








# SoK: A Methodology to Achieve Provable Side-Channel Security in Real-World Implementations

Sonia Belaïd<sup>1</sup> , Gaëtan Cassiers<sup>1</sup> , Camille Mutschler<sup>2</sup>,  
Matthieu Rivain<sup>1</sup> , Thomas Roche<sup>3</sup>, François-Xavier Standaert<sup>4</sup>  and  
Abdul Rahman Taleb<sup>1</sup> 

<sup>1</sup> CryptoExperts, Paris, France

<sup>2</sup> Idemia, Courbevoie, France

<sup>3</sup> NinjaLab, Montpellier, France

<sup>4</sup> UCLouvain, Louvain-la-Neuve, Belgium

**Abstract.** A wide range of countermeasures have been proposed to defend against side-channel attacks, with masking being one of the most effective and commonly used techniques. While theoretical models provide formal security proofs, these often rely on assumptions—sometimes implicit—that can be difficult to assess in practice. As a result, the design of secure masked implementations frequently combines proven theoretical arguments with heuristic and empirical validation. Despite the significant body of work, the literature still lacks a cohesive and well-defined framework for translating theoretical security guarantees into practical implementations on physical devices. Specifically, there remains a gap in connecting provable results from abstract models to quantitative security guarantees at the implementation level.

In this Systematization of Knowledge (SoK), we aim to provide a comprehensive methodology to transform abstract cryptographic algorithms into physically secure implementations against side-channel attacks on microcontrollers. We introduce new tools to adapt the ideal noisy leakage model to practical, real-world scenarios, and we integrate state-of-the-art techniques to build secure implementations based on this model. Our work systematizes the design objectives necessary for achieving high security levels in embedded devices and identifies the remaining challenges in concretely applying security reductions. By bridging the gap between theory and practice, we seek to provide a foundation for future research that can develop implementations with proven security against side-channel attacks, based on well-understood leakage assumptions.

**Keywords:** masking · provable side-channel security · random probing model · noisy leakage model · methodology · physical assumptions

## 1 Introduction

Cryptographic algorithms are traditionally analyzed within the black-box model, where the adversary’s knowledge is limited to inputs and outputs. However, since the late 1990s, side-channel attacks have demonstrated that physical implementations of cryptographic algorithms on embedded devices can be vulnerable to physical emanations, such as

---

E-mail: [sonia.belaid@cryptoexperts.com](mailto:sonia.belaid@cryptoexperts.com) (Sonia Belaïd), [gaetan.cassiers@uclouvain.be](mailto:gaetan.cassiers@uclouvain.be) (Gaëtan Cassiers), [camille.mutschler@idemia.com](mailto:camille.mutschler@idemia.com) (Camille Mutschler), [matthieu.rivain@cryptoexperts.com](mailto:matthieu.rivain@cryptoexperts.com) (Matthieu Rivain), [thomas@ninjalab.io](mailto:thomas@ninjalab.io) (Thomas Roche), [fstandae@uclouvain.be](mailto:fstandae@uclouvain.be) (François-Xavier Standaert), [taleb.abdulrahmani@gmail.com](mailto:taleb.abdulrahmani@gmail.com) (Abdul Rahman Taleb)



the execution time [Koc96], device temperature [HS13], power consumption [KJJ99], or electromagnetic radiation [QS01] during the algorithm execution. These physical emissions leak information about the internal state of cryptographic operations, thus breaking the theoretical security guarantees provided by the black-box model.

In response, several countermeasures have been studied to protect cryptographic algorithms. Among the different approaches, one of the most widely used is known as *masking*, simultaneously introduced by Chari et al. [CJRR99], and by Goubin and Patarin [GP99] in 1999. It consists in splitting a sensitive variable  $x$  into  $n$  random *shares*, among which any combination of  $n - 1$  shares does not reveal any secret information. This can be achieved by generating  $n - 1$  shares uniformly at random  $x_1, \dots, x_{n-1}$  and computing the last share  $x_n$  so that  $x = x_1 * \dots * x_{n-1} * x_n$  according to some group law  $*$ . The motivation is to make it more difficult for an attacker to recover a secret by manipulating the shares instead of the sensitive value. Indeed, the adversary must recombine information from all the shares to learn something about the sensitive value. Assuming that the information gathered through side-channel observations involves some kind of noise, it has been shown that it becomes exponentially harder to recover the secret as the number of shares grows [BCG<sup>+</sup>23, BS21, GS18].

Meanwhile, proving or validating such security levels in practice remains a challenging task. Generally, providing security guarantees against side-channel attacks is tricky, and several works tackle this issue [DFS15a, JS17, GPSS18, BS21]. The approaches currently found in the literature range from purely qualitative solutions such as leakage detection (e.g., ISO17825 [2716, WO19]) or test vector leakage assessment (TVLA) [GGJR<sup>+</sup>11], which aim to detect information leakage using statistical analysis, to more quantitative solutions such as mounting known attacks on the implementation and inferring the security level from the best attacks. For instance, common-criteria certification procedures currently follow this empirical approach to validate the security of implementations for smartcards against side-channel attacks [BS21, BGNT15]. Yet, these methods often lack the formal rigor and comprehensive security guarantees that theoretical models offer. This dichotomy highlights a critical gap between theory and practice in the field of side-channel security.

To address this gap, the community has introduced formal leakage models that provide a theoretical foundation for analyzing masked implementations. The  $t$ -probing model, introduced by Ishai, Sahai, and Wagner in 2003 [ISW03], is the most well-known and allows for formal security proofs by assuming that an adversary can observe the exact values of up to  $t$  intermediate variables. An arithmetic circuit is then secure in this model if no such leakage of  $t$  variables reveals information about the sensitive variables. Despite its wide use by the community [SP06, RP10, CPRR14, BBP<sup>+</sup>16, CRZ18] thanks to its convenience to build security proofs, the  $t$ -probing model sometimes fails to reflect the reality of embedded devices. For instance, it does not capture *horizontal attacks* [BCPZ16], which exploit the repeated manipulation of variables within an execution.

These issues motivated the formalization of the *noisy leakage model* [PR13]. This model better captures the reality of embedded devices by assuming that each intermediate variable leaks a noisy function of its value. However, proving security in the noisy leakage model [BCG<sup>+</sup>23, MS23] is more complex than in the  $t$ -probing model. In 2014, Duc, Dziembowski, and Faust [DDF14, DDF19] proposed a security reduction from the noisy model to the  $t$ -probing model, relying on an intermediate model, the *random probing model*, which benefits from a tighter reduction with the noisy leakage model. In a nutshell, it assumes that every wire in the circuit leaks with some constant leakage probability. This leakage probability is related to the amount of side-channel noise in practice. The random probing model captures *horizontal attacks*, and has been studied recently in many works [Ajt11, ADF16, AIS18, BCP<sup>+</sup>20, BRT21, BRTV21, CFOS21].

Despite the usefulness of these leakage models in providing formal guarantees, their application to practical implementations remains an ongoing challenge. The literature currently lacks a unified methodology for translating these theoretical constructions into secure physical implementations. Additionally, practical implementations often violate the key assumptions of the models, particularly the *data isolation* assumption—which states that the leakage of an elementary operation depends only on its inputs—and the *noise independence* assumption, which assumes that noise from different operations is independent.

The data isolation assumption can be easily broken, for instance, due to physical effects on a device. In particular, transitions occurring on memory buses or CPU registers between a previously processed value  $\mathbf{x}_{i-1}$  and the current one  $\mathbf{x}_i$  usually leak some information correlated to  $\mathbf{x}_{i-1} \oplus \mathbf{x}_i$ , which violates the data isolation principle [CGP<sup>+</sup>12, BGG<sup>+</sup>14]. On the hardware level, glitches further make the successive gates’ leakages mutually dependent on their respective inputs [MPG05, MPO05]. On the software level, CPU synchronization limits, but does not eliminate, the issue of glitches. These issues can be avoided by adding registers and controlling transitions [FGM<sup>+</sup>18, CS21] in hardware, and by trying to avoid transitions using assembly programming tricks in software [GMPO19, BC22, BGG<sup>+</sup>22]. However, these techniques still rely on abstract models for the leakage, and current techniques in the literature test the data isolation assumption only indirectly, by estimating the statistical security order of an implementation [BDF<sup>+</sup>17, SM15].

The noise independence assumption is similarly difficult to guarantee in practice. Side-channel leakage is typically multivariate, and noise from successive operations often exhibits dependencies, complicating the assumption of independence. Although this assumption is frequently made, its impact on security has only been explored at a high level in the literature [GS18, Cho15].

**Contributions.** Our contributions can be summarized as follows.

*Methodology for Secure Implementations.* We present a comprehensive and systematic methodology to transform abstract cryptographic algorithms into concrete implementations that are secure against side-channel attacks on microcontrollers. This process leverages a random probing compiler and carefully applies the reduction from the noisy leakage model to the random probing model, making it applicable to physical implementations. While this reduction is well understood in theoretical contexts, our contribution lies in systematically summarizing the practical assumptions and challenges that must be addressed to achieve formally secure implementations on real-world devices.

*Novel Tools for Practical Challenges.* We introduce new tools to address key technical challenges in adapting theoretical models to physical implementations:

- **Data Isolation Assumption Validation:** We explain how to enforce the data isolation assumption and introduce a novel practical test for its validation on a physical implementation. This test, conducted on a real STM32F3 MCU target using NewAE’s ChipWhisperer-Lite CW1173 board, offers a direct and practical approach, standing in contrast to indirect validation methods previously discussed in the literature. While the test does not offer formal proof of the assumption, it is the first of its kind to directly address and empirically validate this hypothesis in a systematic and reproducible manner.
- **Noise Independence Analysis:** We propose a method for integrating the noise independence assumption into security analyses, allowing for the quantification of security loss when this assumption is not fully met. Specifically, we introduce a relaxation of the assumption that splits the noise affecting each operation during algorithm

execution into independent components. We first demonstrate a straightforward method for achieving this split and then formulate the problem as a constrained optimization to improve scalability for larger computations. While we provide a non-optimal but practical solution, we leave the development of an optimal and efficient solution as an open problem for future research.

*Identifying Design Goals and Remaining Challenges.* We systematically identify the key design objectives necessary to achieve high-security levels in physical implementations using our security reduction methodology. Furthermore, we highlight the remaining limitations and open questions concerning the practical usability of leakage models. Our SoK contribution lies in bridging the gap between theoretical models and practical implementation challenges, motivating further research to close this gap. For example, future work could explore the quantification of signal independence loss or identify an optimal solution for the noise split relaxation to maximize security in practice.

**Organization.** The organization of the paper is as follows. In [Section 2](#), we provide the necessary background and formalize key assumptions. [Section 3](#) outlines our proposed methodology and states our main result: our methodology can produce implementations that are practically secure against side-channel attacks. We further detail the procedures to enforce or relax the identified assumptions in [Section 4](#) and demonstrate the application of our methodology on a real-world target (STM32F3 MCU with ChipWhisperer-Lite CW1173). Finally, we conclude with discussions and future research directions in [Section 5](#).

## 2 Technical Background

### 2.1 Notations

We denote by  $\mathcal{V}$  a finite set called the variable space and by  $\mathcal{X}$  the input space for the leakage. We denote by  $\mathcal{Y}$  the leakage distribution. We use capital letters to denote random variables over a set or a distribution, e.g.,  $X$  denotes a random variable over  $\mathcal{X}$ , and  $Y(\mathbf{x})$  denotes a random variable (or equivalently a leakage function) over the distribution  $\mathcal{Y}$ , taking as input  $\mathbf{x}$ , a value over the input space  $\mathcal{X}$ . We denote by  $\mathbf{y}$  a leakage trace, i.e. a realization of  $Y(\mathbf{x})$ . Any two probability distributions  $D_1$  and  $D_2$  are said  $\varepsilon$ -close, denoted  $D_1 \approx_\varepsilon D_2$ , if their statistical distance is upper bounded by  $\varepsilon$ , that is  $\text{SD}(D_1; D_2) := \frac{1}{2} \sum_x |p_{D_1}(x) - p_{D_2}(x)| \leq \varepsilon$ , where  $p_{D_1}(\cdot)$  and  $p_{D_2}(\cdot)$  denote the probability mass functions of  $D_1$  and  $D_2$ .

### 2.2 Abstract Circuits

We recall the definition of an abstract circuit (family) and the definition of a circuit compiler (CC, Enc, Dec) which turns an abstract circuit into a randomized circuit.

**Definition 1** (Abstract Circuit Family). An *abstract circuit family* is a pair  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$  such that

- $\mathcal{V}$  is the *variable space*,
- $\mathcal{G}$  is a set of functions, called the gate family. For each function  $g \in \mathcal{G}$ , there exists  $\ell, m \in \mathbb{N}$  such that  $g : \mathcal{V}^\ell \rightarrow \mathcal{V}^m$ .

An *abstract circuit*  $C$  belonging to the family  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ , which is written  $C \in \mathbb{C}$ , is defined as an acyclic directed graph whose edges are *wires* carrying values over  $\mathcal{V}$ , and vertices are *gates* processing operations over  $\mathcal{V}$ . It is further formally composed of input gates of fan-in 0 and fan-out 1 and output gates of fan-in 1 and fan-out 0. Evaluating

an  $\ell$ -input  $m$ -output circuit  $C$  consists of writing an input  $\mathbf{x} \in \mathcal{V}^\ell$  in the  $\ell$  input gates, processing the gates from input gates to output gates, then reading the output  $\mathbf{z} \in \mathcal{V}^m$  from the  $m$  output gates. This is denoted by  $\mathbf{z} = C(\mathbf{x})$ . During the evaluation process, each wire in the circuit is assigned with a value on  $\mathcal{V}$ . We call the tuple of all these wire values a *wire assignment* of  $C$  (on input  $\mathbf{x}$ ).

**Definition 2** (Circuit Compiler). A *circuit compiler* is a triplet of algorithms (CC, Enc, Dec) defined as follows:

- CC (circuit compilation) is a deterministic algorithm that takes as input an abstract circuit  $C$  from a family of circuits  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$  and outputs a randomized circuit  $\widehat{C}$ .
- Enc (input encoding) is a probabilistic algorithm that maps an input  $\mathbf{x} \in \mathcal{V}^\ell$  to an encoded input  $\widehat{\mathbf{x}} \in \mathcal{V}^{\ell'}$ .
- Dec (output decoding) is a deterministic algorithm that maps an encoded output  $\widehat{\mathbf{z}} \in \mathcal{V}^{m'}$  to a plain output  $\mathbf{z} \in \mathcal{V}^m$ .

These three algorithms satisfy the following properties:

- **Correctness:** For every circuit  $C$  of input length  $\ell$ , and for every  $\mathbf{x} \in \mathcal{V}^\ell$ , we have

$$P[\text{Dec}(\widehat{C}(\widehat{\mathbf{x}})) = C(\mathbf{x}) \mid \widehat{\mathbf{x}} \leftarrow \text{Enc}(\mathbf{x})] = 1 ,$$

where  $\widehat{C} = \text{CC}(C)$ .

- **Efficiency:** For some security parameter  $\lambda \in \mathbb{N}$ , the running time of  $\text{CC}(C)$  is  $\text{poly}(\lambda, |C|)$ , the running time of  $\text{Enc}(\mathbf{x})$  is  $\text{poly}(\lambda, |\mathbf{x}|)$  and the running time of  $\text{Dec}(\widehat{\mathbf{z}})$  is  $\text{poly}(\lambda, |\widehat{\mathbf{z}}|)$ , where  $\text{poly}(\lambda, q) = O(\lambda^{k_1} q^{k_2})$  for some constants  $k_1, k_2$ .

## 2.3 Random-Probing Model

Let  $p \in [0, 1]$  be some constant leakage probability parameter, usually called *leakage rate*. The random probing leakage can be defined in two ways depending on whether we consider leakage on the wires or the gates of an abstract circuit  $C$  from a family  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ .

In the *wire leakage* setting, the  $p$ -random probing model states that during the evaluation of a circuit  $C$ , each wire leaks its value with probability  $p$  (and leaks nothing otherwise), where all the wire leakage events are mutually independent. To formally define this leakage, we consider two probabilistic algorithms:

- The *leaking-wires sampler* takes as input an abstract circuit  $C$  and a probability  $p \in [0, 1]$ , and outputs a set  $W$ , denoted as

$$W \leftarrow \text{LeakingWires}(C, p) ,$$

where  $W$  is constructed by including each wire label from the circuit  $C$  with probability  $p$  to  $W$  (where all the probabilities are mutually independent).

- The *assign-wires sampler* takes as input an abstract circuit  $C$ , a set of wire labels  $W$  (subset of the wire labels of  $C$ ), and an input  $\mathbf{x} \in \mathcal{V}^\ell$ , and it outputs a  $|W|$ -tuple  $\mathbf{w} \in \mathcal{V}^{|W|}$ , denoted as

$$\mathbf{w} \leftarrow \text{AssignWires}(C, W, \mathbf{x}) ,$$

where  $\mathbf{w}$  corresponds to the assignments of the wires of  $C$  with label in  $W$  for an evaluation on input  $\mathbf{x}$ .

By convention, we do not consider leakage on the output wires (i.e. input wires of the output gate) of a circuit, since when composing several circuits, these wires become input wires to the next circuit.

We can analogously define the *gate leakage* setting with similar probabilistic procedures, with the difference between both leakages illustrated in Figure 1. The leaking-gates sampler  $G \leftarrow \text{LeakingGates}(C, p)$  outputs a set  $G$  of gates labels instead of wire labels ( $\text{LeakingGates}$  ignores the input and output gates, which are merged when composing circuits). Then, the assign-gates sampler  $\mathbf{g} \leftarrow \text{AssignGates}(C, G, \mathbf{x})$  assigns to each gate of label in  $G$  its internal state during the evaluation of  $C$  (i.e.  $\mathbf{g}$  is the assignment of the internal states of the gates of  $C$  with label in  $G$  for an evaluation on input  $\mathbf{x}$ ). The internal state of a gate is modelled as an arbitrary function of the gate's inputs.

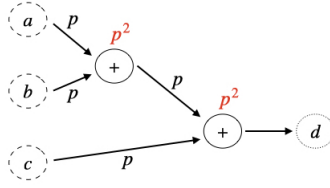


Figure 1: Toy circuit illustrating random probing leakage. Dashed circles ( $a, b, c$ ) are input gates, while the dotted circle ( $d$ ) is the output gate. In the wire leakage setting, each wire leaks with probability  $p$ , while in the gate leakage setting, each gate leaks its internal state with probability  $p^2$ . Lemma 1 states that if the circuit is  $(p, \varepsilon)$ -RP secure in the wire setting, then it is  $(p^2, \varepsilon)$ -RP secure in the gate setting.

Based on these notions, we now formally define the (wire or gate) random probing leakage of a circuit.

**Definition 3** (Random Probing Leakage). The  $p$ -random probing wire leakage of an abstract circuit  $C$  with  $\ell$  inputs, on input  $\mathbf{x} \in \mathcal{V}^\ell$  is the distribution  $\mathcal{L}_p^{\text{wire}}(C, \mathbf{x})$  obtained by composing the leaking-wires and assign-wires samplers as

$$\mathcal{L}_p^{\text{wire}}(C, \mathbf{x}) \stackrel{\text{id}}{=} \text{AssignWires}(C, \text{LeakingWires}(C, p), \mathbf{x}) .$$

For the  $p$ -random probing gate leakage,  $\mathcal{L}_p^{\text{gate}}(C, \mathbf{x})$  is obtained as

$$\mathcal{L}_p^{\text{gate}}(C, \mathbf{x}) \stackrel{\text{id}}{=} \text{AssignGates}(C, \text{LeakingGates}(C, p), \mathbf{x}) .$$

We can define the random probing security of an abstract circuit  $C$ .

**Definition 4** (Random Probing Security). An abstract circuit  $C$  with  $\ell$  inputs, from a family of circuits  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ , is  $(p, \varepsilon)$ -random probing secure (RPS) in the wire leakage setting with respect to encoding  $\text{Enc}$  if there exists a simulator  $\text{Sim}$  such that for every  $\mathbf{x} \in \mathcal{V}^\ell$ :

$$\text{Sim}(C) \approx_\varepsilon \mathcal{L}_p^{\text{wire}}(C, \text{Enc}(\mathbf{x})) . \quad (1)$$

A circuit compiler  $(\text{CC}, \text{Enc}, \text{Dec})$  is  $(p, \varepsilon)$ -random probing secure in the wire leakage setting if, for every circuit  $C$ , the compiled circuit  $\hat{C} = \text{CC}(C)$  is  $(p, |C| \cdot \varepsilon)$ -random probing secure in the wire leakage setting where  $|C|$  is the size of the original circuit.

We equivalently define  $(p, \varepsilon)$ -random probing security for a circuit and a circuit compiler in the gate leakage setting, where we use  $\mathcal{L}_p^{\text{gate}}$  instead of  $\mathcal{L}_p^{\text{wire}}$ .

We have the following reduction of security, which states that if a circuit is random probing secure in the wire leakage setting, then it is secure in the gate leakage setting <sup>1</sup>.

<sup>1</sup>Note that Lemma 1 extends to gates with larger fan-in. Considering gates with  $n$  inputs, one gets  $p' = p^n$

**Lemma 1.** *Let  $C$  be an abstract circuit with  $\ell$  inputs from a family of circuits  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$  such that each gate  $g \in \mathcal{G}$  has at most two input wires. If  $C$  is  $(p, \varepsilon)$ -random probing secure with respect to encoding  $\text{Enc}$  in the wire leakage setting, then  $C$  is  $(p', \varepsilon')$ -random probing secure in the gate leakage setting, with  $p' = p^2$  and  $\varepsilon' = \varepsilon$ .*

*Proof of Lemma 1.* Let  $C$  be an abstract circuit with  $\ell$  inputs and suppose that  $C$  is  $(p, \varepsilon)$ -random probing secure in the wire leakage setting. Then, there exists a simulator that we shall denote  $\text{Sim}_{\text{wire}}$  such that  $\text{Sim}_{\text{wire}}(C) \approx_{\varepsilon} \mathcal{L}_p^{\text{wire}}(C, \text{Enc}(\mathbf{x}))$ . We now construct another simulator  $\text{Sim}_{\text{gate}}$  as follows.  $\text{Sim}_{\text{gate}}$  starts by running  $\text{Sim}_{\text{wire}}$ , and if  $\text{Sim}_{\text{wire}}$  fails (or aborts), then  $\text{Sim}_{\text{gate}}$  aborts too. Otherwise, for each gate  $g$  in  $C$ , if all input wires to  $g$  are simulated and output by  $\text{Sim}_{\text{wire}}$ , we let  $\text{Sim}_{\text{gate}}$  output a simulation of the inner state of  $g$  using the simulation of its input wires by  $\text{Sim}_{\text{wire}}$ . Note that this is possible since the inner state of  $g$  only depends on its input wires. Since we have that  $\text{Sim}_{\text{wire}}(C) \approx_{\varepsilon} \mathcal{L}_p^{\text{wire}}(C, \text{Enc}(\mathbf{x}))$ , then each wire in  $C$  is simulated by  $\text{Sim}_{\text{wire}}$  with probability  $p$  independently of all the other wires. Consequently, each gate in  $C$  is simulated by  $\text{Sim}_{\text{gate}}$  with probability at least  $p^2$  independently of all the other gates. Finally, since  $\text{Sim}_{\text{gate}}$  aborts if and only if  $\text{Sim}_{\text{wire}}$  aborts with probability  $\varepsilon$ , we get that  $\text{Sim}_{\text{gate}}(C) \approx_{\varepsilon} \mathcal{L}_p^{\text{gate}}(C, \text{Enc}(\mathbf{x}))$ . Hence,  $C$  is  $(p^2, \varepsilon)$ -random probing secure in the gate leakage setting, which concludes the proof.  $\square$

## 2.4 Noisy Leakage Model

The noisy leakage model was formalized in [PR13]. In this model, a leaking computation is modeled by a sequence of *elementary operations*  $(g_i)_i$  accessing a common memory called *internal state*. Each elementary operation reads its input and writes its output on the internal state. When processed on some input  $\mathbf{x}$ , an elementary operation  $g_i$  reveals  $f_i(\mathbf{x})$  to the adversary for some *noisy leakage function*  $f_i$ . A noisy leakage function takes two arguments: the value  $\mathbf{x}$  held by the accessed part of the internal state (*data isolation assumption*) and a random string  $\rho$  long enough to model the leakage noise. Each execution leaks the values  $(f_i(\mathbf{x}_i, \rho_i))_i$  where the  $\mathbf{x}_i$ 's are the successive intermediate values (from the internal state) in input of the elementary operations  $g_i$ 's and  $\rho_i$ 's are fresh random strings. We stress that all the  $\rho_i$ 's involved in successive executions are uniformly and independently drawn (*independent noise assumption*).

We note that from a formal point of view, there is an equivalence between the circuit model used by the gate-leakage random probing model and the internal state model used by the noisy leakage model. In both cases the computation is divided into sub-computations (either gates or elementary operations) and the full leakage is composed of the outputs of leakage functions (either random probing functions or noisy functions) applied to all the sub-computations input. The internal state model has the advantage of being cosmetically closer to a real software implementation, moreover it is useful to consider the order of operations while relaxing the data isolation and noise independence assumptions (as discussed later).

For the sake of simplicity, we shall omit the random string parameter, which leads to the notation  $f_i(\mathbf{x})$  where  $\mathbf{x}$  is the accessed value. Note that  $f_i(\mathbf{x})$  can be seen as the output of a probabilistic algorithm. In particular,  $f_i(\mathbf{x})$  can take several values with a given probability distribution, and can therefore be considered as a random variable. The noisy property of  $f$  is captured by assuming that the bias introduced in the distribution of a uniform random variable  $X$  given the leakage  $f(X)$  is bounded. This is formalized in the next definition:

**Definition 5** (Noisy Function). Let  $\mathcal{X}$  be a finite set and let  $\delta \in \mathbb{R}$ . A  $\delta$ -noisy leakage function  $f$  on  $\mathcal{X}$  is a function of domain  $\mathcal{X} \times \{0, 1\}^{|\rho|}$  for some  $|\rho| \in \mathbb{N}$  such that

$$\beta(X|Y) := \sum_{y \in \text{Range}(f)} \Pr(Y = y) \cdot \Delta((X | Y = y); X) \leq \delta, \quad (2)$$

where  $\Delta$  is a statistical distance measure,  $X$  is a uniform random variable over  $\mathcal{X}$  and where  $Y = f(X, R)$  for a uniform random variable  $R$  over  $\{0, 1\}^{|\rho|}$ .

The above definition depends on the notion of statistical distance. In the original definition from [PR13], the authors use the  $L_2$  norm. The authors of [DDF14] then suggested to use the  $L_1$  norm (normalized by  $\frac{1}{2}$ ). It was later suggested in [PGMP19] to use a statistical distance notion based on the *relative error*. Noisy functions based on this distance are referred to as *average relative error* (ARE) noisy leakage functions in [PGMP19] since the relative error is averaged over the distribution of the leakage  $Y$  in Equation 2.

As recalled hereafter, the noisy leakage metrics based on the  $L_1$  statistical distance (SD) and the ARE enjoy useful security reductions to the random probing model. We recall the definition of these two metrics based on the *pointwise mutual information*.

**Definition 6** (Pointwise Mutual Information). Let  $X, Y$  be random variables over  $\mathcal{X}, \mathcal{Y}$  respectively. For any  $x \in \mathcal{X}, y \in \mathcal{Y}$ , the exponential form of the pointwise mutual information (PMI) is defined as:

$$\text{PMI}_{X,Y}(x, y) = \frac{P[X = x, Y = y]}{P[X = x] \cdot P[Y = y]} - 1 .$$

**Definition 7.** Let  $X, Y$  be random variables over  $\mathcal{X}, \mathcal{Y}$  respectively. We can define the L1 statistical distance (SD) as follows:

$$\text{SD}(X|Y) = \frac{1}{2} \mathbb{E}_{Y=y} \mathbb{E}_{X=x} [|\text{PMI}_{X,Y}(x, y)|] .$$

The average relative error (ARE) can also be expressed as:

$$\text{ARE}(X|Y) = \mathbb{E}_{Y=y} \left[ \max_x |\text{PMI}_{X,Y}(x, y)| \right] .$$

**From random probing to noisy leakage security.** In [DDF14], Duc, Dziembowski, and Faust show the following security reduction: any circuit which is  $(p, \varepsilon)$ -secure in the random probing model is also  $(\delta, \varepsilon)$ -secure in the noisy leakage model (for the same parameter  $\varepsilon$ ) defined w.r.t. the metric  $\beta(X|Y) = \text{SD}(X|Y)$  and for any  $\delta \leq p/|\mathcal{X}|$ , where  $\mathcal{X}$  is the input space of the abstract gates / elementary operations.<sup>2</sup> This result was later extended to the noisy leakage model defined w.r.t. the metric  $\beta(X|Y) = \text{ARE}(X|Y)$  in the work of Prest et al. [PGMP19]. Those security reductions directly hold from the following key lemma.

**Lemma 2** ([DDF14, PGMP19]). Let  $\phi_p : \mathcal{X} \rightarrow \mathcal{X} \cup \{\perp\}$  the randomized function defined for every  $p \in [0, 1]$  as

$$\phi_p(x) = \begin{cases} \perp & \text{with probability } 1 - p \\ x & \text{with probability } p \end{cases} \quad (3)$$

Let  $f : \mathcal{X} \rightarrow \mathcal{Y}$  be a  $\delta$ -noisy leakage function (w.r.t. SD or ARE). There exists a randomized function  $f' : \mathcal{X} \cup \{\perp\} \rightarrow \mathcal{Y}$  such that for every  $x \in \mathcal{X}$  we have

$$f(x) = f'(\phi_p(x)) \quad \text{with} \quad \begin{cases} p \leq \delta \cdot |\mathcal{X}| & \text{if } \text{SD}(X|f(X)) \leq \delta \\ p \leq \delta & \text{if } \text{ARE}(X|f(X)) \leq \delta \end{cases} \quad (4)$$

<sup>2</sup>The input space  $\mathcal{X}$  is different than the variable space  $\mathcal{V}$  for the variables in a circuit. Typically, when the leakage is defined on the internal state of the gate, the latter can be described by both its input wires, and hence the input space is  $\mathcal{X} = \mathcal{V}^2$ .



We recall that the ARE is a worst-case metric, contrary to the SD, which is an average-case metric. This explains the tighter reduction (i.e. no loss induced by the size of the input space) using the ARE from the noisy model to the random probing model since the latter is also a worst-case model.

Besides this worst-case vs. average-case question, we note that the SD and ARE can be connected to metrics that are used in practice to evaluate the security of a leaking implementation. For example, the SD can be expressed using Mutual Information (MI) thanks to [Dod12] and the Mutual Information can (under some conditions) be expressed using the Signal-to-Noise Ratio (SNR) [Man04] and the correlation coefficient [BCO04] thanks to [MOS11]. The MI is a standard metric to analyze multivariate leakages while the SNR and correlation coefficient are among the most popular tools for univariate security assessments.

In the following, we shall refer to the reduction from [DDF14] using the SD metric as the *DDF reduction*, and to that from [PGMP19] using the ARE metric as the *PGMP reduction*.

## 2.5 Physical Assumptions

The noisy leakage model has been argued to capture power and electromagnetic leakages effectively. In all generality, an elementary operation processing a value  $\mathbf{x}$  gives rise to a leakage trace  $Y(\mathbf{x})$  which is a multivariate random variable (a.k.a. a random vector) following a distribution whose parameters depend on  $\mathbf{x}$ . In most practical contexts, this distribution is well approximated by a multivariate Gaussian  $\mathcal{N}(\mathbf{m}_x, \Sigma)$  for some parameters  $\mathbf{m}_x$  (mean vector) and  $\Sigma$  (covariance matrix), see e.g. [CRR03, SLP05]. Such parameters can be inferred in practice through a profiling of the device, from which we obtain the noisy leakage metric  $\delta$  by evaluating Equation 2.

We still need to stress that, as is, the noisy leakage model relies on two assumptions about the underlying physical device which might not be verified in practice without further care.

**Assumption 1** (Data isolation). *A leakage function  $f_i$  corresponding to the elementary operation  $g_i(\mathbf{x}_i)$  only depends on the current state  $\mathbf{x}_i$  and not on previously accessed parts of the state:  $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ . The leakage is then assumed to respect some data isolation between successive elementary operations.*

However, as mentioned in the introduction, physical effects such as glitches and transitions will likely break this implicit assumption. Hence, one should take special care and enforce data isolation for the model to be valid.

Note that this assumption can be relaxed to allow the leakage to be composed of linear combinations of independent noisy leakage functions [BDF<sup>+</sup>17]. Such linear combinations do not affect the statistical security order since given leakage samples of which the statistical moments up to a given order are independent of any sensitive variable, their linear combination will have the same guarantee

**Assumption 2** (Noise independence). *The leakage noises from the successive elementary operations are independent from each other. Formally, the random tape  $\rho_i$  in each  $f_i(\mathbf{x}_i, \rho_i)$  is sampled as a fresh uniform string.*

In practice, this assumption does not easily hold: if one cuts a leakage trace into several sub-traces corresponding to successive elementary operations, the noises in the successive sub-traces would likely include some part of dependency. Indeed, a correlation exists between successive leakage points, which makes multivariate statistics particularly useful for side-channel attacks [CRR03].

In the following, we shall refer to the original noisy leakage model, which relies on the two aforementioned assumptions as the *idealized noisy leakage model*. We will explore

how to relax or enforce those physical assumptions to reduce the security of a physical implementation to that of an abstract implementation in the idealized noisy leakage model, which subsequently reduces to the random probing security.

### 3 Methodology

The theoretical community introduced many constructions proven secure in the (random) probing and noisy leakage models with a quantified security level. Meanwhile, it is unclear how to implement such constructions on physical devices while fully/quantitatively leveraging the proven security guarantees. Indeed, the existing literature does not explain all the steps nor states all the hypotheses required for preserving proven security claims in practice. In this section, we rigorously exhibit all the steps to turn an abstract circuit into a physical implementation satisfying provable security against side-channel attacks<sup>3</sup>. As illustrated in Figure 2, these steps can be split into two phases:

- (i) a *characterization phase* which only depends on the device and the side-channel acquisition tool (i.e., without any knowledge of the abstract circuit). It includes the implementation of specific gates (Step 1), the analysis of **Assumption 1** (data isolation) to exhibit a relevant whitening procedure (Step 2), the characterization of the leakage (Step 3), the enforcement and relaxation of **Assumption 2** (noise independence) (Step 4), and the estimation of the noisy leakage parameter (Step 5),
- (ii) a *compilation phase* using the outputs of the characterization phase to turn an abstract circuit  $C \in \mathbb{C}$ , with  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$  an abstract circuit family, into a practically secure implementation for a given security level  $\lambda$  (Step 6). This phase relies on the usage of a secure random probing compiler.

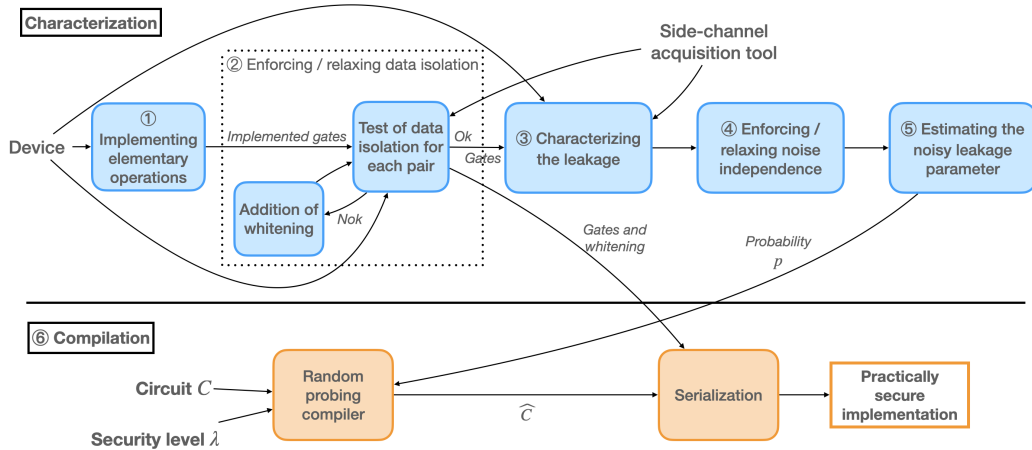


Figure 2: Illustration of our methodology

Overall, our methodology provably ensures security against side-channel attacks, as shown in Section 3.7. All the intermediate steps are described at a high level in the next subsections. In Section 4, we describe the experiments that we ran to validate the practicality of our methodology and develop some dedicated procedures for data and noise isolation.

<sup>3</sup>We describe our methodology in the context of software implementation, where elementary calculations align with software routines. The generalization to hardware implementations is left for future work, see Section 5.

### 3.1 Step 1: Implementing Abstract Gates

The first step of our methodology consists of implementing abstract gates as software routines. A physical elementary operation abstracted as a gate by the noisy leakage model (see Section 2.4) first looks up its operands from memory (the computation state), then executes a sequence of arithmetic instructions (implementing the gate functionality  $g \in \mathcal{G}$ ), and finally writes back the result to memory. This process generates some side-channel leakage depending on the executed instructions and the processed data, which is the leakage of the physical elementary operation abstracted by the noisy leakage model. A developer must first translate this behavior into a software routine on a physical device. We propose to implement such a routine in ARM assembly as follows (with the `xor` operation as an example):

```
operation_xor:
    ldr r0, [r0]
    ldr r1, [r1]
    eor r0, r1, r0 // For other operations, change instruction
    str r0, [r2]
```

with the following C signature:

```
void operation_xor(const uint32* aPtr, const uint32* bPtr, uint32* cPtr);
```

We define a routine for each abstract gate  $g \in \mathcal{G}$  which, when executed on the target device, behaves as a physical elementary operation abstracted by the noisy model. From these implementations of the abstract gates, any circuit  $C \in \mathcal{C}$  can be compiled into a physical implementation on the target device. This implementation takes the form of a sequence of calls to the elementary operations, looking like the following C-syntax example:

```
operation1(a1Ptr, b1Ptr, c1Ptr);
operation2(a2Ptr, b2Ptr, c2Ptr);
...
```

The routines `operation1`, `operation2`... are all among the implemented gate routines which are mapped from the gates of the circuit. The pointer arguments (`a1Ptr`, `b1Ptr`, `c1Ptr`), (`a2Ptr`, `b2Ptr`, `c2Ptr`), ... are constant addresses triplets which encode the data dependency of the implementation, i.e., the wires in the abstract circuit.<sup>4</sup>

### 3.2 Step 2: Enforcing / Relaxing Data Isolation

Once the syntax of elementary operations is fixed, the physical assumptions made in the noisy leakage model must be satisfied by the implementations in order to use the security reduction. Our methodology first focuses on the data isolation assumption (i.e., Assumption 1), which requires that the leakage of an elementary operation only depends on its inputs, i.e., is independent of the inputs of the previous and the following operations. This assumption rarely holds in practice, since elementary operations executed successively might leak jointly on their manipulated data. Indeed, after the execution of an elementary operation, the data it has processed might be stored in the physical state of the CPU. The leakage of the following elementary operation will then be a (probabilistic) function of the data it processes and of the physical state of the CPU, hence of the previously processed data. This is a well-known issue in the side-channel literature. In particular, this data non-isolation includes the so-called transition leakage observed and analyzed in many works [MOW17, MPW22, MKSM22]. These pitfalls have a direct practical impact, typically leading to losing security orders in the masking scheme [BGG<sup>+</sup>14]. In the provable security setting, this translates to breaking the data isolation assumption. Assuming that each elementary operation leaks a (probabilistic) function of the accessed part of the state

<sup>4</sup>The proposed implementation style is admittedly not very efficient. This paper mainly targets security and simplicity, leaving optimization to future works.

is incorrect: the leakage also depends on the state’s previously accessed part(s). Hence, a developer can not simply implement a circuit as a sequence of the routines introduced in Section 3.1, as the side-channel security can no longer be reduced to the random probing model.

As data isolation plays a crucial role in upholding security proofs and stands as a critical step in our methodology, we introduce a method to enforce it, inspired by prior works (e.g., [BC22, CS21]). Additionally, we design a dedicated test to validate data isolation on a target device.

**Enforcing data isolation.** We use *data whitening* to enforce the data isolation assumption in our methodology. The principle is to call a routine on constant or random data whose sole purpose is to clean the CPU state from any dependency on the previously processed data. Specifically, after each call to an elementary operation routine, we insert one or more calls for which the arguments point to random or constant data in memory. The intuition is that by relying on a call to a similar elementary operation routine, we expect to clean the data path, namely to write random or constant data in any hardware register containing data-dependent information from the previous call. Nevertheless, although natural, this solution might not suffice to ensure data isolation on some devices. The effectiveness of a whitening routine depends on the microarchitecture of the device’s CPU. Therefore, a developer might have to empirically test several approaches before reaching successful and efficient isolation.

Even with an isolation that avoids all transition and glitches effects across operations, it might not be possible to partition the leakage trace in time intervals whose leakage corresponds to only a single operation. Indeed, the leakage is often subject to low-pass filtering inside the target chip or the measurement chain. As a result, the independent intrinsic leakage of many operations will be linearly combined in the measured trace. As previously mentioned, we can relax the noisy leakage model to allow the leakage to be composed of linear combinations of independent noisy leakage functions. In this case, we aim at ensuring that a leakage function  $f_i$  corresponding to the elementary operation  $g_i(\mathbf{x}_i)$  does not jointly depend on the current state  $\mathbf{x}_i$  and previously accessed parts of the state:  $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ . In other words,  $f_i$  depends on the current state and has at most linear dependencies on the previous parts  $\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots$ . With this relaxation, we still provide independence between the inputs of the different operations, i.e., data isolation.

**Testing data isolation.** In Section 4.2, we introduce a novel way to test the effectiveness of a data whitening routine. The idea is to suppose that the leakage distribution can be modeled as a sum of a deterministic function of the first operation’s inputs, a deterministic function of the second operation’s inputs, and some noise value. In other words, we test that the leakage can be decomposed in additive parts that do not jointly depend on the inputs of both operations.

### 3.3 Step 3: Characterizing the Leakage

Once data isolation is enforced and tested, one can safely infer the leakage distribution of each physical elementary operation. This is a classical problem in the side-channel literature, and we can rely on a solid theoretical and practical ground for this step. We rely on the common assumption [CK13, SLP05] that the leakage distribution  $\mathcal{Y}$  of an elementary operation with inputs  $\mathbf{x} \in \mathcal{V}^\ell$  takes the form of a deterministic function of  $\mathbf{x}$  plus an additive Gaussian noise:

$$\mathcal{Y}_{\mathbf{x}} = d(\mathbf{x}) + \mathcal{N}(\mathbf{0}, \Sigma) , \quad (5)$$

where the deterministic part of the leakage can be written as a linear combination of a predetermined basis of functions  $\mathcal{H} = \{h_1, \dots, h_m\}$ , i.e.:

$$d(\mathbf{x}) = \sum_{i=1}^m \alpha_i \cdot h_i(\mathbf{x}) . \quad (6)$$

The choice of the basis of functions  $\mathcal{H}$  is determined for each elementary operation routine depending on its internal variables. The basis should at least contain one function for each internal variable bit but might also include monomials of higher degrees due to possible coupling effects [FGM<sup>+</sup>18].

In our methodology, we suggest relying on linear regression in order to estimate the deterministic leakage  $d(\cdot)$ . It involves acquiring an initial set of  $\ell_1$  traces, which measure the leakage while executing the operation on  $\ell_1$  inputs generated uniformly at random. We then use this set to infer the coefficients  $\{\alpha_i\}_{i=1, \dots, m}$ . Subsequently, we can compute the covariance matrix using a new set of  $\ell_2$  traces on uniform random inputs, allowing us to recover (an estimation of) the covariance matrix  $\Sigma$ .

It's worth emphasizing that while we propose linear regression for leakage estimation, our methodology remains adaptable to other estimation methods. Techniques such as template attacks [CRR03], combined with dimensionality reduction [SA08, CDSU23], and the emerging use of machine learning for side-channel analysis [MDP19, PPM<sup>+</sup>23], all offer viable alternatives. Furthermore, we assume that the engineering effort of optimizing the measurement setup has been performed properly, such that it is infeasible for an adversary to acquire more informative measurements than the ones used for the characterization. Indeed, the characterization steps of the methodology and in particular the estimation of the noise level depend heavily on the optimization of the experimental setup [BUS21, CM24].

### 3.4 Step 4: Enforcing / Relaxing Noise Independence

Next, we consider the noise independence assumption (Assumption 2) needed for the reduction from the noisy leakage to the random probing model. Namely, in the idealized noisy leakage model, the noise that occurs during the execution of an elementary operation is drawn independently of the noise that occurs during the execution of the previous ones. Hence, this assumption must be satisfied in practice. Meanwhile, it is hard to enforce and test since no clear separation of the noise occurs during a leakage trace. In our methodology, we propose a novel way to relax this assumption. Namely, we keep the Gaussianity hypothesis, but we allow the leakage of the different operations to overlap. We characterize this relaxation and directly reflect it on the security level by providing a reduction from the noisy leakage model with potential noise dependence to the idealized noisy leakage model.

Concretely, we propose to partition the noise distribution into multiple distributions, all while minimizing leakage during each operation. In simpler terms, given  $k$  consecutive elementary operations of inputs  $\{\mathbf{x}_i\}_{1 \leq i \leq k}$ , we can represent the overall leakage distribution as

$$\mathcal{Y} = \sum_{i=1}^k d_i(\mathbf{x}_i) + \mathcal{N}(\mathbf{0}, \Sigma) \quad (7)$$

where  $d_i(\mathbf{x}_i)$  are the different deterministic signals of the operations, and the noise drawn from  $\mathcal{N}(\mathbf{0}, \Sigma)$  is the global noise. Thanks to the data isolation enforcement and test from Section 3.2, the deterministic signals are mutually data independent. Specifically, while the  $d_i$ 's might overlap on some time samples, they independently apply to the inputs  $\mathbf{x}_i$ . This ensures that the global deterministic leakage can be expressed as a sum in Equation 7.

In order to relax the noise independence, our approach consists of finding a set of covariance matrices  $\{\Sigma_i\}_{i \in [k]}$  such that  $\sum_{i \in [k]} \Sigma_i = \Sigma$ . This way, we can split the Gaussian noise distribution  $\mathcal{N}(\mathbf{0}, \Sigma)$  into  $k$  independent Gaussian distributions  $\mathcal{N}(\mathbf{0}, \Sigma_1), \dots, \mathcal{N}(\mathbf{0}, \Sigma_k)$ . This representation enables us to split the leakage distribution into several functions  $\mathcal{Y}_i = d_i(\mathbf{x}_i) + \mathcal{N}(\mathbf{0}, \Sigma_i)$  for every  $i \in \{1, \dots, k\}$ . An adversary given a leakage sample of each  $\mathcal{Y}_i$  is more powerful than an adversary given a sample of the global leakage  $\mathcal{Y}$  because the former can always sum the  $\mathcal{Y}_i$  samples to get a  $\mathcal{Y}$  sample.

**Instantiation.** Section 4.4 introduces several methods to define such matrices  $\Sigma_i$ , which sum to the original covariance matrix  $\Sigma$  while minimizing the mutual information between the leakage and the signals.

### 3.5 Step 5: Estimating the Noisy Leakage Parameter

Recall that once data isolation is enforced and tested as described in Section 3.2, we can empirically characterize the leakage distribution of each elementary operation as described in Section 3.3. Assuming that the leakage takes the form of Equation 5, we compute the coefficients in the deterministic function for each elementary operation (as defined in Equation 6). Then, we can apply the noise relaxation described in Section 3.4 to get the covariance matrix of the Gaussian noise, which we suppose is the same for each elementary operation, following the representation in our optimization problem.

We can then compute the noisy leakage parameter  $\delta$  related to the  $\delta$ -noisy leakage model. As explained in Section 2.4, reducing the *idealized* noisy leakage model to the random probing model provides the leakage probability in the latter. More precisely, in order to achieve  $(\delta, \varepsilon)$ -security in the idealized noisy leakage model, an abstract circuit should achieve  $(p, \varepsilon)$ -security in the random probing model with  $p = \gamma \cdot \delta$  for some constant factor  $\gamma$  depending on the noisy leakage metric (e.g.  $\gamma = |\mathcal{X}|$  for the SD metric,  $\gamma = 1$  for the ARE metric). We describe the two main options:

- The original security reduction [DDF14] (DDF reduction) relies on the statistical distance between a (uniform) variable  $X$  and the same variable conditioned on its leakage  $Y$ :  $\delta = E_y[\text{SD}((X|Y = \mathbf{y}); X)]$ . When reducing to the random probing model, this value is multiplied by the size of the input space  $\mathcal{X}$  (i.e., the definition set of inputs  $\mathbf{x}$  of an elementary operation), hence losing tightness in the tolerated leakage rate through the reduction. In this SD metric,  $\gamma = |\mathcal{X}|$ .
- In a more recent work [PGMP19] (PGMP reduction), the authors express  $\delta$  using different noisiness metrics from the pointwise mutual information. From the security reduction viewpoint, the most interesting metric is the average relative error (ARE), a worst-case metric (just like the random probing model), contrary to the statistical distance, which is an average-case metric. When computing  $\delta = \text{ARE}(X; X|Y)$ , the reduction to the random probing model thus yields tighter results with  $\gamma = 1$  and thus  $p = \delta$ .

In order to estimate the noisy leakage parameter  $\delta$  in our methodology, we compute both ARE and SD metrics using the inferred leakage model to compare both reductions to the random probing model. Using the pointwise mutual information (Definition 7), we have

$$\begin{aligned} \text{ARE} &= \mathbb{E}_Y \max_{X=\mathbf{x}} \left| \frac{P[X = \mathbf{x}, Y = \mathbf{y}]}{P[X = \mathbf{x}] \cdot P[Y = \mathbf{y}]} - 1 \right| \\ &= \mathbb{E}_Y \max_{X=\mathbf{x}} \left| \frac{P[Y = \mathbf{y}|X = \mathbf{x}]}{\sum_{X=\mathbf{x}'} P[Y = \mathbf{y}|X = \mathbf{x}']} \cdot \frac{1}{P[X = \mathbf{x}]} - 1 \right| \end{aligned} \quad (8)$$

and

$$SD = \mathbb{E}_Y \frac{1}{2} \sum_{X=\mathbf{x}} \left| \frac{P[Y = \mathbf{y}|X = \mathbf{x}]}{\sum_{X=\mathbf{x}'} P[Y = \mathbf{y}|X = \mathbf{x}']} \cdot \frac{1}{P[X = \mathbf{x}]} - 1 \right| \quad (9)$$

From the equations above, we need to compute the sum of the conditional probabilities  $P[Y = \mathbf{y}|X = \mathbf{x}']$  for  $\mathbf{x}' \in \mathcal{X}$  for ARE and SD estimations. Then, in the case of ARE estimation, we need to find the maximal value of the expression given between  $|\cdot|$  in Equation 8 over the values taken by  $X$ . While in the case of the SD estimation, we compute a sum over the expression given between  $|\cdot|$  in Equation 9. Finally, we must compute the expected value for ARE and SD for  $Y$ .

**Computing the conditional distribution.** In order to estimate the conditional distribution  $P[Y = \mathbf{y}|X = \mathbf{x}]$  given a leakage trace and  $\mathbf{x}$ , we can use the leakage characterization computed earlier, since the conditional distribution is known to be expressed as

$$P[Y = \mathbf{y}|X = \mathbf{x}] = \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - d(\mathbf{x}))^T \Sigma^{-1}(\mathbf{y} - d(\mathbf{x}))\right)}{\sqrt{(2\pi)^n |\Sigma|}} \quad (10)$$

where  $n$  is the number of samples in  $\mathbf{y}$  [CRR03].

**Estimating the expected value.** Each sample point in the leakage distribution  $\mathcal{Y}$  is a continuous random variable over  $\mathbb{R}$ . Hence, the expected value  $\mathbb{E}_Y$  is computed as an integral. Instead of computing the integral, we use a Monte Carlo integration method to estimate the expected value. Namely, we draw several random leakage values to estimate the expected value. The ARE is then computed as

$$ARE = \frac{\sum_{Y=\mathbf{y}} \max_{X=\mathbf{x}} \left| \frac{P[Y = \mathbf{y}|X = \mathbf{x}]}{\sum_{X=\mathbf{x}'} P[Y = \mathbf{y}|X = \mathbf{x}']} \cdot \frac{1}{P[X = \mathbf{x}]} - 1 \right|}{k}, \quad (11)$$

and the SD is computed as

$$SD = \frac{\sum_{Y=\mathbf{y}} \sum_{X=\mathbf{x}} \left| \frac{P[Y = \mathbf{y}|X = \mathbf{x}]}{\sum_{X=\mathbf{x}'} P[Y = \mathbf{y}|X = \mathbf{x}']} \cdot \frac{1}{P[X = \mathbf{x}]} - 1 \right|}{2 \cdot k}. \quad (12)$$

In both equations,  $k$  denotes the number of leakage vectors  $\mathbf{y}$  drawn from the leakage distribution  $\mathcal{Y}$ . Each leakage vector  $\mathbf{y}$  is generated as  $\mathbf{y} = d(\mathbf{x}) + \boldsymbol{\varphi}$ , where  $\mathbf{x}$  is an input generated uniformly at random,  $d(\cdot)$  is the deterministic function for an elementary operation, and  $\boldsymbol{\varphi}$  is generated from the Gaussian distribution  $\mathcal{N}(\mathbf{0}, \Sigma)$  following the noise covariance  $\Sigma$ . The ARE being a worst-case metric, it is more expensive to estimate than the (average-case) SD metric, which is the price to pay for the tighter reductions it enables.

Thanks to the law of large numbers, we know that as  $k$  approaches infinity, this estimation converges to the expected value  $\mathbb{E}_Y$ . We show in Section 4 in our experimental results that the convergence curve reaches a plateau after some number of samples, which corresponds to the value of  $\mathbb{E}_Y$ .

### 3.6 Step 6: Compiling the Cryptographic Implementation

At this stage, we have estimated the leakage parameter of each isolated noise-independent elementary operation. As defined in Section 3.5, this parameter can be computed using the SD or ARE metric. We obtain an equivalent leakage probability  $p$  in the random probing

model in both cases by applying the reduction. As already mentioned, the reduction is tighter in the case of ARE, where the leakage probability in the random probing model can be directly set to the estimated noisy leakage parameter.

**Different vs. maximum leakage probabilities.** As we might obtain different noisy leakage parameters  $\delta_i$  for the different elementary operations, leading to different leakage probabilities in the random probing model, we consider that all the gates leak with the maximum probability corresponding to maximum noisy metric  $\delta = \max_i \delta_i$ . A random probing secure circuit with maximum leakage probability is straightforwardly random probing secure with different leakage probabilities.

**Gate vs. wire leakage model.** In our characterization, we rely on the leakage of the elementary operations abstracted as gates in a circuit. Meanwhile, most random probing secure constructions suppose leakage on wires instead. Our methodology applies an additional transition from the gate to the wire leakage model to circumvent this issue. As proven in Lemma 1, we can reduce the security of a circuit in the gate random probing model with leakage probability  $p$ , to the wire random probing model with leakage probability  $\sqrt{p}$ , assuming that each gate has at most two inputs.

**Compiling a cryptographic implementation.** The final step consists of a two-stage compilation process applied to the input abstract circuit  $C \in \mathbb{C}$  representing the target cryptographic implementation.

- One first applies the random probing secure compiler which, for the obtained leakage probability  $p$  and the target security level  $\varepsilon = 2^{-\lambda}$ , transforms  $C$  into a functionally equivalent randomized circuit  $\widehat{C}$  achieving  $(p, \varepsilon)$ -random probing security.
- One then serializes  $\widehat{C}$  into a physical implementation, making a sequence of calls to the (whitened) elementary operation routines on the target device. Each elementary operation in the sequence corresponds to a gate in  $\widehat{C}$  whose output is written in a fresh memory cell. The circuit wiring is hard-coded in the pointer arguments passed to the successive calls to the elementary operation routines.

The obtained physical implementation achieves  $\lambda$  bits of side-channel security for the target side-channel acquisition tool and physical device, and under relaxed or empirically verified physical assumptions. The overall process and provable security guaranty are wrapped up in Section 3.7.

### 3.7 Security Proof

After describing our methodology, we aim to state our main result.

**Theorem 1.** *Let  $C$  be an abstract circuit in the abstract circuit family  $\mathbb{C}$  with  $\mathbb{C} = (\mathcal{V}, \mathcal{G})$ . Let  $\mathcal{D}$  be a target device and  $\mathcal{A}$  a target side-channel acquisition tool, for which the six-step methodology described in Section 3 is applied to  $\mathcal{D}$  and  $C$ . Let us assume that:*

1. *The data isolation effectively ensures that the deterministic signal can be expressed as a sum  $\sum_{i=1}^k d_i(a_i, b_i)$ ;*
2. *The leakage characterization yields the exact leakage distribution (i.e., the exact deterministic functions  $\{d_i\}$  and covariance matrix  $\Sigma$ ).*



Further, if  $\hat{C}$  is a  $(\sqrt{\gamma\delta}, \varepsilon)$ -random probing secure circuit functionally equivalent to  $C$  where  $\delta$  is the noisy leakage parameter computed in the fifth step of the methodology and  $\gamma$  is the associated reduction constant, then there exists a simulator  $\mathcal{S}$  such that

$$\mathcal{S}(C) \approx_\varepsilon \mathcal{A}(\mathcal{D}, \hat{C})$$

where  $\approx_\varepsilon$  statistical distance bounded by  $\varepsilon$ .

While the ideal data isolation and leakage characterize assumptions might not be perfectly met in practice, they can be naturally relaxed. We thus assume that an adversary cannot effectively exploit the approximation error between this ideal world and the actual leakage to increase the advantage beyond  $\varepsilon$ .

*Proof.* Figure 3 gives a specific view of our characterization phases with the leakage assumptions, we next sketch how these fit together to provide the guarantees of [Theorem 1](#). At a high level, our proof builds a sequence of leakage simulators, working backwards w.r.t. the steps of [Figure 3](#): starting from the most abstract leakage model, then each simulator takes the output of the previous simulator in the sequence, and adapts the simulated leakage to the next leakage model, ending with physical leakage.

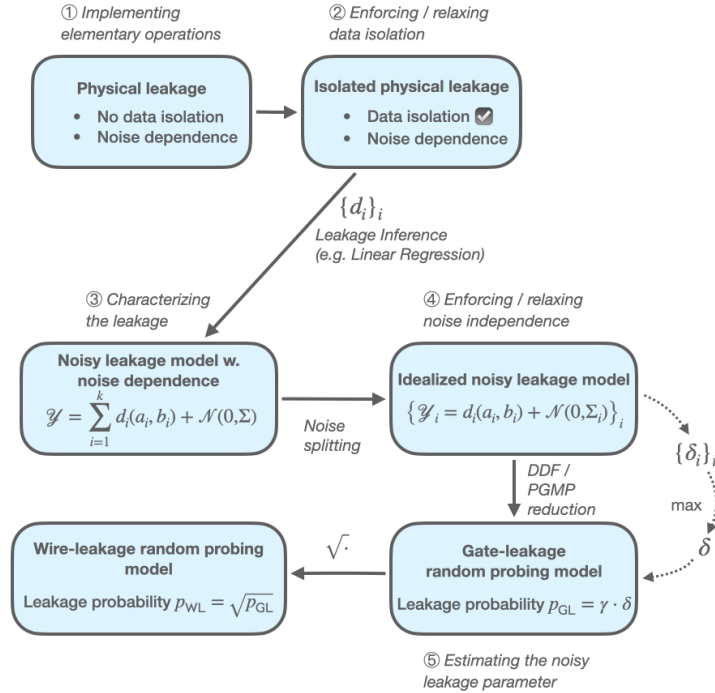


Figure 3: Illustration of the characterization phase with the leakage assumptions

As a starting point, we have  $\mathcal{S}_{WL}$ , a  $\varepsilon$ -close simulator for  $\hat{C}$  in the  $\sqrt{\gamma\delta}$ -wire random probing model. Next, using [Lemma 1](#), we get  $\mathcal{S}_{GL}$ , a  $\varepsilon$ -close simulator for  $\hat{C}$  in the  $\gamma\delta$ -gate random probing model. Moving to the idealized noisy leakage model, we have  $\mathcal{S}_{INL}$ , a  $\varepsilon$ -close simulator for the idealized noisy leakage model. When using the SD metric,  $\gamma = |\mathcal{X}|$  and  $\mathcal{S}_{INL}$  is derived from  $\mathcal{S}_{GL}$  using the DDF reduction [[DDF14](#)]. Otherwise, we use the ARE with  $\gamma = 1$ , and the simulator is built by the DDF reduction [[PGMP19](#)].

Let us now discuss the (perfect) simulation of the physical leakage from the ideal noisy leakage. The noisy leakage with noise dependence can be perfectly simulated by summing all the components of the ideal noisy leakage, thanks to [Equation 7](#) and the construction of

$\Sigma_i$  such that  $\Sigma = \sum_{i \in [k]} \Sigma_i$ . We conclude the proof by remarking that the two hypotheses of the theorem imply that the noisy leakage with noise dependence is equal to the actual physical leakage  $\mathcal{A}(\mathcal{D}, \hat{C})$ .  $\square$

## 4 Application of the Methodology: Procedures and Case Study

In order to validate our methodology, we conducted tests on an STM32F3 MCU using NewAE’s ChipWhisperer-Lite CW1173 board, which is based on an ARM Cortex-M4 processor. We first describe how we implemented elementary operations and our whitening approach. We then introduce and apply a novel test to verify data isolation (Assumption 1, Step 2). Next, we conducted a leakage characterization for the operations, computed the device’s noise covariance, and estimated the noisy leakage parameters. We also introduce noise splitting procedures that implement the relaxation of the noise independence (Assumption 2, Step 4). Our findings revealed low noise levels on the device, rendering it unsuitable for masking purposes, thus highlighting the need for secure hardware offering sufficient noise levels to attain robust provable security. Finally, we showcase an application of our methodology on a masked AES implementation. Overall, although the realizations of some steps are basic proof-of-concept versions that require further refinement, when considered alongside our contributions in Section 3, they collectively constitute a practical instantiation of our overall methodology.

### 4.1 Step 1: Implementing the Operations

We implement operations as routines described in Section 3.1. Since the cost of the data isolation test is quadratic in the number of elementary operations, we limit ourselves to four elementary operations for this proof-of-concept: 8-bit XOR, 8-bit AND, 8-bit Right Shift, and 8-bit Left Shift. These operations would be enough to implement a masked bit-sliced AES at any chosen order for instance. Of course, more elementary operations would make the implementation more efficient but would imply a higher cost in side-channel characterization.

In our example, each elementary operation relates to a single ARM Cortex-M4 instruction, simplifying the analysis (listing the intermediate variables of each operation is trivial) while not mandatory for the methodology. We implement the four operations in assembly as shown in Figure 4.

```

xor_func:
    ldr r0, [r0]
    ldr r1, [r1]
    eor r0, r1 r0
    str r0, [r2]

and_func:
    ldr r0, [r0]
    ldr r1, [r1]
    and r0, r1 r0
    str r0, [r2]

left_shift_func:
    ldr r0, [r0]
    mov r0, r0, LSL 1
    str r0, [r1]

right_shift_func:
    ldr r0, [r0]
    mov r0, r0, LSR 1
    str r0, [r1]

void whitening(void) {
    xor_func(a1Ptr, b1Ptr, c1Ptr);
    xor_func(a2Ptr, b2Ptr, c2Ptr);
    xor_func(a3Ptr, b3Ptr, c3Ptr);
}
    
```

Figure 4: Elementary operations (xor, and, left shift, right shift) and whitening as implemented on the STM32F3 MCU.

## 4.2 Step 2: Data Isolation

### 4.2.1 Procedure for Testing Data Isolation

Masking security proofs require independence between the leakage of all operations. However, enforcing and testing this independence assumption is challenging, leading to another approach in practice based on the test vector leakage assessment (TVLA) [SM16]. This approach verifies the statistical security order (i.e., the smallest statistical moment that leaks) of a masked implementation by detecting secret-dependencies in the statistical moments of the leakage [DFS19]. While dependence in the moment corresponding to the security order is expected due to dependence in inputs of the leakage functions (e.g., all the shares of a value), lower-order moments in a threshold-probing secure implementation with independent leakage functions are independent of the secret. This test can indeed detect typical leakage independence violations due to physical defaults like glitches [MPG05, MPO05] or transitions [CGP<sup>+</sup>12, BGG<sup>+</sup>14] when they lead to a security order reduction. Due to the difficulty in enforcing strict independence in the implementation and to verify it, a commonly accepted relaxation is to ensure that if there are detectable lower-order leakages, they are of significantly lower amplitude than those at the target security order [DFS19].

While this heuristic works reasonably well in practice, it has two significant limitations. First, by verifying only a security order, it cannot detect leakage dependence issues that would result in other kinds of weaknesses than security order reductions, e.g., easing horizontal attacks. Second, while this approach is always applicable in theory, it requires testing all the mixed statistical moments corresponding to all the tuples of leakage points in the traces of a masked implementation [BDF<sup>+</sup>17], and it is therefore computationally impractical (it scales exponentially with the length of the trace) at large security orders (even the second order can be challenging).

Therefore, we propose another approach with much improved practical efficiency, which can detect leakage dependencies that do not reduce the security order. Our approach is based on testing the independence between the leakage of two consecutive operations. Then, we use an argument based on physics to extend the result of this test to long sequences of operations.

**Leakage independence for adjacent operations.** Let us consider two operations  $\text{op}_1$  and  $\text{op}_2$  with their respective inputs  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . We assume that these two operations are executed sequentially, giving rise to a leakage trace  $Y$ .

We say that the operations have independent leakage if

$$Y(\mathbf{x}_1, \mathbf{x}_2) = d_1(\mathbf{x}_1) + d_2(\mathbf{x}_2) + N \quad (13)$$

where  $d_1$  and  $d_2$  are the deterministic functions (like in [SLP05]) and  $N$  follows a Gaussian noise distribution  $\mathcal{N}$ . This definition indeed ensures independence, as it is possible to decompose  $N$  into two independent Gaussian noises  $N_1$  and  $N_2$ , giving  $Y = (d_1(\mathbf{x}_1) + N_1) + (d_2(\mathbf{x}_2) + N_2)$ . Despite the sequential execution context, we cannot assume that the leakage is a sequential combination of the leakage of the operations (e.g.,  $Y = (d_1(\mathbf{x}_1) + N_1, d_2(\mathbf{x}_2) + N_2)$ ) due to data dependency effect between successive operations (e.g. transitions in CPU buses/registers) and low-pass filtering in the measured circuit or acquisition chain.

We use the following statistical test to verify that Equation 13 holds:

1. Generate uniformly at random two pairs  $\alpha$  and  $\beta$  of inputs  $(\mathbf{x}_1^\alpha, \mathbf{x}_2^\alpha)$  and  $(\mathbf{x}_1^\beta, \mathbf{x}_2^\beta)$  (similarly to *fixed-vs-fixed* leakage assessment).
2. Acquire a set  $T_{(\alpha,0)}$  of  $\ell$  traces corresponding to executing the operations with the inputs set to  $(\mathbf{x}_1^\alpha, \mathbf{0})$  and compute the average trace  $\bar{T}_{(\alpha,0)}$ .
3. Do the same thing for  $T_{(\beta,0)}$  with inputs  $(\mathbf{x}_1^\beta, \mathbf{0})$ ,  $T_{(0,\alpha)}$  with inputs  $(\mathbf{0}, \mathbf{x}_2^\alpha)$ ,  $T_{(0,\beta)}$  with inputs  $(\mathbf{0}, \mathbf{x}_2^\beta)$ ,  $T_{(\alpha,\alpha)}$  with inputs  $(\mathbf{x}_1^\alpha, \mathbf{x}_2^\alpha)$  and  $T_{(\beta,\beta)}$  with inputs  $(\mathbf{x}_1^\beta, \mathbf{x}_2^\beta)$ . We hence acquire in total  $6 \cdot \ell$  traces.
4. Compute the following

$$T'_{(\alpha,\alpha)} = T_{(\alpha,\alpha)} - \bar{T}_{(\alpha,0)} - \bar{T}_{(0,\alpha)} \quad T'_{(\beta,\beta)} = T_{(\beta,\beta)} - \bar{T}_{(\beta,0)} - \bar{T}_{(0,\beta)},$$

(i.e., from each trace in  $T_{(\alpha,\alpha)}$  (resp.  $T_{(\beta,\beta)}$ ), we subtract  $\bar{T}_{(\alpha,0)} + \bar{T}_{(0,\alpha)}$  (resp.  $\bar{T}_{(\beta,0)} + \bar{T}_{(0,\beta)}$ )).

5. Compute the statistical mean equality test on the sets  $T'_{(\alpha,\alpha)}$  and  $T'_{(\beta,\beta)}$  as

$$t = \frac{\bar{T}'_{(\alpha,\alpha)} - \bar{T}'_{(\beta,\beta)}}{\sqrt{\frac{s_{(\alpha,\alpha)}^2 + s_{(0,\alpha)}^2 + s_{(\alpha,0)}^2 + s_{(\beta,\beta)}^2 + s_{(\beta,0)}^2 + s_{(0,\beta)}^2}{\ell}}} \quad (14)$$

where  $s_{(i,j)}^2$  is the unbiased estimator for the population variance of  $T_{(i,j)}$ . If no significant difference pops up (e.g.,  $|t| < 4.5^5$ ), conclude that Equation 13 holds (at least, we could not contradict it).

The motivation for this test is that under the null hypothesis (i.e., Equation 13 holds),  $\bar{T}_{(\alpha,0)}$  converges to  $d_1(\mathbf{x}_1^\alpha) + d_2(\mathbf{0})$  and  $\bar{T}_{(0,\alpha)}$  to  $d_1(\mathbf{0}) + d_2(\mathbf{x}_2^\alpha)$ . Therefore, the distribution of  $T'_{(\alpha,\alpha)}$  converges to the distribution of  $N + d_1(\mathbf{0}, \mathbf{0}) + d_2(\mathbf{0}, \mathbf{0})$ , and likewise for  $T'_{(\beta,\beta)}$ . We finally compute  $t$  such that, under the null hypothesis, it follows a standard normal distribution.

Since the noise comes from physical, electronic phenomena, its Gaussian distribution and independence on the data is a reasonable assumption. However, in case of doubt, further statistical tests can be performed. For instance, a test of Gaussianity of  $Y(\mathbf{x}_1, \mathbf{x}_2)$  can be performed for fixed  $(\mathbf{x}_1, \mathbf{x}_2)$ , as well as an equality test for the (co)variance of  $Y(\mathbf{x}_1, \mathbf{x}_2)$  across different  $(\mathbf{x}_1, \mathbf{x}_2)$ .

Finally, it is worth clarifying that a statistical test can demonstrate the inability to detect dependencies with a specific number of measurements but does not prove independence. Nonetheless, if the test fails to identify a dependency for  $\ell$  traces, it reasonably suggests

<sup>5</sup>The 4.5 threshold is given for simplicity, a better approach would be to adapt the  $t$ -score threshold with respect to the length of the traces (e.g. as proposed in [DZD<sup>+</sup>17]).

that this dependency is unlikely to be exploited for an attack with significantly fewer than  $\ell$  traces. In contrast to traditional higher-order TVLA, our test offers greater statistical power: being a first-order test, it exhibits lower sensitivity to noise, regardless of the masking order under consideration.

**Independence in longer operation sequences.** Let us now discuss the dependencies between operations that are not adjacent. In this section, we argue that, given an understanding of the evaluated processor’s architecture and within reasonable physical assumptions, the absence of dependency between adjacent operations ensures the independence of leakage for non-adjacent operations.

Considering the processor (excluding the memory), we first assume that the “core” leakage for any clock cycle is a function of all the state of the processor (and the input data, e.g., on the memory bus).<sup>6</sup> This “core” leakage may then get filtered (i.e., undergo a linear transformation) before it is measured (*linear physics* hypothesis, denoted LP). Next, given a sufficiently simple processor, we may assume that when the processor executes  $m$  well-chosen non-branching/conditional instructions, the microarchitectural state of the processor does not depend on the state computed by the first of these operations (provided that the other operations do not also compute this state, and a few cycles after the last instruction retires) — *m-state-erasing* (denoted *m-SE*) hypothesis. Concretely, for an elementary processor whose state is only the architectural state, the 2-SE hypothesis is satisfied. For more complex processors,  $m$  might be larger (or even not exist). For a simple in-order processor, *m-SE* with  $m$  close to the pipeline depth appears as a reasonable assumption — this assumption can easily be verified if the design of the processor is known.

Our operations all follow the same structure: load the operands in two registers, perform a logic instruction, and store the result (always using the same registers). Then, between two operations, we clear the register state, then we execute a sequence of “cleaning” operations that operate on constant public data (i.e., whitening operations). The *m-SE* hypothesis, combined with LP, implies independent leakage when  $m - 1$  cleanings separate the operations. Our two operations test presented in the next section is a way to validate the hypotheses (and the  $m$  parameter).

Finally, regarding the memory leakage, it is reasonable to assume LP for the static leakage from the memory cells and *m-SE* for the remaining logic.

Let us conclude this section by remarking that if an open-source processor is used, the analysis of leakage independence is greatly simplified. Indeed, as the hardware is known, we may apply the robust-probing leakage model to instructions sequences (or to verify the *m-SC* hypothesis). Providing hardware support for micro-architectural state “cleaning” through a dedicated fence instruction may also solution to improve code simplicity and performance, with very low hardware cost, in the spirit of masking-supporting instruction set extensions [GMPP20]. The software-level overhead can be further reduced by integrating the isolation semantics inside the data-processing instructions themselves, instead of needing an explicit fence. For example, [CPW24] proposes to add instruction prefixes or processor control registers leading to semantics “preventing any architectural and micro-architectural overwriting”.

#### 4.2.2 Data Isolation on the STM32F3

To perform our data isolation test, we need to capture the side-channel execution traces of two consecutive elementary operations separated by a whitening process and use the test to validate or not data isolation between the two operations. This approach must be re-iterated for all combinations of two successive elementary operations.

---

<sup>6</sup>This assumption is conservative since it is weaker than the well-studied robust probing model for hardware leakage [FGM<sup>+</sup>18].

The whitening process does not have to be the same for all pairs of elementary operations, but for our operations selection, the acquisition setup, and the chip, a single whitening process allows us to pass all tests: three consecutive `xor` operations with constant public inputs<sup>7</sup>. The corresponding assembly code is shown in Figure 4, where `{aiPtr, biPtr, ciPtr}` for  $i \in \{1, 2, 3\}$  are memory pointers to the two constant operands `{ai, bi}` and the memory location to store the result `ci`.

The test procedure is applied as follows:

- For each pair of elementary operations, the target code is the following sequence of calls:

```
whitening();
operation1(a1Ptr, b1Ptr, c1Ptr);
whitening();
operation2(a2Ptr, b2Ptr, c2Ptr);
```

The inputs are pre-generated and loaded in memory once for all before iterating on the target code.

- Triggers surround the target code to ease the trace acquisition.
- For randomly chosen values  $(\mathbf{x}_1^\alpha, \mathbf{x}_2^\alpha), (\mathbf{x}_1^\beta, \mathbf{x}_2^\beta)$ <sup>8</sup> we collect  $10^6$  traces for each set  $T_{(\alpha,\alpha)}, T_{(\alpha,0)}, T_{(0,\alpha)}, T_{(\beta,\beta)}, T_{(\beta,0)}, T_{(0,\beta)}$ , for a total of  $6 \times 10^6$  traces.
- From the sets of traces, we test if Equation (13) holds.

The above process is applied for each pair of elementary operations and iterated 2-3 times for different values of  $(\mathbf{x}_1^\alpha, \mathbf{x}_2^\alpha), (\mathbf{x}_1^\beta, \mathbf{x}_2^\beta)$ . Figure 5 illustrates the test result for the pair of operations (`xor`, `and`) and a single choice of fixed inputs. Figure 5b (resp. Figure 5c) illustrates the captured leakage (through the T-Test) of the computation of `xor` (resp. `and`) and the manipulation of its inputs/outputs. Namely, Figure 5b shows high T-test values at the moment of the execution of `xor` with a residual leakage slowly decreasing afterward. While Figure 5c shows high T-test values later at the moment of the execution `and`, and no leakage is detected before, ensuring that the inputs of `and` were not manipulated before the execution of the operation. Then, Figure 5d illustrates the captured leakage of both `xor` and `and` simultaneously, including the individual leakages of `xor` and `and`. Figure 5e is the result of our proposed test: it represents the captured leakage of both `xor` and `and` simultaneously while individual leakages of `xor` and `and` are removed. The T-Test results show that the data isolation process (whitening function) successfully removes the combined leakage of `xor` and `and`.

### 4.3 Step 3: Linear Regression as the Leakage Function

We use the experimental setting described in Section 4.2, where we consider operations with at most 2 inputs of 8 bits. We start by inferring each elementary operation’s deterministic part of the leakage function separately. We use the linear regression with a specific choice of basis of functions  $\mathcal{H} = \{h_1, \dots, h_m\}$ . The result of the linear regression is the set of  $\{\alpha_i\}_i$  such that, for all inputs  $(a, b)$  of the selected elementary operation,

$$d(a, b) = \sum_{i=1}^m \alpha_i h_i(a, b) \quad (15)$$

holds. In order to capture the leakage function fully, we construct the basis of functions as follows (where  $n$  is the bit-length of the inputs  $a$  and  $b$ , here  $n = 8$ ):

<sup>7</sup>Using only two consecutive `xor` operations was not enough to pass all tests for all pairs of elementary operations.

<sup>8</sup>where  $\mathbf{x}_i$  (resp.  $\mathbf{x}'_i$ ) contains the two inputs of `operation $i$`

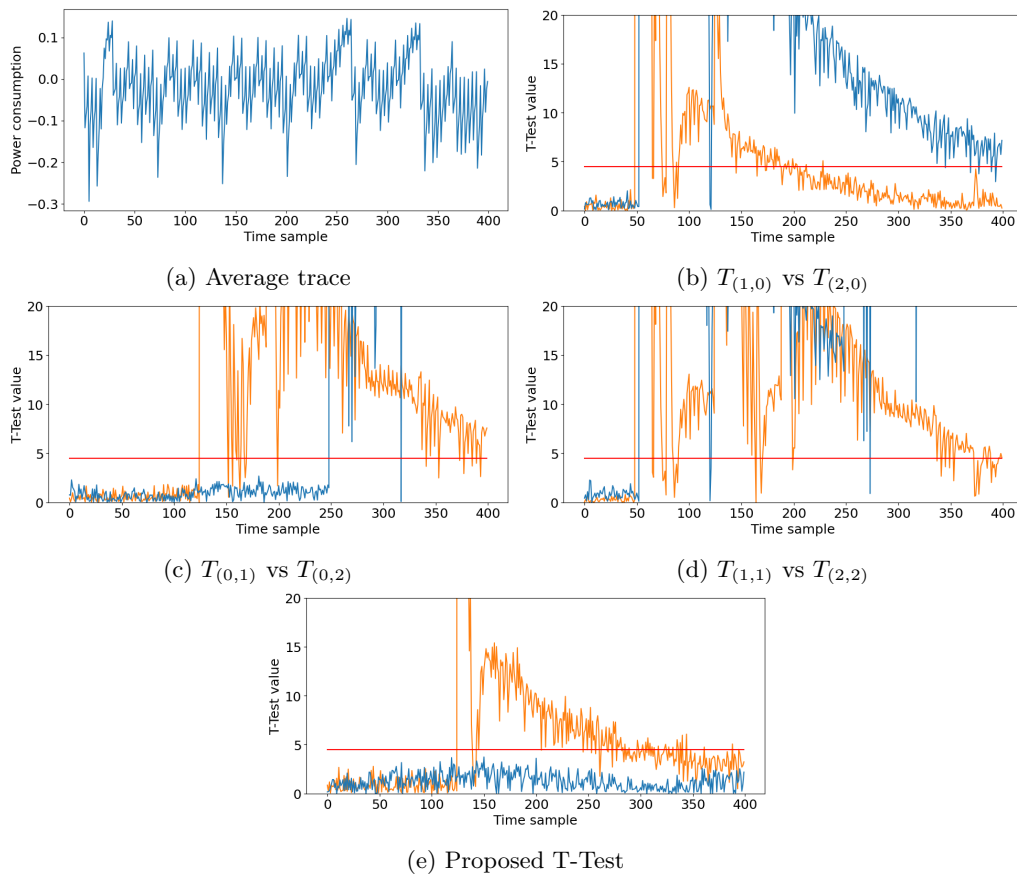


Figure 5: Data isolation test: curves with whitening in blue and without whitening in orange.

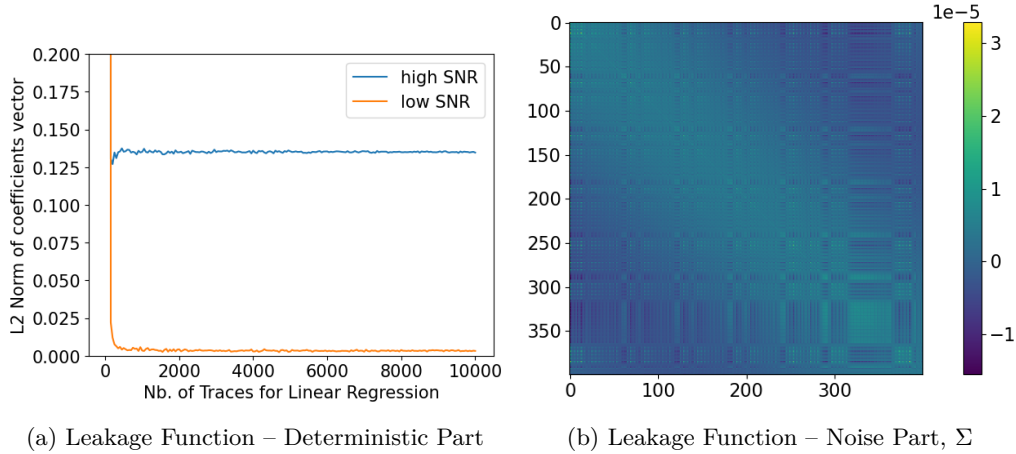


Figure 6: Linear Regression of the Leakage Function.

- $h_0(a, b) = 1$
- for all  $i \in [1 \cdots n]$ ,  $h_i(a, b)$  returns the  $i$ th bit of  $a$
- for all  $i \in [1 \cdots n]$ ,  $h_{n+i}(a, b)$  returns the  $i$ th bit of  $b$
- for all  $(i, j) \in [1 \cdots 2n]^2$ , with  $i < j$ ,  $h_{i,j}(a, b)$  returns  $h_i(a, b) \oplus h_j(a, b)$

This gives a total of  $m = 1 + 2 \cdot n + n \cdot (2 \cdot n - 1) = 2 \cdot n^2 + n + 1$  functions in the function basis  $\mathcal{H}$ .

For each elementary operation, the target code is the following concatenation

```
operation(aPtr, bPtr, cPtr);
whitening();
```

surrounded by triggers to ease the trace acquisition (similarly to [Subsubsection 4.2.2](#)). The inputs to the operations are randomly generated (using a Mersenne Twister RNG) on the chip before each execution of the target code. Then, we collect  $10^6$  traces in a single set and apply linear regression over the function basis  $\mathcal{H}$  on each independent time sample. The output is the set of vectors  $\{\alpha_i\}_i$ .

Figure 6a illustrates the convergence of the L2 norm of the coefficient vectors at two different time samples for the `xor` operation. We consider one vector where the SNR ratio is high (i.e., more information leakage) and one where the ratio is low. We can see that for both vectors, the L2 norm converges from a few hundred traces, meaning that the linear regression can quickly estimate the coefficients of the deterministic part of the function. This behavior can further be explained by the low noise in the leakage depicted through the covariance matrix in Figure 6b. We compute this covariance matrix on the same time samples of the operation as for the deterministic function. The covariance matrix clearly shows low noise levels, which implies more information leakage. We can also observe through Figure 6a that for a sample with high SNR, the coefficients converge to more significant values than for a sample with low SNR, which gives more confidence about the results, since for samples with more information leakage, the deterministic functions should have more weight than when there is not much information leakage.

## 4.4 Step 4: Implementing Noise Splitting

### 4.4.1 Noise Splitting Procedures

A necessary physical assumption in the noisy leakage model is noise independence as described in [Assumption 2](#). Since enforcing this assumption is hard to achieve, we propose a way to relax it instead, as discussed in [Section 3.4](#).



Consider a sequence of  $k$  operations and the corresponding leakage trace  $Y$ . Assuming that the noise is additive, we have the decomposition  $Y = S + N$  where  $S$  is the signal (typically a deterministic function of the input data as presented in the previous sections), and  $N$  is the data-independent noise. Further, thanks to the data isolation test of the previous section, we know that the signal can be rewritten as  $S = \sum_{i=1}^k S_i$ , where  $S_i$  is the leakage caused by operation  $i$ . We aim to decompose the noise into a sum of  $k + 1$  independent contributions (one for each operation and a “leftover” one)  $N = \sum_{i=0}^k N_i$ . Assuming that the noise contributions  $N_i$  are Gaussian, we only have to ensure that they are not correlated to ensure independence. This gives us a decomposition of the leakage signal  $Y = \sum_{i=0}^k Y_i$  where  $Y_0 = N_0$  and  $Y_i = S_i + N_i$  for  $i \neq 0$ , which ensures signal and noise independence between the components.

We argue that an implementation secure against an adversary with access to  $\{Y_i\}_{i \in \{0, \dots, k\}}$  is also secure against an adversary with access to the global leakage trace  $Y$ . The former is stronger than the latter adversary since each sample in  $Y$  can be obtained by the sum of the corresponding samples in  $\{Y_i\}_{i \in \{0, \dots, k\}}$ . In our methodology, we suggest computing the noisy leakage parameter based on the split leakages  $Y_i$ . This approach results in a weaker noisy leakage parameter compared to that computed from  $Y$ , as the noise  $N$  is divided into  $k + 1$  smaller components. Consequently, each independent operation inherently leaks more information. Hence, an abstract circuit secure against an adversary with access to the split leakage is also secure against an adversary with access to the global leakage  $Y$ .

We will present two solutions to the above decomposition problem in the following. We first discuss a trivial solution, which has the advantage of being easily applicable but induces a loss in the security level as the size of the implementation grows. Then, we express the decomposition as an optimization problem that better scales with the size of the circuit but is more challenging to solve. We propose a direct solution to the optimization and leave the question of optimally solving it as an open problem.

**Trivial Solution.** We can perform a trivial split of the noise described above. Namely, for a sequence of  $k$  operations, we can split the Gaussian noise  $N = \mathcal{N}(\mathbf{0}, \Sigma)$  such that  $N_0 = \mathbf{0}$  and  $N_i = \mathcal{N}(\mathbf{0}, (1/k) \cdot \Sigma)$  for  $i \in \{1, \dots, k\}$ . This decomposition ensures that the leakage  $Y$  can be expressed as a sum of  $Y_i$  with  $Y_0 = \mathbf{0}$  and  $Y_i = S_i + \mathcal{N}(\mathbf{0}, (1/k) \cdot \Sigma)$ , with noise and signal independence.

Meanwhile, the above decomposition scales poorly with the size of the circuit. As the number of operations grows, the noise occurring on each operation decreases, leading to lower security in the noisy leakage model when applying the reduction after the relaxation (i.e., increasing the ARE leakage).

Hence, the noise decomposition, in addition to ensuring the independence of the components, should minimize this leakage. As explained in [Section 3.6](#), the chosen ARE (or SD) metric for the noisy leakage model is the maximum among all operations executed. Hence, the chosen decomposition should balance the noise on all operations and scale with the number of operations executed.

**Better Noise Splitting.** We propose a better way to split the noise taking advantage of a relaxed noise independence assumption. For a sequence of  $k$  elementary operations, we can split the leakage trace into  $k$  sub-traces, all of the same size and including the time samples of one elementary operation each. We call distance  $d$  between two sub-traces the number of operations (or sub-traces) between them during the computation’s sequence (for example, two consecutive sub-traces have distance  $d = 1$ ). For the sake of simplicity, we assume that all the sub-traces have identical noise distributions and that the dependence (i.e., the covariance matrix) between the noises of two sub-traces solely depend on their

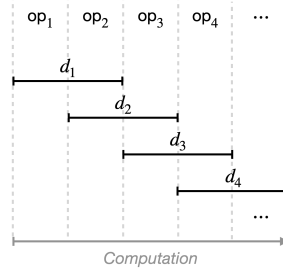


Figure 7: Data dependency spanning.

distance  $d$ .<sup>9</sup> This means that each operation’s noise covariance matrix is the same denoted  $\Sigma'_0$ , and that the covariance matrix between two sub-traces with distance  $d$  is the same along the computation denoted  $\Sigma'_d$ . We then formulate the following relaxed noise-independence assumption.

**Assumption 3** (Relaxed noise independence assumption). *There exists  $d_{max} \in [0, k)$  such that the sub-traces with a distance  $d > d_{max}$  have null covariance:  $\Sigma'_d = 0 \forall d > d_{max}$ .*

Intuitively, the above assumption captures the expectation that, after some delay, the noise in an operation sub-trace is fully independent of the noise in an earlier operation sub-trace. While we introduce it as a “relaxed assumption”, we stress that it is without loss of generality since there always exists such a  $d_{max}$ . In particular, the case  $d_{max} = k - 1$  captures that the independence between the noise of two operations is never reached.

Under this relaxed noise independence assumption, the global covariance matrix for  $k$  operations has the following structure (assuming  $d_{max} = 1$ ):

$$\Sigma = \begin{pmatrix} \Sigma'_0 & \Sigma'_1 & & & \\ \Sigma'_1 & \Sigma'_0 & \Sigma'_1 & & \\ & \Sigma'_1 & \Sigma'_0 & \Sigma'_1 & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \quad (16)$$

We introduce another parameter, which we call the *data-dependency depth*,  $\ell_{max}$ . This is the number of sub-traces over which the data dependency of an elementary operation spans. Specifically, the deterministic part of the leakage  $d_i$  of the  $i$ -th operation is non-zero for samples spanning on sub-traces  $i, i + 1, \dots, i + \ell_{max}$ . This is represented in Figure 7 for  $\ell_{max} = 1$ .

We now explain how a better splitting of the noise can be achieved, first by assuming  $d_{max} = \ell_{max} = 1$  (and generalize later). Consider a split of the leakage in three as:

$$\begin{cases} L_1 := (S_1 + S_4 + \dots) + N_1 \\ L_2 := (S_2 + S_5 + \dots) + N_2 \\ L_3 := (S_3 + S_6 + \dots) + N_3 \end{cases}$$

where  $S_i = d_i(\mathbf{x}_i)$  denotes the signal of the  $i$ -th operation, which spans over time samples as represented on Figure 7, and with  $N_i \sim \mathcal{N}(0, (1/3)\Sigma)$  so that  $N_1 + N_2 + N_3 = N \sim \mathcal{N}(0, \Sigma)$ . We have that  $(L_1 + L_2 + L_3) \sim Y$ , the global leakage. Let us now consider  $(1/3)\Sigma = AA^T$  the Cholesky decomposition of the global covariance matrix (scaled by 1/3), so that the

<sup>9</sup>This assumption is not strictly necessary to the application of our method but makes the presentation much simpler.

$N_i$  noises follow a distribution  $N_i \sim A \cdot X_i$  with  $X_i \sim \mathcal{N}(0, I)$ , for  $I$  the identity matrix. We have that  $A^{-1}$  has the same zero matrix blocks as  $\Sigma$  (see Equation 16). Namely, it can be written as:

$$A^{-1} = \begin{pmatrix} B_0 & B_1^T & & & \\ B_1 & B_0 & B_1^T & & \\ & B_1 & B_0 & B_1^T & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \quad (17)$$

for some matrices  $B_0, B_1$  (with  $B_0$  being symmetric). Then we get

$$\begin{aligned} A^{-1} \cdot L_1 &:= (S'_1 + S'_4 + \dots) + X_1 \\ A^{-1} \cdot L_2 &:= (S'_2 + S'_5 + \dots) + X_2 \\ A^{-1} \cdot L_3 &:= (S'_3 + S'_6 + \dots) + X_3 \end{aligned} \quad (18)$$

with  $S'_i = A^{-1} \cdot S_i$ . One can then check that for each of the three leakages,  $L_1, L_2$ , and  $L_3$ , the successive signals  $S'_i, S'_{i+3}, \dots$  are strictly disjoint (meaning that they are non-zero over disjoint time samples). This is due to the structure of  $A^{-1}$  (see Equation 17) and the fact that each  $S_i$  spans over two sub-traces. Then the  $S'_i$  span over three sub-traces so that  $S'_i$  and  $S'_{i+3}$  are disjoint. Moreover, the normalized noises  $A^{-1} \cdot N_i = X_i \sim \mathcal{N}(0, I)$  can be trivially separated as

$$\begin{aligned} X_1 &= X'_1 + X'_4 + \dots \\ X_2 &= X'_2 + X'_5 + \dots \\ X_3 &= X'_3 + X'_6 + \dots \end{aligned} \quad (19)$$

such that the  $X'_i$  span the same time samples as the  $S'_i$ . We finally split the leakage in variables  $Y_i = A \cdot (S'_i + X'_i)$  which satisfy  $Y = \sum_{i=1}^k Y_i$ . In this splitted leakage, the amount of noise is scaled by a factor  $1/3$  compared to the factor  $1/k$  of the trivial solution.

The same reasoning applies to higher values of  $d_{max}$  and  $\ell_{max}$ . But instead of dividing the noise in 3 and scaling the covariance by factor  $1/3$ , one has to divide it in  $d_{max} + \ell_{max} + 1$  and hence scale the covariance by a factor  $1/(d_{max} + \ell_{max} + 1)$ . Depending on the noise dependency depth and data dependency depth, this might still be way better than a factor  $1/k$ .

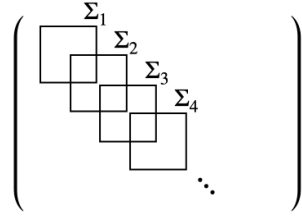
**Towards Optimal Noise Splitting.** While better than the trivial solution, the above method is non-optimal since it roughly splits the noise in  $d_{max} + \ell_{max} + 1$  regardless of the signals  $S_i$ . While the signal  $S_i$  may span over the  $(i + 1)$ -th sub-trace, it might be much weaker than on the  $i$ -th sub-trace and should receive a smaller amount of noise than the signal  $S_{i+1}$  on these time samples.

Once again, we state our optimization problem for  $d_{max} = \ell_{max} = 1$  but stress that it can be generalized to higher depths. Recall that we want to split the global covariance matrix into  $k + 1$  covariance matrices  $\Sigma_0, \dots, \Sigma_k$  such that

$$\Sigma = \Sigma_0 + \Sigma_1 + \dots + \Sigma_k \quad (20)$$

to split the leakage into  $n$  leakages:  $Y_i := S_i + N_i$  with  $N_i \sim \mathcal{N}(0, \Sigma_i)$ .

Given the data-dependency spanning (c.f. Figure 7),  $\Sigma_i$  is only required to span the same leakage samples as  $d_i$ . Then the (lowered) global covariance matrix  $\Sigma$  has the following structure:



From this structure, we observe that for an operation, say the  $i$ -th one, we need to split the covariance matrix  $\Sigma'_0$  between  $Y_i$  and  $Y_{i-1}$  (since  $d_{i-1}$  spans over time samples of the  $i$ -th operation). On the other hand,  $\Sigma'_1$  does not need to be split. Namely, defining  $\Sigma_i$ , for every  $i \in \{1, \dots, n\}$ , as the symmetric positive semi-definite matrix

$$\Sigma_i := \begin{pmatrix} \bar{\Sigma}_0^{(0)} & \bar{\Sigma}_1 \\ \bar{\Sigma}_1 & \bar{\Sigma}_0^{(1)} \end{pmatrix} \quad (21)$$

with

$$\bar{\Sigma}_0^{(0)} + \bar{\Sigma}_0^{(1)} \leq \Sigma'_0 \quad \text{and} \quad \bar{\Sigma}_1 \leq \Sigma'_1, \quad (22)$$

we ensure Equation 20, where by  $\Sigma' \leq \Sigma$  we mean that there exists a positive semi-definite matrix  $\Sigma''$  (i.e.,  $\Sigma''$  is a covariance matrix) such that  $\Sigma' + \Sigma'' = \Sigma$ .

Let  $\delta_i$  be a leakage metric corresponding to  $Y_i$ . Our optimization goal is

$$\min_{\bar{\Sigma}_0^{(0)}, \bar{\Sigma}_0^{(1)}, \bar{\Sigma}_1} \max \{\delta_i\}_i$$

under the constraints of Equation 22, and for  $\Sigma_i$  symmetric and positive semi-definite.

To sum up, under the assumptions stated above, we infer the leakage parameters which are the functions  $d_i$ , and the covariance matrices  $\Sigma'_0$  and  $\Sigma'_1$  and we look for a matrix  $\Sigma_i$  as defined in Equation 21 (in particular, a split of the  $\Sigma'_0$  matrix into  $\bar{\Sigma}_0^{(0)} + \bar{\Sigma}_0^{(1)}$ ) for which the maximal  $\delta_i$  is minimized.

**Choosing  $\delta_i$ .** Ideally, we would find the decomposition as the one that minimizes the SD or ARE leakage metric. Meanwhile, choosing metrics simpler to express can lead to optimization problems with simpler constraints, theoretically and efficiently solvable with current tools. For instance, we can choose our metric to be the multivariate SNR denoted  $\text{SNR}_i$  for the leakage  $Y_i$ , defined as the maximal eigenvalue of the matrix  $\Sigma_{d_i} \bar{\Sigma}_i^{-1}$ , where  $\Sigma_{d_i}$  is the covariance matrix of  $d_i(X)$ , for  $X$  uniform over  $\mathcal{X}$ . Then, our optimization goal becomes

$$\min_{\bar{\Sigma}_0^{(0)}, \bar{\Sigma}_0^{(1)}, \bar{\Sigma}_1} \max \{\text{SNR}_i\}_i$$

under the same constraints as earlier, which leads to a convex optimization problem. Minimizing the SNR ultimately leads to lowering the SD or ARE and therefore appears as a natural first step before solving the more general case. It is in line with our goal to exhibit a first complete connection between the theory and practice of the masking countermeasure, leaving the question of an optimized methodology relying on the best combination of metrics and proofs as an interesting direction for further research.

#### 4.4.2 Noise Splitting in Practice

Our experimental results on the ChipWhisperer show that the noise levels on the ChipWhisperer we use are very low. Namely, Figure 6 shows deterministic leakage of variance about  $1 \times 10^{-1}$  while the highest noise variance is about  $3 \times 10^{-5}$ , hence our measurements are essentially noise-free.

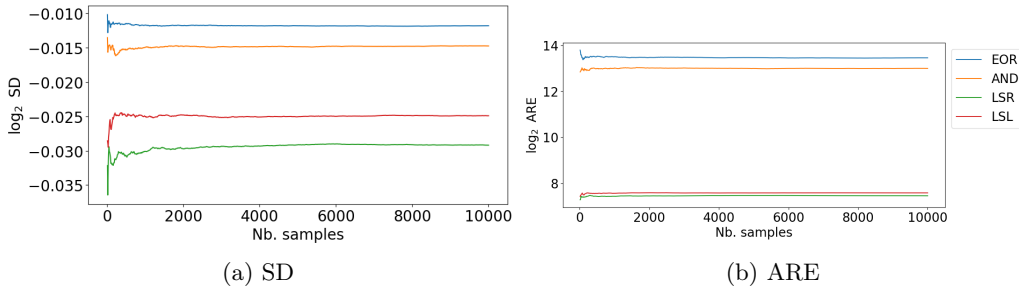


Figure 8: ARE and SD Monte-Carlo convergence as described in Section 3.5.

We nevertheless used the noise covariance matrix Figure 6b (computed from a set of traces with fixed input value for the same operation) to apply the relaxation from the previous section. The very low noise failed our attempts to apply the optimization problem of Section 4.4.1. In this case, we can apply the trivial noise decomposition from Section 4.4, making the security reduction work at the cost of decreasing the noise levels even further as the size of the circuit grows.

#### 4.5 Step 5: Estimating the Noisy Leakage Parameter

Still with the same experimental setting as Subsubsection 4.2.2, we consider operations with at most 2 inputs of 8 bits. Then, iterating through all possible values in  $\mathcal{X}$  for a given  $\mathbf{y}$  amounts to performing  $2^{16}$  iterations. An extra  $2^{16}$  iterations are performed to compute the max for ARE or the sum for SD. Indeed, iterating over all possible values in  $\mathcal{X}$  does not scale well when considering larger inputs. In such a case, other methods can be applied to make the computation tractable. For instance, one can use the nearest-neighbor-based approach from [CDSU23] to efficiently and quickly compute the conditional probabilities and to find the max over  $\mathbf{x} \in \mathcal{X}$  in the case of ARE. We leave the computation of the noisiness metric for larger inputs as an open research question.

To simplify our analysis, and since we already observe inefficient noise levels on the ChipWhisperer, we estimate the ARE and SD metrics using the original covariance matrix from Figure 6b to exhibit the noisiness levels achievable on this device in the best cases. We end our estimations by discussing the challenge of designing hardware that generates enough noise to implement circuits secure in the noisy leakage mode with reasonable security levels and finding optimal ways to solve the relaxation on such a device.

Figure 8 shows the convergence of the Monte Carlo estimation of ARE and SD metrics for the four operations as considered in Section 4.2. The curves show that both metrics converge to a stable value after around 4000 samples for each operation. For the ARE metric, it converges to a maximal value of  $\approx 2^{13.4}$  for the `xor` operation, which is enormous as this value is the same as the leakage probability in the random probing model (c.f. Section 2). Recall that the final ARE value in the noisy leakage model is the maximal ARE among all operations. This result means no constructions secure in the random probing model can be used on this device. We compare this value to the SD metric, which converges to a maximal value of  $\approx 2^{-0.0093}$  for the `xor` operation. While this value is much lower, we recall that the reduction to the random probing model with the SD metric (c.f. Section 2) induces a factor of  $2^{16}$ , equal to the size of the input space (2 inputs of 8 bits each). In other words, the leakage probability in the random probing model using the SD reduction would be almost  $2^{16}$ , which is even higher than with the ARE reduction. We observe that the values of the SD and ARE metrics are smaller for the shift operations than for the `xor` and `and` operations. We argue that this comes from the fact that the `xor` and `and` operations have two operands of 8 bits and perform an additional instruction between

registers, contrary to the shift operations, leading to more information leakage and hence higher values for the noisiness metrics.

Such values for the ARE and SD noisiness metrics imply critical leakage levels on this component, making attacks most likely possible with very few traces. It also matches the conclusions of previous works (e.g. [CDSU23]) on this component. In addition, such levels of noisiness metrics make it challenging to have provably secure implementations on the device. To show the amount of noise that needs to be added to this component to be able to use secure constructions from the literature, we present in the following section concrete results on the AES cipher and use artificial noise that we add to the samples to demonstrate the obtained security levels.

## 4.6 Step 6: Applying a Random Probing Secure Compiler

To achieve arbitrary levels of security in the random probing model, current literature proposes using an *expanding compiler* [BCP<sup>+</sup>20, BRT21, BRTV21]. We recall that the latter consists of recursively applying some base gadgets to the original circuit until achieving the desired security level. After  $k$  applications, the achieved random probing security is  $\varepsilon = f^k(p_{\text{WL}})$  where  $p_{\text{WL}}$  is the random probing wire-leakage probability and  $f$  is the simulation failure function achieved by the set of gadgets. The maximal tolerated leakage probability for current 3-share and 5-share constructions is around  $p_{\text{WL}} \approx 2^{-7.5}$  [BRT21]. In our context,  $p_{\text{WL}}$  is the square root of the maximal ARE metric over the different operations, meaning that the maximum tolerated ARE is of  $ARE \approx 2^{-15}$ . This value is very far from what we estimate in Section 4.5 on the ChipWhisperer.

### 4.6.1 Adding artificial noise and impact on ARE / SD

We illustrate the impact of noise on security in the noisy leakage model by adding artificial noise to the traces acquired with the ChipWhisperer. For simplicity, we add noise to each time sample of each trace, drawn from a univariate Gaussian distribution of mean 0 and standard deviation  $\sigma$ . We illustrate the evaluation of the ARE value for the `xor` operation, which showed the highest ARE and SD values in Section 4.5. This operation’s signal variance is about  $\sigma_{\text{signal}}^2 \approx 10^{-5}$  at the leakiest point during the execution of the operation. Table 1 shows the values of convergence for the SD and ARE metrics as done in Section 4.5, after adding different amounts of noise to the traces (i.e., different  $\sigma$  values). The table shows that the ARE value reaches  $2^{-7}$  when adding a univariate Gaussian noise of mean 0 and standard deviation  $\sigma = 5$  to each sample in the traces. Recall that this corresponds to a leakage probability of  $2^{-3.5}$  in the random probing model. Meanwhile, the SD value reaches  $2^{-10}$ , which must then be multiplied by  $2^{16}$  to obtain the leakage probability in the random probing model (because we consider 2-input 8-bit `xor` operation), making the reduction still not usable. These results showcase the difference in the tightness of the reduction from the noisy leakage to the random probing model using the SD and ARE metrics on this device. We also recall that the reduction using the ARE metric is theoretically tighter (c.f. Section 2) because the latter is a worst-case metric, matching the definition of the random probing model, a worst-case model. We then remark that the values of the ARE and SD metrics decrease as the  $\sigma$  value increases by the same factor. For instance, the ARE and SD values are halved whenever the  $\sigma$  is doubled. In order to use random probing secure gadgets from the literature, as mentioned above, we need to tolerate a leakage probability of almost  $2^{-7.5}$ , translating to an ARE value of  $2^{-15}$ . This value is reached when adding Gaussian noise with a significant standard deviation  $\sigma \approx 1280$ .

Table 1: ARE and SD values after adding noise to the leakage traces on the ChipWhisperer.

$\sigma$	ARE	SD
5	$2^{-7}$	$2^{-10}$
10	$2^{-8}$	$2^{-11}$
20	$2^{-9}$	$2^{-12}$
40	$2^{-10}$	$2^{-13}$
1280	$2^{-15}$	$2^{-18}$

Table 2: AES operations complexity.

AES Operation	Complexity
	$(N_{\text{XOR}}, N_{\text{LSL}}, N_{\text{copy}}, N_{\text{and}}, N_{\text{rnd}})$
AddRoundKey	(16, 0, 0, 0, 0)
SubBytes	(174, 0, 111, 64, 0)
Linear layer	(54, 16, 46, 0, 0)
AES-128 (10 rounds)	(2440, 160, 1570, 640, 0)

#### 4.6.2 Application to AES

We now illustrate the impact of the implementation’s noise level on the complexity of the expanding compiler in the random probing model. We choose a provably secure implementation of AES as in [BCP<sup>+</sup>20], under the verified and relaxed leakage assumptions. We consider a bitslice implementation of AES using the 8-bit bitwise operations (`xor`, `and`, `left shift logical`). Table 2 gives the operation counts for such an implementation. The `copy` operation outputs two values equal to the single input value, and the `rnd` operation outputs a fresh uniform random value.  $N_g$  denotes the number of operations for the operation  $g$  in the circuit.

For the s-box, we use the optimized Boolean circuit from [BMP13]. This circuit computes the AES s-box with 32 ANDs, 83 XORs and 4 NOTs. Moreover, it involves 111 copies. In our context, NOTs are computed as XORs with a constant operand, which makes 32 ANDs + 87 XORs + 111 copies. For a full SubBytes layer, composed of 16 bitsliced s-boxes, this makes 32 ANDs + 87 XORs + 111 copies in terms of 16-bit operations, which is 64 ANDs + 174 XORs + 222 copies in terms of 8-bit operations.

For the linear layer, we rely on the *fixslicing approach* proposed in [AP21]. For the linear layer in one round, this approach requires 27 32-bit XORs, 16 word-wise rotations, 16 byte-wise rotations and 23 copies. In our context, word-wise rotations are free since they are by multiples of 8. A byte-wise rotation requires 2 LSHs (one left shift, one right shift). This makes a total of 108 XORs + 32 LSH + 92 copies in terms of 8-bit operations for the MixColumn layer for two blocks, which is 54 XORs + 16 LSH + 46 copies per block.<sup>10</sup>

We apply the expanding compiler proposed in [BCP<sup>+</sup>20] with the 3-share gadgets proposed in [BRT21]. The LSL gadget applies the LSL operation to each input share before refreshing the sharing using a refresh gadget. Table 3 summarizes the complexities of these gadgets. As for the failure functions for the set of gadgets, we compute them using the verification tool IronMask [BMRT22].

The operation counts after  $k$  applications of the expanding compiler is given by  $\mathbf{N}^k \cdot \vec{N}_{\text{AES}}$

<sup>10</sup>We note that this is for the even rounds, while odd rounds are further optimized in [AP21] but we consider the same count for all the rounds.

Table 3: Complexities for the 3-share gadgets from [BRT21] achieving  $(t = 1, f)$ -RPE.

Gadget	Complexity
	$(N_{\text{XOR}}, N_{\text{LSH}}, N_{\text{COPY}}, N_{\text{AND}}, N_{\text{RND}})$
$G_{\text{refresh}}$	(4, 0, 2, 0, 2)
$G_{\text{XOR}}$	(11, 0, 4, 0, 4)
$G_{\text{LSL}}$	(4, 3, 2, 0, 2)
$G_{\text{copy}}$	(8, 0, 7, 0, 4)
$G_{\text{AND}}$	(40, 0, 29, 9, 17)

Table 4: Masked AES for different levels of expansion.

Expansion level $k$	Masked AES Complexity	ARE for $2^{-\lambda}$ security		
		$\lambda = 32$	$\lambda = 64$	$\lambda = 128$
1	0.24 Mop	$2^{-40}$	$2^{-72}$	$2^{-136}$
2	6.14 Mop	$2^{-28}$	$2^{-44}$	$2^{-76}$
3	163 Mop	$2^{-22}$	$2^{-30}$	$2^{-46}$
4	4.33 Gop	$2^{-18}$	$2^{-22}$	$2^{-30}$

(c.f. [BCP<sup>+</sup>20]) where  $\mathbf{N}$  is the gadget gate-count matrix defined as

$$\mathbf{N} = (\vec{N}_{\text{XOR}} \mid \vec{N}_{\text{LSH}} \mid \vec{N}_{\text{COPY}} \mid \vec{N}_{\text{AND}} \mid \vec{N}_{\text{RND}}), \quad (23)$$

and  $\vec{N}_{\text{AES}}$  is the gate-count vector for AES given by the last row of Table 2.

Table 4 summarizes the complexities of the obtained masked AES with expansion levels  $k \in \{1, 2, 3, 4\}$ . For each expansion level, it further gives the maximal value of the ARE in order to reach a provable security of  $\varepsilon = 2^{-\lambda}$ , for  $\lambda \in \{32, 64, 128\}$ . In order to compute the ARE value, we use the failure functions computed with IronMask and numerically estimate the required leakage probability  $p$  to achieve the security level given the expansion level  $k$ . This translates to the required noise level (or ARE) on the physical device to achieve the proven security level. We also recall that the ARE value is then obtained as  $p^2$ .

Table 4 shows that by doing one level of expansion, which consists of replacing each gate of the circuit with the corresponding gadget, the required levels of ARE are very high and reach  $2^{-136}$  for 128-bit security. As we further apply the expansion, this required level decreases to almost  $2^{-30}$  for 128-bit security, but the complexity of the circuit becomes quickly impractical ( $4.33 \times 10^9$  operations). Meanwhile, to have an ARE value of  $2^{-30}$  on the chip we use for our tests, for example, huge amounts of noise must be added ( $\sigma \approx 5 \cdot 2^{23}$ , c.f. Table 1).

Our results emphasize the trade-off between the physical noise on a device and the complexity of circuit implementation on this device with proven security. Higher noise levels lead to less complex constructions while achieving such noise requires specialized hardware that enables considerable noise independent of the operations. This also emphasizes the challenge of constructing such hardware, taking provable security into account and the limitations of the noise levels that can be achieved in practice. We recall that the ChipWhisperer we use is far from suitable for such a case, and the question of having adapted hardware needs to be studied further in the literature.

While the complexities obtained through our results are yet to be practical, they show that it is possible to obtain physical implementations with provable security and that the noisy leakage security of the considered device highly influences the complexity of the constructions.



## 5 Discussions and Perspectives

In this paper, we propose the first comprehensive methodology that systematically bridges the gap between the theory of provable security for masked implementations and their practical deployment. Our main objective is to provide more rigorous security guarantees than the heuristic approaches currently predominant in the field. By integrating models and metrics from the literature in a structured manner, we enable a transfer of formal security claims into quantitative security levels, grounded on hypotheses that can be experimentally validated. The main technical novelty lies in relaxing ideal assumptions of the noisy leakage model—namely, data isolation and noise independence—into more realistic but still stringent requirements. These relaxed assumptions maintain a strong connection to the ideal hypotheses, minimizing the gap between theoretical proofs and practical implementations.

We also introduce and validate an experimental methodology to assess the validity of these relaxed assumptions, demonstrating its application through the analysis of a masked AES implementation on a commercial off-the-shelf (COTS) Cortex-M4 microcontroller. This case study reveals three main challenges: low noise levels inherent to the device, performance overheads caused by the costly isolation, and the non-tightness of the security proofs for the masking scheme.

The issue of insufficient noise on COTS microcontrollers is well-known in the literature on software masking [CDSU23, BS21, BCS21] and is therefore not specific to our methodology. Simple COTS microcontrollers typically exhibit low noise, which can compromise security. While more complex microcontrollers often have higher noise levels, ensuring isolation on these devices—especially when treated as black boxes—poses additional challenges. From a research perspective, the noise level is a significant constraint but not a fundamental limitation, as it is possible to design custom microcontrollers with higher noise levels (e.g., through the addition of noise engines).

Our isolation technique of inserting “cleaning” instructions is suboptimal: it has a large performance cost ( $\times 13$  to  $\times 16$  overhead in the number of instructions) while being best-effort (and therefore heavily relying on empirical validation). Leveraging micro-architectural knowledge can facilitate achieving high-confidence and high-performance isolation. Further, strengthening the ISA definition to describe leakage [BGG<sup>+</sup>22] or even provide support for isolation [GMPP20, CPW24], could significantly simplify design, improve guarantees, and dramatically reduce the isolation performance overheads at a low hardware cost.

Finally, an alternative solution for the noise and isolation challenges is to apply our methodology to hardware implementations. Indeed, such implementations have typically higher noise levels and perform operations in parallel. The approach for ensuring isolation (i.e., avoiding glitches, transitions, and couplings) will be different but is already well-understood [FGM<sup>+</sup>18, CS21].

Regarding the non-tightness of security proofs, the high noise requirements and the number of shares needed to achieve a given security level (Table 4) may seem excessive from a practitioner’s perspective, especially when confronted with the current state-of-the-art attacks or the security level recently proven for a single sharing [BCG<sup>+</sup>23]. Below, we identify several sources of non-tightness and suggest directions for improvement.

- In our experiments, we consider a masking scheme based on the expansion technique [BCP<sup>+</sup>20]. While this scheme has the state-of-the-art minimum noise level requirement, the scaling of the security level with respect to the leakage parameter  $p$  is sub-optimal: an optimal masking scheme with  $n$  shares would scale as  $\mathcal{O}(p^n)$ , while ours has a lower exponent [BRT21, CFOS21]. This issue can be solved by using tighter random probing security proofs such as the ones based on probe distribution

tables [CFOS21], but this approach would require further development to scale with large computations.

- The next step in the proof is the reduction of noisy leakage to random probing. Using the ARE metric over the SD metric is already a significant gain, as it avoids a field-size loss in the proof. However, in our experiments, the ARE worst-case metric is significantly larger than the SD, canceling part of the gain. This may be due to the low noise level, as adding noise reduces the ARE vs. SD gap. If the use of worst-case leakage metrics remains an issue on noisy devices, one potential solution is to use the reduction to the average random probing model [DFS15b], which would relax the noise requirements at the cost of a stronger security model (hence more complex masking compiler).
- Finally, the reduction from gate-probing to wire-probing introduces a quadratic loss in the random probing parameter. This loss appears to be an artifact of the proof and should be optimized in future work.

On the practical side, our methodology relies on well-defined and realistic physical assumptions. We propose concrete tests to validate some of these assumptions, while others are supported by physics-based arguments. Although our approach advances towards experimentally validated hypotheses, some assumptions (e.g., noise Gaussianity, isolation of consecutive operations implying isolation of non-consecutive ones) could benefit from more direct experimental validation. Further, due to the nature of statistical tests, quantitative assumptions (e.g., independence, isolation) can never be proven, only invalidated. A fully-proven approach would instead rely on quantitative variants of these assumptions, which could be proven with high confidence using statistical tests (e.g., using statistical power and effect size [WO19]). Finally, there is room for efficiency improvements, such as optimizing the noise splitting or reducing the data isolation performance overhead.

In conclusion, our work demonstrates how to achieve provable side-channel security in practice under relaxed leakage assumptions, although current state-of-the-art constructions remain inefficient for the noise levels typical of COTS devices. We have identified several concrete paths for improving the tightness of security proofs, both from a theoretical and practical standpoint. Along with a more optimized isolation implementation, these improvements could lead to significant performance enhancements. By addressing these challenges, we aim to close the gap between the theory and practice of masked implementations, moving towards more efficient and secure designs for embedded cryptographic systems.

## Acknowledgements

Gaëtan Cassiers and François-Xavier Standaert are respectively postdoctoral researcher and senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the European Union through the ERC projects 101077506 (acronym AMAskZONE) and 101096871 (acronym BRIDGE). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

## References

- [2716] ISO/IEC JTC 1/SC 27. Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic

- modules. Standard, International Organization for Standardization, Geneva, CH, January 2016.
- [ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with  $O(1/\log(n))$  leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49896-5\_21.
- [AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96878-0\_15.
- [Ajt11] Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 715–724, San Jose, CA, USA, June 6–8, 2011. ACM Press. doi:10.1145/1993636.1993731.
- [AP21] Alexandre Adomnicaï and Thomas Peyrin. Fixslicing AES-like ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):402–425, 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8739>, doi:10.46586/tches.v2021.i1.402-425.
- [BBP<sup>+</sup>16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49896-5\_22.
- [BC22] Olivier Bronchain and Gaëtan Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):553–588, 2022. doi:10.46586/tches.v2022.i4.553-588.
- [BCG<sup>+</sup>23] Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standaert. Removing the field size loss from duc et al.’s conjectured bound for masked encodings. In Elif Bilge Kavun and Michael Pehl, editors, *COSADE 2023: 14th International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 13979 of *Lecture Notes in Computer Science*, pages 86–104, Munich, Germany, April 3–4, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-29497-6\_5.
- [BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29, Cambridge, Massachusetts, USA, August 11–13, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-28632-5\_2.
- [BCP<sup>+</sup>20] Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random probing security: Verification, composition, expansion and new constructions. In Daniele Micciancio and Thomas

- Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-56784-2\_12.
- [BCPZ16] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 23–39, Santa Barbara, CA, USA, August 17–19, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53140-2\_2.
- [BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. Cryptology ePrint Archive, Report 2021/817, 2021. URL: <https://eprint.iacr.org/2021/817>.
- [BDF<sup>+</sup>17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56620-7\_19.
- [BGG<sup>+</sup>14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014. doi:10.1007/978-3-319-16763-3\_5.
- [BGG<sup>+</sup>22] Roderick Bloem, Barbara Gigerl, Marc Gourjon, Vedad Hadzic, Stefan Mangard, and Robert Primas. Power contracts: Provably complete power leakage models for processors. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 381–395, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560600.
- [BGNT15] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Tégli. Multivariate high-order attacks of shuffled tables recomputation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 475–494, Saint-Malo, France, September 13–16, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48324-4\_24.
- [BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *Journal of Cryptology*, 26(2):280–312, April 2013. doi:10.1007/s00145-012-9124-7.
- [BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. IronMask: Versatile verification of masking security. In *2022 IEEE Symposium on Security and Privacy*, pages 142–160, San Francisco, CA, USA, May 22–26, 2022. IEEE Computer Society Press. doi:10.1109/SP46214.2022.9833600.

- [BRT21] Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 313–343, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-77886-6\_11.
- [BRTV21] Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 157–188, Singapore, December 6–10, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-92075-3\_6.
- [BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):202–234, 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8973>, doi:10.46586/tches.v2021.i3.202-234.
- [BUS21] Davide Bellizia, Balazs Udvarhelyi, and François-Xavier Standaert. Towards a better understanding of side-channel analysis measurements setups. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2021. doi:10.1007/978-3-030-97348-3\_4.
- [CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations towards analytical attacks against 32-bit implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):270–293, 2023. doi:10.46586/tches.v2023.i3.270-293.
- [CFOS21] Gaëtan Cassiers, Sebastian Faust, Maximilian Ortl, and François-Xavier Standaert. Towards tight random probing security. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 185–214, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84252-9\_7.
- [CGP<sup>+</sup>12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In Werner Schindler and Sorin A. Huss, editors, *COSADE 2012: 3rd International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81, Darmstadt, Germany, May 3–4, 2012. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-29912-4\_6.
- [Cho15] Marios O Choudary. Efficient multivariate statistical techniques for extracting secrets from electronic devices. Technical report, University of Cambridge, Computer Laboratory, 2015.
- [CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J.

- Wiener, editor, *Advances in Cryptology – CRYPTO’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany. doi:[10.1007/3-540-48405-1\\_26](https://doi.org/10.1007/3-540-48405-1_26).
- [CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013. doi:[10.1007/978-3-319-08302-5\\_17](https://doi.org/10.1007/978-3-319-08302-5_17).
- [CM24] Gaëtan Cassiers and Charles Momin. The SMAeSH dataset. *Cryptology ePrint Archive*, Paper 2024/1521, 2024. URL: <https://eprint.iacr.org/2024/1521>.
- [CPRR14] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424, Singapore, March 11–13, 2014. Springer Berlin Heidelberg, Germany. doi:[10.1007/978-3-662-43933-3\\_21](https://doi.org/10.1007/978-3-662-43933-3_21).
- [CPW24] Hao Cheng, Daniel Page, and Weijia Wang. eLIMInate: a leakage-focused ISE for masked implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):329–358, 2024. doi:[10.46586/tches.v2024.i2.329-358](https://doi.org/10.46586/tches.v2024.i2.329-358).
- [CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28, Redwood Shores, CA, USA, August 13–15, 2003. Springer Berlin Heidelberg, Germany. doi:[10.1007/3-540-36400-5\\_3](https://doi.org/10.1007/3-540-36400-5_3).
- [CRZ18] Jean-Sébastien Coron, Franck Rondepierre, and Rina Zeitoun. High order masking of look-up tables with common shares. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):40–72, 2018. URL: <https://tches.iacr.org/index.php/TCHES/article/view/832>, doi:[10.13154/tches.v2018.i1.40-72](https://doi.org/10.13154/tches.v2018.i1.40-72).
- [CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):136–158, 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8790>, doi:[10.46586/tches.v2021.i2.136-158](https://doi.org/10.46586/tches.v2021.i2.136-158).
- [DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany. doi:[10.1007/978-3-642-55220-5\\_24](https://doi.org/10.1007/978-3-642-55220-5_24).
- [DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *Journal of Cryptology*, 32(1):151–177, January 2019. doi:[10.1007/s00145-018-9284-1](https://doi.org/10.1007/s00145-018-9284-1).

- [DFS15a] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46800-5\_16.
- [DFS15b] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 159–188, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46803-6\_6.
- [DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version. *Journal of Cryptology*, 32(4):1263–1297, October 2019. doi:10.1007/s00145-018-9277-0.
- [Dod12] Yevgeniy Dodis. Shannon impossibility, revisited. In Adam Smith, editor, *ICITS 12: 6th International Conference on Information Theoretic Security*, volume 7412 of *Lecture Notes in Computer Science*, pages 100–110, Montreal, QC, Canada, August 15–17, 2012. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-32284-6\_6.
- [DZD<sup>+</sup>17] A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pages 105–122. Springer, 2017. doi:10.1007/978-3-319-75208-2\_7.
- [FGM<sup>+</sup>18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, 2018. URL: <https://tches.iacr.org/index.php/TCHES/article/view/7270>, doi:10.13154/tches.v2018.i3.89-120.
- [GGJR<sup>+</sup>11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011. URL: [https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08\\_goodwill.pdf](https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf).
- [GMPO19] Si Gao, Ben Marshall, Dan Page, and Elisabeth Oswald. Share-slicing: Friend or foe? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):152–174, 2019. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8396>, doi:10.13154/tches.v2020.i1.152-174.
- [GMPP20] Si Gao, Ben Marshall, Dan Page, and Think Pham. FENL: an ISE to mitigate analogue micro-architectural leakage. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):73–98, 2020. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8545>, doi:10.13154/tches.v2020.i2.73-98.

- [GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the “duplication” method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES’99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48059-5\_15.
- [GPSS18] Benjamin Grégoire, Kostas Papagiannopoulos, Peter Schwabe, and Ko Stoffelen. Vectorizing higher-order masking. In Junfeng Fan and Benedikt Gierlichs, editors, *COSADE 2018: 9th International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 10815 of *Lecture Notes in Computer Science*, pages 23–43, Singapore, April 23–24, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-89641-0\_2.
- [GS18] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 385–412, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78375-8\_13.
- [HS13] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2013. doi:10.1007/978-3-319-08302-5\_15.
- [ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-45146-4\_27.
- [JS17] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643, Taipei, Taiwan, September 25–28, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-66787-4\_30.
- [KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48405-1\_25.
- [Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO’96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-68697-5\_9.
- [Man04] Stefan Mangard. Hardware countermeasures against DPA – A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology*



- *CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235, San Francisco, CA, USA, February 23–27, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24660-2\_18.
- [MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):348–375, 2019. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8402>, doi:10.13154/tches.v2020.i1.348-375.
- [MKSM22] Nicolai Müller, David Knichel, Pascal Sasdrich, and Amir Moradi. Transitional leakage in theory and practice unveiling security flaws in masked circuits. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):266–288, 2022. doi:10.46586/tches.v2022.i2.266-288.
- [MOS11] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Inf. Secur.*, 5(2):100–110, 2011. doi:10.1049/iet-ifs.2010.0096.
- [MOW17] David McCann, Elisabeth Oswald, and Carolyn Whitnall. Towards practical tools for side channel aware software engineering: ‘grey box’ modelling for instruction leakages. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017: 26th USENIX Security Symposium*, pages 199–216, Vancouver, BC, Canada, August 16–18, 2017. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/mccann>.
- [MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365, San Francisco, CA, USA, February 14–18, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-30574-3\_24.
- [MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, UK, August 29 – September 1, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11545262\_12.
- [MPW22] Ben Marshall, Dan Page, and James Webb. MIRACLE: MiCRo-ArChitectural leakage evaluation A study of micro-architectural power leakage across many devices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):175–220, 2022. doi:10.46586/tches.v2022.i1.175-220.
- [MS23] Loïc Masure and François-Xavier Standaert. Prouff and Rivain’s formal security proof of masking, revisited - tight bounds in the noisy leakage model. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 343–376, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38548-3\_12.
- [PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a Rényi day. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712, Santa Barbara,

- CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-26948-7\_24.
- [PPM<sup>+</sup>23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023. doi:10.1145/3569577.
- [PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-38348-9\_9.
- [QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001. doi:10.1007/3-540-45418-7\_17.
- [RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427, Santa Barbara, CA, USA, August 17–20, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-15031-9\_28.
- [SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425, Washington, DC, USA, August 10–13, 2008. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-85053-3\_26.
- [SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46, Edinburgh, UK, August 29 – September 1, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11545262\_3.
- [SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513, Saint-Malo, France, September 13–16, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-48324-4\_25.
- [SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *Journal of Cryptographic Engineering*, 6(2):85–99, June 2016. doi:10.1007/s13389-016-0120-y.
- [SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225, San Jose, CA,

USA, February 13–17, 2006. Springer Berlin Heidelberg, Germany. doi:  
[10.1007/11605805\\_14](https://doi.org/10.1007/11605805_14).

- [WO19] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284, Kobe, Japan, December 8–12, 2019. Springer, Cham, Switzerland. doi:[10.1007/978-3-030-34618-8\\_9](https://doi.org/10.1007/978-3-030-34618-8_9).