks, we will obtain misleading results. In this case, $rank_{k_c^1} \cdot rank_{k_c^2} = 2 \cdot 3 = 6$, i.e., a result that appears (incorrectly) better than the optimal strategy. We provide in Figures 8 and 9 results of two key rank estimations after performing template attacks. Figure 8 shows an attack exploiting simulated leakage of the unprotected AES Sbox, while Figure 9 shows the exploitation of the real leakage for the same target on the Sasebo-GII board. Note that the obtained upper/lower bounds are very tight while only requiring a small amount of effort to perform the rank estimation. Figures 8 and 9 also demonstrate the heuristic based on the sum of log ranks and we can observe that it only provides a lower bound on the rank; hence, it underestimates the resistance of the target and overestimates the attackers knowledge.

8.2 Limitations

Non-Bayesian Rank Estimation and Key Enumeration. The optimality and the soundness of both rank estimation and key enumeration techniques rely on the condition that the attack scores are probability or log probability vectors, and additionally on the fact that in classic divide-and-conquer attacks, key bytes are targeted independently. Although independent probabilities are naturally combined by multiplication, to find the probability of a full key candidate, non-probability scores such as correlation coefficients do not have a natural and mathematically correct combination function. Some heuristics can be adopted to combine the scores (e.g., by adding or multiplying them) but they lead to inaccurate full key scores and suboptimal key enumeration and rank estimation. These limitations can be potentially mitigated in practice (since we do not always have access to Bayesian scores) by using other heuristics, such as Bayesian extension defined by Veyrat-Charvillon et al. [93]. For a detailed discussion and analysis, we refer to the work of Choudary et al. [17].

9 STATISTICAL HYPOTHESIS TESTS

Several leakage evaluation procedures utilize statistical hypothesis tests or informal adaptations thereof [96, 97]. Such procedures often commence with the *null* hypothesis H_0 that the DUT is secure and the *alternate* hypothesis H_1 that the DUT has security flaws[20]. The test procedure typically entails the computation of a statistic with a known or derivable distribution under H_0. If the test statistic value crosses a threshold, then security flaws are detected (H_0 is rejected). Not crossing the threshold implies that there is no evidence of flaws (H_0 is *not* rejected). Note that not rejecting H_0 is not the same as accepting H_0; the DUT is deemed secure since the data did not detect leakage, i.e., it is innocent until proven guilty. The computed statistic, along with the binary test result (pass or fail) can be considered as a qualitative metric for SCA. These metrics can be computed using various procedures, including the t-test in the TVLA framework [32], the Pearson's χ^2 test [61, 65], the two sample Kolmogorov–Smirnov test [44, 82, 98] and the MI test [58].

9.1 Description

In this overview we limit the discussion to the t-test metric computed using the TVLA framework, whose goal is to certify a device's vulnerability (i.e., find an arbitrary leak in at least one trace sample) or to certify a device's security (i.e., find no leaks after sufficiently exhaustive testing). A common TVLA configuration is that the DUT contains a fixed key and is being repeatedly supplied a plaintext that alternates between random and fixed values. The observed leakage traces $\mathcal{L}$ are subsequently collected and split into two sets: $\mathcal{L}_r$ containing n_r random plaintext traces and $\mathcal{L}_f$ containing n_f fixed plaintext traces. Subsequently, the TVLA methodology rephrases the original dichotomy H_0 :*"the device is secure" vs.* H_1 : *"the device has security flaws"* to the following hypothesis test.

$$H_0 : \hat{\mu}_r = \hat{\mu}_f \qquad \textit{vs.} \qquad H_1 : \hat{\mu}_r \neq \hat{\mu}_f \quad \text{where} \;\; \hat{\mu}_i = \frac{1}{n_i} \sum_{l \in \mathcal{L}_i} l \quad \text{and } i \in \{r, f\} \tag{41}$$

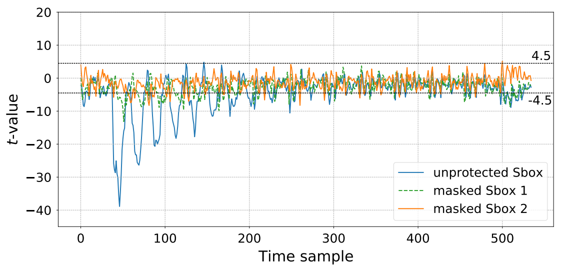
Then, the evaluator can use the two-tailed Welch's t-test to make informed judgment about the similarities of these two sets by computing the test statistic t and the degrees of freedom df in Equation 42.

$$t = \frac{\hat{\mu}_r - \hat{\mu}_f}{\sqrt{\dfrac{\hat{\sigma}_r^2}{n_r} + \dfrac{\hat{\sigma}_f^2}{n_f}}} \qquad df = \frac{\left(\dfrac{\hat{\sigma}_r^2}{n_r} + \dfrac{\hat{\sigma}_f^2}{n_f}\right)^2}{\left(\dfrac{\hat{\sigma}_r^4}{n_r^2(n_r - 1)}\right) + \left(\dfrac{\hat{\sigma}_f^4}{n_f^2(n_f - 1)}\right)} \qquad \hat{\sigma}_i^2 = \frac{1}{n_i - 1} \sum_{l \in \mathcal{L}_i} (l - \hat{\mu}_i)^2 \tag{42}$$

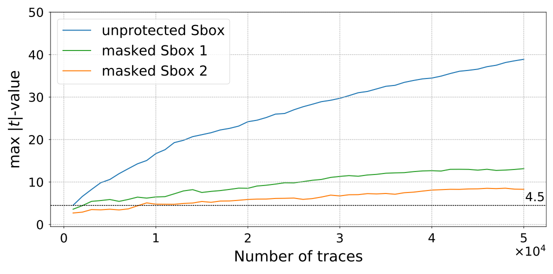
The test statistic metric t is itself an instance of a random variable T that follows a Student t-distribution, thus conclusions drawn from statistical tests are subject to error. The two-tailed test will reject H_0 when $|t| > th$, where the specified threshold th is set in a manner that balances the test errors. In particular, the decision to reject the null hypothesis H_0 when it is, in fact, true (i.e., declare the device as vulnerable, even though it is secure) is called a Type I error (false positive). A statistical test seeks to control this probability at a *significance level* α and TVLA suggests setting $\alpha = 10^{-5}$ to ensure this costly error is avoided.

$$\alpha = Pr(\textit{"detect flaws"} | \textit{"device is secure"}) = P(\text{reject } H_0 | H_0 \text{ holds}) = P(t < -th) + P(t > th) = 2P(t > th) \tag{43}$$

[20]The structure of most statistical tests imposes a logical asymmetry in the decision process by putting emphasis on the null hypothesis and considering it to hold unless proven otherwise. However, the choice of null and alternative can be a matter of perspective. Device manufacturers possibly prefer the null hypothesis *"the device is secure"*, since it must be invalidated before they have to patch/alter their product. On the contrary, a public organization tasked with certifying secure products for citizens may prefer the null hypothesis *"the device has security flaws"*. Such a hypothesis must be rejected before a product is deployed.



(a) t-value



(b) Maximum absolute t-value

Fig. 10. The t-value per trace sample and the progression of the maximum absolute t-value with the number of traces.

Setting $2P(t > th) = 10^{-5}$ and inverting the Student cdf (assuming $df > 1000$) results in the well-known $th = 4.5$ [19, 80]. Continuing, the failure to reject H_0 when it is, in fact, false (i.e., declare the device as secure, even though it is not) is called a Type II error (false negative), commonly denoted as β and it can also be set to a specified value via consensus. The TVLA process sets only the parameter α and conducts independent tests on every sample point of the measured traces. Figure 10 demonstrates the behavior of the unprotected and masked implementations. As expected, the unprotected implementation produces t-values that reject H_0 and detect flaws. Unexpectedly, security flaws are also detected in both masked implementations, albeit under the current TVLA configuration (random vs. fixed plaintext) it is unclear which intermediate values leak. We note that numerous masking schemes face practical issues and imperfections once deployed on a real-world device [4, 64].

Figure 10 shows a straightforward way to utilize the t-value as a metric and decide on the security level of the device. The metric can also be used in a heuristic manner: High t-values (and respectively low p-values[21]) indicate leaky time samples in a trace, thus the metric can provide guidance for point-of-interest selection and more detailed leakage profiling. Regardless of the usage of the metric, the evaluator should keep in mind that the metric value and the statistical significance of the results usually says more about the parameters of the experiment rather than the experimental results [63]. Equation 44 demonstrates the tradeoff between these test parameters $\alpha, \beta, n_r = n_f = n, \hat{\sigma}_r^2, \hat{\sigma}_f^2$ and δ [81, 96]. The parameter δ is the effect size, i.e., the difference between means that the test can detect reliably. The quantities $z_{\alpha/2}, z_\beta$ are the respective quantiles of the standard normal distribution[22]. Any use of the t-value as a metric implicitly utilizes all the involved test parameters and they must be set in a meaningful way in advance by the evaluator.

$$n = 2\frac{(z_{\alpha/2} + z_\beta)^2(\hat{\sigma}_r^2 + \hat{\sigma}_r^2)}{\delta^2} \tag{44}$$

Note that the described random vs. fixed test targets all intermediate values and transitions and is referred to as *nonspecific*. Even though any intermediate value (or combination) is covered by it, a negative result cannot be concluded from a single test, due to its dependency of the selected fixed plaintext. It is therefore recommended to repeat the test with at least a few different inputs to avoid a false conclusion on resistance of the device[23]. The nonspecific test can also be performed using a custom plaintext that leads to a certain intermediate value and then is referred to as the *semi-fixed vs. random* test [6]; it is often better suited for testing devices where the precise start and end of the AES operations cannot be easily determined. The test can also be adapted to detect

[21]The probability to observe a statistic as extreme as t.

[22]$Pr(Z > z_\alpha) = \alpha$ and $Z \sim \mathcal{N}(0, 1)$

[23]Good practice also recommends that the random and fixed plaintext is randomly interspersed and the internal device states is reset between subsequent measurements

higher-order leakage, e.g., to assess a masked implementation by preprocessing and then testing higher-order or mixed moments [80]. Finally, the TVLA methodology describes *specific* tests that target a particular intermediate value or transitions in the implementation (e.g., Sbox output or Hamming distance in a round register). Such a test requires knowing the device key and carefully choosing the device inputs. If a nonspecific t-test reports a detectable leakage, the specific one results in the same conclusion, but with a higher confidence [79].

9.2 Limitations

Contrary to the classical approach of attacking a device under test and linking the attack effectiveness to a metric, statistical test metrics and methodologies decouple the detection of leakage from its exploitation. The main advantages of this decoupling is increased efficiency (w.r.t. time and data complexity) and the ability to be applied with fairly little implementation knowledge. These advantages are due to two factors: a reduction of the number of classes for which the leakage has to be estimated to only two classes (e.g., random vs. fixed plaintext), and a simple statistical approach based on the estimation and comparison of statistical moments. The main drawback of such simplifications is a discrepancy between the test results (which decide about $H_0 : \hat{\mu}_r = \hat{\mu}_f$ vs. $H_1 : \hat{\mu}_r \neq \hat{\mu}_f$) and the actual device security (which must decide about H_0 :“device is secure” vs. H_1 :“flaws are detected”) [27]. This discrepancy takes several forms.

Attack effort. Detecting leakage in certain intermediate values does not automatically imply that the intermediates are exploitable in a simple divide and conquer side-channel attack. For instance, recovering values in certain parts of the cipher may require a large brute-force effort when guessing key candidates and this overhead is not captured by the t-test metric. For example, a CPA attack on the AES Sbox output requires 2^8 key guesses, while a similar attack on the MixColumns output requires 2^{32} key guesses. While both attacks are feasible, the increased brute-force effort is not reflected in the t-test metric.

Plaintext leakage detection. Naturally, the random vs. fixed t-test will detect leakage when the device processes the plaintext. The plaintext is independent of the key thus any plaintext-related spikes should be neglected. However, consider an example scenario where the initial plaintext is being transferred to memory at the very same time that a well-protected AES is in the fifth round of encryption (assume that the device is capable of some concurrency and the plaintext transfer can happen in parallel to its encryption). Any experiment aims to capture the full AES and thus it will unintentionally capture the concurrent plaintext transfer as well. The test is unable to distinguish between the parallel operations and it will raise a false alarm due to the unprotected (yet safe) plaintext transfer.

Statistical vs. practical metric significance. As stated, the TVLA result (pass/fail) can only conclude whether a difference of means is significant or not. Even if the leaky intermediate is easy to attack (e.g., AES Sbox output), the statistically significant difference still does not guarantee that a divide-and-conquer attack is able to recover the key. In Equation 44 we observe that the difference of means δ that can be detected is inversely proportional to the number of traces n used in the test. That is, increasing n will produce a test that can spot gradually smaller δ. This seemingly good property has a downside: The fact that a certain δ is detected does not imply such δ is large enough to be utilized by a side-channel distinguisher. Thus, using the t-statistic to decide on security (pass/fail) or to select leaky samples when δ is very small (and typically n is very high) can be statistically significant when, in fact, it is inconsequential.

Detection based on moments. Equation 42 shows that the t-test metric requires the computation of mean and variance. In general, moment-based approaches for detecting or exploiting flaws are very common in the literature (e.g., in higher-order DPAs using a combination function [62, 70]), yet may become suboptimal compared to using the full leakage distribution or an approximation thereof [86]. This risk is particularly difficult to anticipate as the number of shares and security order of a masked implementation increases [85]. Mitigating this drawback, while avoiding quantitative information theoretic analysis of leakages [83], requires coupling Welch's t-test with the χ^2-test [61].

Multiple comparisons. By construction, repeating any t-test multiple times will always result on false positives (controlled by the type I error α). Consider the following scenario. We apply (as in Figure 10a) the t-test on every sample of traces and such traces possess a very large number of samples. This implies a large number of test repetitions and thus a high probability of false positives [22]. The underlying issue is that TVLA assumes independence between samples and the threshold $th = 4.5$ reflects that. To counteract this problem of multiple comparisons the evaluator can adjust the threshold using the Bonferroni [25] or Sidak [92] correction formulas.

10 CONCLUSIONS

In this work, we critically analyzed the main metrics used in the field of side-channel analysis. They range from simple attack-based sub-key metrics to full key recovery metrics. For each of them, we provide a self-contained definition and its interpretation in terms of security, along with the discussion of its limitations. We showcase examples computed on both simulated and real side-channel traces and we make the software implementing all the presented metrics available to the community as open source. Since the literature about the topic is extremely fragmented, our paper follows the style of a cheat sheet, collecting in a single place all the information that designers and analysts need to assess and evaluate the side channel resistance of a device. Thanks to this format, our work represents a solid base for future analysis, including the promising direction of investigation of machine and deep learning related metrics, as side-channel analysis keeps evolving [57, 99, 101].

ACKNOWLEDGMENTS

This work is partially supported by the European Union Horizon 2020 Program under the CERBERO project (grant agreement number 732105) and EVEREST project (grant agreement No 957269), and by the Swiss National Science Foundation (grant No. 182428).

REFERENCES

[1] [n.d.]. ISO/IEC 17825:2016 [ISO/IEC 17825:2016] Information technology - Security techniques - Testing methods for the mitigation of non-invasive attack classes against cryptographic modules.

[2] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. [n.d.]. Template Attacks in Principal Subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, Louis Goubin and Mitsuru Matsui (Eds.).

[3] Melissa Azouaoui, Romain Poussier, François-Xavier Standaert, and Vincent Verneuil. 2019. Key Enumeration from the Adversarial Viewpoint. In *CARDIS 2019*, Sonia Belaïd and Tim Güneysu (Eds.). Springer, 252–267.

[4] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. 2014. On the Cost of Lazy Engineering for Masked Software Implementations. In *Smart Card Research and Advanced Applications (CARDIS)*. Paris, France, 64–81.

[5] Gilles Barthe, Marc Gourjon, Benjamin Grégoire, Maximilian Orlt, Clara Paglialonga, and Lars Porth. 2021. Masking in Fine-Grained Leakage Models: Construction, Implementation and Verification. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2021), 189–228.

[6] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab. 2013. Test Vector Leakage Assessment (TVLA) Methodology in Practice. In *International Cryptographic Module Conference*. Gaithersburg area, MD, USA, 1–13.

[7] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 2020. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.* (2020), 163–188.

[8] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. 2015. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptol. ePrint Arch.* (2015), 221.

[9] Eli Biham and Adi Shamir. 1991. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology* (1991), 3–72.

[10] Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. 2015. Fast and Memory-Efficient Key Recovery in Side-Channel Attacks. In *SAC 2015*, Orr Dunkelman and Liam Keliher (Eds.).

[11] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*. 16–29.

[12] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. 2019. Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I (Lecture Notes in Computer Science)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 713–737.

[13] David Canright and Lejla Batina. 2008. A Very Compact "Perfectly Masked" S-Box for AES (corrected). Cryptology ePrint Archive, Report 2009/011.

[14] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. [n.d.]. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO '99*, Michael J. Wiener (Ed.).

[15] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. [n.d.]. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar (Eds.).

[16] Marios O. Choudary and Markus G. Kuhn. 2018. Efficient, Portable Template Attacks. *IEEE Trans. Information Forensics and Security* (2018), 490–501.

[17] Marios O. Choudary, Romain Poussier, and François-Xavier Standaert. [n.d.]. Score-Based vs. Probability-Based Enumeration - A Cautionary Note. In *INDOCRYPT 2016*, Orr Dunkelman and Somitra Kumar Sanadhya (Eds.).

[18] Omar Choudary and Markus G. Kuhn. [n.d.]. Efficient Template Attacks. In *CARDIS 2013*, Aurélien Francillon and Pankaj Rohatgi (Eds.).

[19] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. 2016. Masking AES with $d+1$ Shares in Hardware. In *Conference on Cryptographic Hardware and Embedded Systems (CHES)*. Santa Barbara, CA, USA, 1–21.

[20] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA.

[21] Liron David and Avishai Wool. 2017. A Bounded-Space Near-Optimal Key Enumeration Algorithm for Multi-subkey Side-Channel Attacks. In *CT-RSA 2017*, Helena Handschuh (Ed.).

[22] A. Adam Ding, Liwei Zhang, Francois Durvaux, Francois-Xavier Standaert, and Yunsi Fei. 2017. Towards Sound and Optimal Leakage Detection Procedure. In *Smart Card Research and Advanced Applications (CARDIS)*. Lugano, Switzerland, 105–22.

[23] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. 2011. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering* (2011), 123–144.

[24] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. 2019. Making Masking Security Proofs Concrete (Or How to Evaluate the Security of Any Leaking Device), Extended Version. *J. Cryptology* (2019), 1263–1297.

[25] Olive Jean Dunn. 1961. Multiple Comparisons Among Means. *J. Amer. Statist. Assoc.* (1961), 52–64.

[26] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. 2014. How to Certify the Leakage of a Chip?. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings (Lecture Notes in Computer Science)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, 459–476.

[27] François Durvaux and François-Xavier Standaert. 2016. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In *Advances in Cryptology - EUROCRYPT*. Vienna, Austria, 240–62.

[28] B. Efron and R. Tibshirani. 1986. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statist. Sci.* (1986), 54–75.

[29] Yunsi Fei, Qiasi Luo, and A. Adam Ding. 2012. A Statistical Model for DPA with Novel Algorithmic Confusion Analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings (Lecture Notes in Computer Science)*, Emmanuel Prouff and Patrick Schaumont (Eds.). Springer, 233–250.

[30] Research Center for Information Security. 2009. Sasebo-GII Quick Start Guide. http://satoh.cs.uec.ac.jp/SASEBO/en/board/sasebo-g2.html.

[31] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. 2015. 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 (Lecture Notes in Computer Science)*, Gregor Leander (Ed.). Springer, 117–129.

[32] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. 2011. A Testing Methodlogy for Side-Channel Resistance Validation. In *NIST Non-Invasive Attack Testing Workshop*. Nara, Japan, 1–15.

[33] Vincent Grosso and François-Xavier Standaert. 2018. Masking Proofs Are Tight and How to Exploit it in Security Evaluations. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II (Lecture Notes in Computer Science)*, Jesper Buus Nielsen and

Vincent Rijmen (Eds.). Springer, 385–412.
[34] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. 2014. Masking vs. multiparty computation: how large is the gap for AES? *J. Cryptographic Engineering* (2014), 47–57.
[35] Vincent Grosso, François-Xavier Standaert, and Emmanuel Prouff. [n.d.]. Low Entropy Masking Schemes, Revisited. In *CARDIS 2013*, Aurélien Francillon and Pankaj Rohatgi (Eds.).
[36] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. 2020. Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2020), 209–238.
[37] M. Hellman. 1980. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory* (1980), 401–406.
[38] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. 2011. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.* (2011), 293–302.
[39] Yuval Ishai, Amit Sahai, and David Wagner. 2003. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings.* 463–481.
[40] Information technology — Security techniques Evaluation criteria for IT security Part 1: Introduction ISO/IEC 15408-1:2009 and general model. [n.d.].
[41] Anthony Journault and François-Xavier Standaert. 2017. Very High Order Masking: Efficient Implementation and Security Evaluation. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings (Lecture Notes in Computer Science)*, Wieland Fischer and Naofumi Homma (Eds.). Springer, 623–643.
[42] Lars R. Knudsen. 1999. Contemporary Block Ciphers. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998.* Springer-Verlag, Berlin, Heidelberg, 105–126.
[43] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings.* 388–397.
[44] Andrei N. Kolmogorov. 1933. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell'Instituto Italiano degli Attuari* (1933), 83–91.
[45] Boris Köpf and David Basin. 2007. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '07)*. Association for Computing Machinery, New York, NY, USA, 286–296.
[46] R.J. Larsen and M.L. Marx. 2006. *An Introduction to Mathematical Statistics and Its Applications.* Pearson Prentice Hall.
[47] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. 2014. Power Analysis Attack: An Approach Based on Machine Learning. *Int. J. Appl. Cryptol.* (2014), 97–115.
[48] Oleksiy Lisovets, David Knichel, Thorben Moos, and Amir Moradi. 2021. Let's Take it Offline: Boosting Brute-Force Attacks on iPhone's User Authentication through SCA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2021), 496–519.
[49] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. 2014. How to Estimate the Success Rate of Higher-Order Side-Channel Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings (Lecture Notes in Computer Science)*, Lejla Batina and Matthew Robshaw (Eds.). Springer, 35–54.
[50] Victor Lomné, Emmanuel Prouff, and Thomas Roche. 2013. Behind the Scene of Side Channel Attacks. In *Advances in Cryptology - ASIACRYPT 2013*, Kazue Sako and Palash Sarkar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 506–525.
[51] Jake Longo, Daniel P. Martin, Luke Mather, Elisabeth Oswald, Benjamin Sach, and Martijn Stam. 2016. How low can you go? Using side-channel data to enhance brute-force key recovery. *IACR Cryptol. ePrint Arch.* (2016), 609.
[52] F. Mace, F. x. St, I. Hassoune, J. d. Legat, and J. j. Quisquater. 2004. A Dynamic Current Mode Logic to Counteract Power Analysis Attacks. In *In The Proceedings of DCIS 2004.* 186–191.
[53] Stefan Mangard. 2004. Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness. In *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings (Lecture Notes in Computer Science)*, Tatsuaki Okamoto (Ed.). Springer, 222–235.
[54] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. 2015. Counting Keys in Parallel After a Side Channel Attack. In *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 (Lecture Notes in Computer Science)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, 313–337.
[55] Zdenek Martinasek, Jan Hajny, and Lukas Malina. 2014. Optimization of Power Analysis Using Neural Network. In *Smart Card Research and Advanced Applications*, Aurélien Francillon and Pankaj Rohatgi (Eds.). Springer International Publishing, Cham, 94–107.
[56] J. L. Massey. 1994. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory.*
[57] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. 2020. A Comprehensive Study of Deep Learning for Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2020), 348–375.

[58] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. 2013. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In *Advances in Cryptology - ASIACRYPT 2013*, Kazue Sako and Palash Sarkar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 486–505.

[59] Mitsuru Matsui and Atsuhiro Yamagishi. 1993. A New Method for Known Plaintext Attack of FEAL Cipher. In *Advances in Cryptology — EUROCRYPT' 92*, Rainer A. Rueppel (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 81–91.

[60] Thomas S. Messerges. 2001. Securing the AES Finalists Against Power Analysis Attacks. In *Fast Software Encryption*, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 150–164.

[61] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. 2018. Leakage Detection with the χ^2-Test. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 209–37.

[62] Amir Moradi and François-Xavier Standaert. 2016. Moments-Correlating DPA. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*. Vienna, Austria, 5–15.

[63] Kevin R Murphy, Brett Myors, and Allen Wolach. 2014. *Statistical power analysis: A simple and general model for traditional and modern hypothesis tests*. Routledge.

[64] Kostas Papagiannopoulos and Nikita Veshchikov. 2017. Mind the Gap: Towards Secure 1st-Order Masking in Software. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10348)*, Sylvain Guilley (Ed.). Springer, 282–297. https://doi.org/10.1007/978-3-319-64647-3_17

[65] Karl Pearson. 1992. *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*. Springer New York, New York, NY, USA, 11–28.

[66] Guilherme Perin, Baris Ege, and Lukasz Chmielewski. 2019. Neural Network Model Assessment for Side-Channel Analysis. *IACR Cryptol. ePrint Arch.* (2019), 722.

[67] Romain Poussier, Qian Guo, François-Xavier Standaert, Claude Carlet, and Sylvain Guilley. 2017. Connecting and Improving Direct Sum Masking and Inner Product Masking. In *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers (Lecture Notes in Computer Science)*, Thomas Eisenbarth and Yannick Teglia (Eds.). Springer, 123–141.

[68] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. 2016. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings (Lecture Notes in Computer Science)*, Benedikt Gierlichs and Axel Y. Poschmann (Eds.). Springer, 61–81.

[69] Santos Merino Del Pozo and François-Xavier Standaert. 2015. Blind Source Separation from Single Measurements Using Singular Spectrum Analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings (Lecture Notes in Computer Science)*, Tim Güneysu and Helena Handschuh (Eds.). Springer, 42–59.

[70] Emmanuel Prouff, Matthieu Rivain, and Régis Bévan. 2009. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.* (2009), 799–811.

[71] Francesco Regazzoni. 2017. Physical Attacks and Beyond. In *Selected Areas in Cryptography – SAC 2016*, Roberto Avanzi and Howard Heys (Eds.). Springer International Publishing, Cham, 3–13.

[72] Francesco Regazzoni, Wang Yi, and François-Xavier Standaert. 2011. FPGA Implementations of the AES Masked Against Power Analysis Attacks. In *Proceedings of 2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*. Darmstadt, Germany, 1–11.

[73] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. 2011. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings (Lecture Notes in Computer Science)*, Kenneth G. Paterson (Ed.). Springer, 109–128.

[74] Matthieu Rivain. 2009. On the Exact Success Rate of Side Channel Analysis in the Gaussian Model. In *Selected Areas in Cryptography*, Roberto Maria Avanzi, Liam Keliher, and Francesco Sica (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 165–183.

[75] Matthieu Rivain and Emmanuel Prouff. 2010. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, Stefan Mangard and François-Xavier Standaert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 413–427.

[76] Dorian Ros, Ognjen Glamočanin, and Mirjana Stojilović. 2021. MetriSCA: A Library of Metrics for Side-Channel Analysis (Ver 1.0). metrisca.epfl.ch

[77] Niels Samwel, Lejla Batina, Guido Bertoni, Joan Daemen, and Ruggero Susella. 2018. Breaking Ed25519 in WolfSSL. In *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings (Lecture Notes in Computer Science)*, Nigel P. Smart (Ed.). Springer, 1–20.

[78] Werner Schindler, Kerstin Lemke, and Christof Paar. 2005. A Stochastic Model for Differential Side Channel Cryptanalysis. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings (Lecture Notes in Computer Science)*, Josyula R. Rao and Berk Sunar (Eds.). Springer, 30–46.

[79] Tobias Schneider and Amir Moradi. 2015. Leakage Assessment Methodology: A Clear Roadmap for Side-Channel Evaluations. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Saint-Malo, France, 495–513.

[80] Tobias Schneider and Amir Moradi. 2016. Leakage Assessment Methodology. *Journal of Cryptographic Engineering* (2016), 85–99.

[81] Hansheng Wang Shein-Chung Chow, Jun Shao. [n.d.]. A Note on Sample Size Calculation for Mean Comparisons based on Noncentral t-Statistics.

[82] Nikolai Smirnov. 1948. Table for Estimating the Goodness of Fit of Empirical Distributions. *The Annals of Mathematical Statistics* (1948), 279–81.

[83] François-Xavier Standaert, Tal Malkin, and Moti Yung. 2009. 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 (Lecture Notes in Computer Science)*, Antoine Joux (Ed.). Springer, 443–461.

[84] F.-X. Standaert. [n.d.]. High (Physical) Security & Lightweight (Symmetric) Cryptography, invited talk, HighLight: High-Security Lightweight Cryptography, Leiden, The Netherlands, November 2016, and LightCrypto Workshop, Cannes, France, November 2016.

[85] François-Xavier Standaert. 2018. How (Not) to Use Welsch's T-Test in Side-Channel Security Evaluations. In *Smart Card Research and Advanced Applications (CARDIS)*. Montpellier, France, 65–79.

[86] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. 2010. The World Is Not Enough: Another Look on Second-Order DPA. In *Advances in Cryptology - ASIACRYPT*. Singapore, 112–29.

[87] Dhruv Thapar, Manaar Alam, and Debdeep Mukhopadhyay. 2020. TranSCA: Cross-Family Profiled Side-Channel Attacks using Transfer Learning on Deep Neural Networks. *IACR Cryptol. ePrint Arch.* (2020), 1258.

[88] K. Tiri, M. Akmal, and I. Verbauwhede. 2002. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European Solid-State Circuits Conference*. 403–406.

[89] Thomas Unterluggauer, Thomas Korak, Stefan Mangard, Robert Schilling, Luca Benini, Frank K. Gürkaynak, and Michael Muehlberghuber. [n.d.]. Leakage Bounds for Gaussian Side Channels. In *CARDIS 2017*, Thomas Eisenbarth and Yannick Teglia (Eds.).

[90] http://reassure.eu/wp-content/uploads/2018/09/REASSURE_D23.pdf. [n.d.]. REASSURE, Deliverable D2.1, Shortcut Formulas for Side Channel Evaluation.

[91] https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-0.pdf. [n.d.]. Application of Attack Potential to Smartcards and Similar Devices.

[92] Zbyněk Šidák. 1967. Rectangular Confidence Regions for the Means of Multivariate Normal Distributions. *J. Amer. Statist. Assoc.* (1967), 626–33.

[93] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. 2012. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers (Lecture Notes in Computer Science)*, Lars R. Knudsen and Huapeng Wu (Eds.). Springer, 390–406.

[94] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. [n.d.]. Security Evaluations beyond Computing Power. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings (Lecture Notes in Computer Science)*, Thomas Johansson and Phong Q. Nguyen (Eds.).

[95] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. 2012. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings (Lecture Notes in Computer Science)*, Xiaoyun Wang and Kazue Sako (Eds.). Springer, 740–757.

[96] Carolyn Whitnall and Elisabeth Oswald. 2019. A Cautionary Note Regarding the Usage of Leakage Detection Tests in Security Evaluation. Cryptology ePrint Archive, Report 2019/703. , 44 pages.

[97] Carolyn Whitnall and Elisabeth Oswald. 2019. A Critical Analysis of ISO 17825 ('Testing Methods for the Mitigation of Non-invasive Attack Classes Against Cryptographic Modules'). In *Advances in Cryptology – ASIACRYPT 2019*, Steven D. Galbraith and Shiho Moriai (Eds.). Cham, 256–284.

[98] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. 2011. An Exploration of the Kolmogorov-Smirnov Test as a Competitor to Mutual Information Analysis. In *Smart Card Research and Advanced Applications (CARDIS)*. Leuven, Belgium, 234–51.

[99] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2021. On the Evaluation of Deep Learning-based Side-channel Analysis. *IACR Cryptol. ePrint Arch.* (2021), 952.

[100] Xin Ye, Thomas Eisenbarth, and William Martin. [n.d.]. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Information Enables Key Recovery, Marc Joye and Amir Moradi (Eds.).

[101] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. 2020. A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2020), 73–96.

A SIDE-CHANNEL METRICS CHEAT SHEET

Table 2. Side-channel metrics cheat sheet: summary table.

Metric	***Definition***	***Prerequisites***
Number of traces	Number of traces required for a metric to reach a specified value (e.g., number of traces s.t. success rate reaches 95%)	Chosen metric estimation
Signal to Noise Ratio	$SNR = \frac{Var(L_d)}{Var(L_n)} = \frac{E(L_d^2)-E(L_d)^2}{\sigma^2}$	Trace set with known key or intermediate
Score	Attack distinguisher (e.g., Pearson's correlation coefficient for CPA)	Attack results
Rank and Guessing entropy	$rank_{guess}$ = position of *guess*'s score in the sorted vector of scores $GE = \frac{1}{p}\sum_{i=1}^{p} \log_2(rank^i_{k_c})$	Attack results and correct key
Success Rate	$SR^i = \begin{cases} 1, \text{ if } rank_{k_c} = 1 \\ 0, \text{ otherwise} \end{cases}$ and $SR = \frac{1}{p}\sum_{i=1}^{p} SR^i$	Attack results and correct key
Success Rate of order o	$SR^i_o = \begin{cases} 1, \text{ if } rank_{k_c} \leq o \\ 0, \text{ otherwise} \end{cases}$ and $SR_o = \frac{1}{p}\sum_{i=1}^{p} SR^i_o$	Attack results and correct key
Correlation	$\rho_{k_c} = \frac{\sum_{i=1}^{n}\left(l_i - \frac{1}{n}\sum_{i=1}^{n} l_i\right)\left(g(f(x_i,k_c)) - \frac{1}{n}\sum_{i=1}^{n} g(f(x_i,k_c))\right)}{\sqrt{\sum_{i=1}^{n}\left(l_i - \frac{1}{n}\sum_{i=1}^{n} l_i\right)^2}\sqrt{\left(g(f(x_i,k_c)) - \frac{1}{n}\sum_{i=1}^{n} g(f(x_i,k_c))\right)^2}}$	Correct key
Mutual Information	$MI(\mathbf{L};K) = H(K) + \sum_{k\in\mathcal{K}} Pr(k) \int_{\mathbf{l}\in\mathbb{R}^m} Pr(\mathbf{l}\|k) \log_2 Pr(k\|\mathbf{l})\mathrm{d}\mathbf{l}$	Knowledge of the true leakage distribution
Hypothetical Information	$HI(\mathbf{L};K) = H(K) + \sum_{k\in\mathcal{K}} Pr(k) \int_{\mathbf{l}\in\mathbb{R}^m} \hat{Pr}[\mathbf{l}\|k] \log_2 \frac{\hat{Pr}(\mathbf{l}\|k)}{\sum_{k^*\in\mathcal{K}} \hat{Pr}(\mathbf{l}\|k^*)}\mathrm{d}\mathbf{l}$	Hypothetical leakage distribution
Perceived Information	$PI(\mathbf{L};K) = H(K) + \sum_{k\in\mathcal{K}} Pr(k) \sum_{\mathbf{l}\in\mathcal{L}} Pr_{true}(\mathbf{l}\|k) \log_2 \frac{\hat{Pr}_{model}(\mathbf{l}\|k)}{\sum_{k^*\in\mathcal{K}} \hat{Pr}_{model}(\mathbf{l}\|k^*)}$	Profiling set and probability-based scores
Full key rank and Guessing entropy	Number of full key candidates to enumerate before reaching the correct full key	Attack results and key ranking algorithm with correct key or enumeration algorithm
T-statistic	$t = \frac{\hat{\mu}_r - \hat{\mu}_f}{\sqrt{\frac{\hat{\sigma}_r^2}{n_r} + \frac{\hat{\sigma}_f^2}{n_f}}}$ with $\hat{\mu}_i = \frac{1}{n_i}\sum_{l\in\mathcal{L}_i} l$; $\hat{\sigma}_i^2 = \frac{1}{n_i - 1}\sum_{l\in\mathcal{L}_i}(l - \hat{\mu}_i)^2$ for $i \in \{r, f\}$	Two trace sets corresponding to the fixed vs. random or fixed vs. fixed TVLA

B METRISCA LIBRARY DESCRIPTION

B.1 Introduction

MetriSCA is a software library written in C and C++, which provides a collection of leakage models and the metrics presented in this paper. The library also includes an application that lets the user process the collected data traces, to help in the device security evaluation process. The code is available for download at `metrisca.epfl.ch` (link).

B.2 Architecture

The block diagram of MetriSCA is depicted in Figure 11. The library contains five main modules: file loader, leakage model, distinguisher, metric, and profiler. These modules have multiple built-in implementations, which correspond to different algorithms that can be selected by the user.

The power side-channel traces, the plaintexts, and the encryption keys can be loaded from any file format provided that a matching file loader implementation is available. The application then converts the input data into a binary format that is optimized for fast data loading. The latter format can be loaded natively by the application, making it convenient to share datasets across platforms. The performance of custom loaders is not particularly important as the loaders are only used once, to create the optimized file. Currently, the library provides the support for the binary format only; adding support for other formats is in progress.

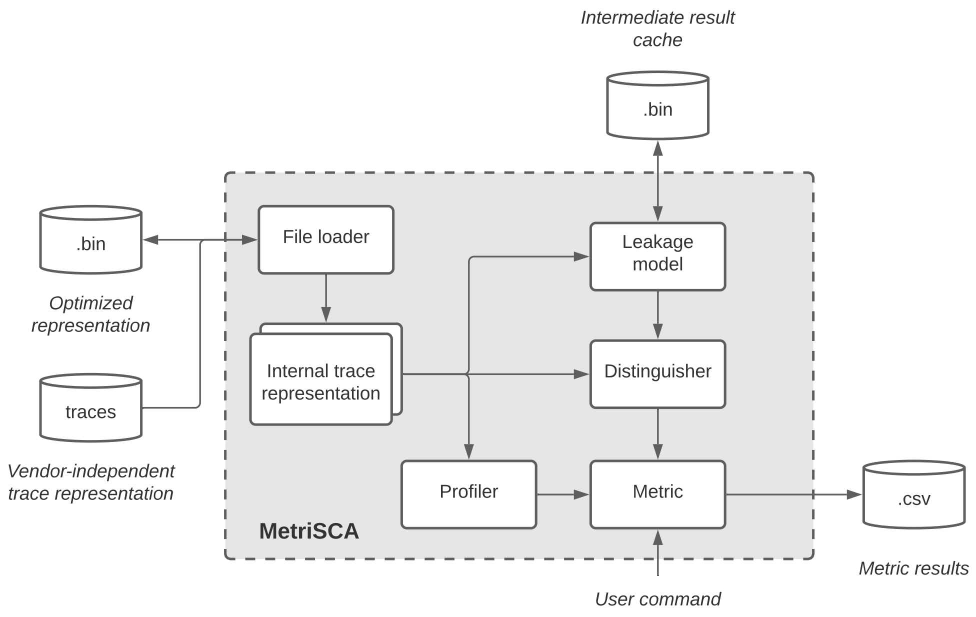


Fig. 11. Block diagram of the MetriSCA library.

B.3 Extensibility

The library is built with extensibility in mind, so that new algorithms and file formats can conveniently be added and integrated by the users. This is achieved by internally representing each module in Figure 11 as a function pointer. Hence, not only it is easy to extend the functionality of the library, the number of lines of code required for these extensions is greatly reduced.

B.4 Usage

The library is accompanied by a command-line application, serving as the user interface. It implements commands to load and analyse trace files. It also integrates documentation about the various commands and related arguments. The metric outputs are saved in comma-separated value (CSV) format. All the plots in this manuscript are generated using the MetriSCA application.

Additionally, the application provides commands to manage the trace data. As the size of a dataset can be large, the application lets the user monitor the datasets currently loaded and unload them, if necessary. Datasets can also be split into multiple smaller chunks; splitting is helpful when metrics require multiple repetitions of the experiments for the same device under test.

The application supports scripting: The users can load text files with a list of commands to be executed sequentially, thus creating pipelines of operations on the data and avoiding redundant typing otherwise required by the command-line interface.

B.5 Optimization And Performance

At the moment of writing, MetriSCA is not multi-threaded. However, it implements various techniques to optimize the analysis runtime. For example, the application supports incremental computation of the correlation, which allows the metrics to be computed with a different number of traces without having to recompute the correlation at each computational step.

Some metrics computation, depending on the dataset size, can take considerable time. Furthermore, some intermediate results can often be reused across metrics. The application, therefore, provides a caching mechanism that saves intermediate results to the disk, as binary files. The application is then able to recognize the files that can be reused, based on the information encoded in the file names, and further speed up the computation. If desired, the performance optimizations can be disabled using the corresponding command-line arguments.