Side-Channel Resistant Ciphers: Model, Analysis and Design

François-Xavier Standaert[1,2], Tal G. Malkin[1], Moti Yung[1,3]

[1] Dept. of Computer Science, Columbia University.
[2] UCL Crypto Group, Université Catholique de Louvain.
[3] RSA Laboratories.

e-mails: fstandae@uclouvain.be, tal,moti@cs.columbia.edu

Abstract. Formal models that allow one to understand side-channel attacks and are also directly meaningful to practice have been an open question. Motivated by this challenge, in this work we propose an information theoretic framework for the analysis and design of cryptographic implementations secure against such attacks. It is illustratively applied to block ciphers, although it could be used to analyze a larger class of cryptosystems. The model is based on weak and commonly accepted hypotheses about side-channels that computations give rise to. It allows us to quantify the effect of practically relevant leakage functions as reductions of certain entropy measurements on the secret parameters in a cryptographic implementation. We then demonstrate that classical attacks such as Differential Power Analysis are easily integrated into the model. Our analysis results in basic design principles for side-channel resistant ciphers and in a formal definition of leakage proofness. Thus, our investigations provide, both, a theoretical underpinning of leakage resistance and meaningful guidelines for actual practical designs. We employ these principles in designing what we call "re-keying based pseudo random number generators" that are the first designs proven secure against side-channel attacks.

1 Introduction

Traditionally, cryptographic algorithms provide security against an adversary who has only black box access to cryptographic devices. That is, the only thing the adversary can do is to query the cryptographic algorithm on inputs of its choice and analyze the responses, which are always computed according to the correct original secret information. However, such a model does not always correspond to the realities of physical implementations, and actually very rarely does. During the last decade, significant attention has been paid to the physical security evaluation of cryptographic devices. In particular, it has been demonstrated that actual attackers may be much more powerful than what can be captured by the black box model.

In this paper, we investigate the security of cryptographic implementations with respect to side-channel attacks, in which adversaries are enhanced with the possibility to exploit physical leakages such as power consumption or electromagnetic radiation. A large body of experimental work has been created on the subject, *e.g.* [1, 2, 4, 5, 10, 16, 20, 21, 25, 31], and although numerous countermeasures are proposed in the literature, *e.g.* [3, 8, 11, 15, 18, 22–24, 34, 37, 38], protecting implementations against such attacks is usually difficult and expensive. Moreover, all proposals we are aware of only increase the difficulty of performing the attacks, but do not fundamentally prevent them.

Perhaps surprisingly (and to the best of our knowledge), there have been only a few attempts to model side-channel attacks properly, and to provably address their security. A notable example is the work of Micali and Reyzin [26] who initiated a theoretical analysis of side-channels. The model they suggest is very general, capturing almost any conceivable form of physical leakage. However, arguably because of the great generality of their model, the obtained positive results (*i.e.* leading to useful constructions) are quite restricted in nature, and it is not clear how they apply to practice. This is especially true for primitives such as modern block ciphers (*e.g.* the DES or AES Rijndael) for which even the black box model does not provide provable security. Thus, they suggested as an open question the study of more specialized contexts and specific scenarios which may lead to practical applications. Motivated by this challenge, we propose to analyze side-channel attacks in a model of computation that captures the structure and operations of modern block ciphers. Still, the model is general and can be used to analyze other cryptosystems. Given a cryptographic primitive (such as a block-cipher) which is secure against a traditional (black-box limited) adversary, we study how to modify it to be secure against side-channel attacks based on signal leakages. Our model is based on an information theoretic framework, in which the leakages are quantified as reductions of certain entropy measurements (defined in the paper) on the secret parameters in a cryptographic implementation. We use weak and commonly admitted hypotheses such as "only computation leads to leakage" and address a number of well known side-channel attacks, *e.g.* where the Hamming weight of a computed value is leaked. Other assumptions on the leakages obtained by an attacker could easily be considered since the quantity of information delivered by a single point of computation is a parameter of our framework. Also, the information theoretic nature of the model implies that proving the security of a construction yields real world security and does not relate to any specific kind of attack. By contrast, an insecurity in our model may or may not be secure in the real world, depending on whether or not the information available can be exploited algorithmically.

More specifically, our results are:

1. We make hypotheses about side-channels to describe the behavior of actual leakages and quantify them with sound entropy measurements.
2. We investigate general properties that can be used to analyze a wide variety of cryptographic designs, and summarize them into basic lemmas.
3. We show that classical attacks such as Differential Power Analysis (DPA) can easily be integrated in our framework.

These investigations give rise to concrete design principles we put forward[1]:

- the "re-keying design principle" indicates that having a 2-input function with one secret input and performing multiple traces repeatedly, each time revealing further partial knowledge of the secret value, can eventually reveal the secret completely. It implies that it is preferable to re-key the secret value between various computations.
- the "bit-chunk design principle" indicates that side-channel measurements deliver more information if an adversary can focus his observations on small parts of a computation than if he is forced to consider the global leakage of large chunks of bits. It involves that parallel computing is preferred for side-channel resistance.

[1] We note that large parts of these principles were implicitly used and/or intuitively known in previous works on practical side-channel attacks.

– the "table lookup design principle" indicates that (under certain assumptions) it is less leaky to have a design based on table lookup operations than employing switching network logic, since the former is more uniform and less sensitive to the input values, *i.e.* gives rise to less computations.

4. Then, we provide a formal definition of security against side-channel adversaries that we denote as *leakage proofness* (*i.e.* intuitively, a leakage proof implementation is the one for which the physical leakages are not exploitable by the adversary).
5. Based on our analysis and the derived principles, we propose two block cipher based pseudo random number generators secure against side-channel attacks (the precise class of attacks against which they are secure is defined later in the paper). We stress that this result is of both theoretical and practical interest as the security parameter used in the proofs can be kept sufficiently low to allow efficient implementations.

The rest of the paper is structured as follows. Section 2 discusses the relation between this work and the model of [26] and provides details about our hypotheses, assumptions and claims. Section 3 presents our model including the target circuit, block cipher and hypotheses about side-channel leakages. Section 4 provides basic lemmas allowing to analyze cryptographic implementations with respect to side-channel attacks and investigates a classical DPA attack in our model. The analysis results in our definition of security against side-channel attacks. Section 5 finally applies our model and analysis tools to new constructions proven secure against a class of side-channel adversaries. Concluding remarks are in Section 6 and open research problems in Section 7.

2 Preliminary remarks

In this section we introduce the relation between our results and the previous work of Micali and Reyzin. We also introduce the hypotheses, assumptions and claims that will be carefully discussed in the paper.

The traditional cryptographic model considers only abstract notions of computation, and hence cannot protect against attacks that exploit the information leakage (via electromagnetic fields, power consumption, ...) inherent in the physical execution of any cryptographic algorithm. By opposition, the aim of physically observable cryptography is to integrate adversaries possibly enhanced with these physical observations. In [26], a model of computation is consequently introduced that takes the modularity of physically observable computations into account. It notably defines the notion of *physical computer* that is basically the combination of an abstract computer (*i.e.* a Turing machine) and a leakage function. With many respects, our following results can be viewed as a specialization of this previous model with four distinct objectives:

1. To meaningfully restrict the general (worst possible) setting of [26] to reasonable (*i.e.* practically relevant) adversaries and leakage functions.
2. To relate the abstract (*i.e.* Turing machine-based) computation model of [26] to more intuitive physical notions (*e.g.* circuits, signals and operations).
3. To quantify the effect of physical leakages with a well defined entropy measurement.
4. To apply this framework to actual cryptographic primitives like block ciphers.

As a matter of fact, the aim of this specialization is to reduce the gap between the previously introduced theoretical notions of physical security and the actual attacks performed and understood by cryptographic engineers. So basically, we would like to trade some theoretical generality for more efficiency (and possibly intuition).

For this purpose, we clearly distinguish between hypotheses and working assumptions. Hypotheses are required for the model to be meaningful and are fundamental building blocks in our framework. They roughly correspond to the informal axioms introduced in [26] with two additional items:

1. We require a notion of entropy to evaluate the information leakages.
2. We require some read-only memory in the device.

We note that the definition of entropy suggested in the paper is proposed and justified as a reasonable abstraction of certain side-channel adversaries behavior. On the other hand, the point on read-only memories is rather a requirement to circuit designers, if security against side-channel attacks (as modeled in our framework) is wanted.

Next to the hypotheses, we introduce a number of working assumptions through the paper of which the objective is generally to simplify our explanations or to restrict an adversarial context. None of the working assumptions have to be considered as fundamentally correct. They only aim to characterize a target implementation or an adversary as easy to manipulate objects.

A consequence is that our results can be improved and tighter bounds of security could be obtained by relaxing these working assumptions. This is the starting point of our concluding list of open problems in Section 7. A typical example of working assumption is to consider that the measurements obtained by a side-channel adversary are perfect (*i.e.* not affected by noise). Other examples will be detailed in the paper.

Based on this hierarchy of assumptions, our claims are, as stated in the introduction:

1. The development of simple tools for the analysis of side-channel attacks.
2. The integration of DPA-like attacks in our framework.
3. The derivation of a formal definition of security.
4. The design of provably secure pseudo-random number generators.

We note finally that any attempt to model side-channel attacks properly is obviously dependent on possible improvements of the state of the art physical attacks. Positive results in our model do not mean that side-channel attacks are prevented in *all* possible physical contexts, against *any* possible adversary. What is proposed in this paper is rather to analyze a number of realistic adversaries, corresponding to the present state-of-the-art attacks. For this purposes, we illustrate our hypotheses and framework with a number of reasonable leakage functions and then determine how the entropy of secret parameters in actual implementations is affected by the resulting physical leakages. Improved attacks (if there are) could obviously be modeled with more severe entropy losses (that possibly would not provide enough security anymore). Otherwise said, our model determines what are the maximum leakages that one can tolerate if security against side-channel attacks is required.

Notations

- $A := \{A_1, A_2, ..., A_n\}$: an abstract computer composed of n abstract virtual Turing machines (VTMs).
- $\mathbf{Adv}^{na-sca}_{f_K,\text{Ł}}(\tau, q)$: the side-channel advantage of a cryptographic implementation f_K against non adaptive adversaries enhanced with a leakage function Ł.
- $C(z = f(x, y), N_x, N_y, 2^n)$: the average number of possible outputs of an n-bit 2-input function $z = f(x, y)$ if the inputs x, y are affected by two leakages giving rise to respectively N_x and N_y candidates.
- $C = E_K(P)$: the encryption of a plaintext P under a key K.
- D: a diffusion layer in a block cipher.
- $D_H(x, y)$: the Hamming distance between two bit-vectors x, y.
- $D_S(x, y)$: the signed distance between two bit-vectors x, y.
- $\mathcal{F}$: a fault insertion function.
- F_0: an exemplary signal leakage function defined in Figure 3.
- $F_1(\delta)$: an exemplary signal leakage function defined in Figure 3.
- $\mathbf{H}[S]$: the Shannon entropy of a signal S.
- $\mathbf{H}_\infty[S]$: the minimum entropy of a signal S.
- $\mathbf{H}[S|L]$: the conditional entropy of a signal S given a leakage L.
- $\overline{\mathbf{H}}_\infty[S|L]$: the average minimum entropy of a signal S given a leakage L.
- $\overline{\mathbf{H}}_\propto[S|L]$: the average secrecy of a signal S given a leakage L.
- $\mathbf{I}(S, L)$: the mutual information between a signal S and a leakage L.
- KR_i: the i^{th} key round in a block cipher.
- $L(C_A, M, R)$: a general leakage function with inputs:
  - C_A: internal configuration of the abstract computer A,
  - M: setting of the measuring apparatus,
  - R: random string to model randomness in the measurements.
- Ł(y, n, n_{obs}): the single point leakage due to the computation of an n-bit value y, assuming that the adversary is able to observe sets of n_{obs} bits (l for short).
- $\mathcal{L}(\Sigma)$: a signal leakage function.
- $\mathcal{L}(\Omega)$: an operation leakage function.
- O: the set of all the black box oracles in a physical implementation.

- $P_\infty(z = f(x, y), N_x, N_y)$: the probability of the most likely output of a 2-input function $z = f(x, y)$ if the inputs x, y are affected by two leakages giving rise to respectively N_x and N_y candidates.
- ω_i^j: an elementary operation in a physical implementation.
- Ω_i: a black box oracle in a physical implementation.
- Φ_i^j: a side-channel oracle in a physical implementation.
- $\mathcal{R}$: a probing read function.
- R_i: the i^{th} round in a block cipher.
- S: a substitution box layer in a block cipher.
- $\sigma_i(t)$: a time-dependent signal in a physical implementation.
- $\Sigma(t)$: the set of all the time-dependent signals in a physical implementation.
- $\mathcal{W}$: a probing write function.
- $W_H(x)$: the Hamming weight of a bit-vector x.
- $\oplus$: the bitwise XOR .

3 Model

In this section we present our model. First, we recall certain definitions introduced in [26]. Then, we give an intuitive description of our target circuit for physically secure applications and provide details about the class of attacks we want to prevent. We also discuss how this intuitive description can be translated into the more formal model of Micali and Reyzin. Third, we detail the block cipher to which security against side-channel attacks is to be added. We note that the target block cipher is described for concreteness, but much of our work is independent of these details. As will become clear later in the paper, the main requirement of a given cryptographic design to be analyzed is that it has clearly specified elementary operations and points of computation (representing potential leakage points), so that the overall leakage can be quantified using our framework. Fourth, we present our hypotheses about side-channel leakages. Finally, we illustrate these hypotheses and their consequences for two exemplary leakage functions.

3.1 Micali-Reyzin computational model

In order to enable the analysis of physically observable cryptography, Micali and Reyzin introduced a model of computation of which we recall certain definitions of interest with respect to our following results. It is based on five informal axioms:

Axiom 1. Computation and only computation leaks information.

That is, we assume that it is possible to store some secret information securely in a cryptographic device. No leakages will compromise this secret as long as it is not used in any computation. As a matter of fact, this implies that probing attacks are out of the scope of our analysis and we rely on physical protections to prevent them.

Axiom 2. The same computation leaks different information on different computers.

In other words, an algorithm is an abstraction: a set of general instructions whose physical implementation may vary. As a result, the same elementary operation may leak different information on different platforms.

Axiom 3. The information leakage depends on the chosen measurement.

The amount of information that is recovered by an adversary during an actual attack depends on the chosen measurement, that possibly introduces some randomness.

Axiom 4. The information leakage is local.

In other words, the maximum amount of information that may be leaked by a physically observable device is the same in any execution of the algorithm with the same inputs, since it relates to the computer's internal configuration.

Axiom 5. All the information leaked through physical observations can be efficiently computed from a computer's internal configuration.

That is, given an algorithm and its physical implementation, the information leakage is a polynomial time computable function of (1) the algorithms internal configuration (because of Axiom 4), (2) the chosen measurement (because of Axiom 3), and possibly (3) some randomness outside anybodys control (also because of Axiom 3).

We note that, from the practical point of view, these axioms may not reflect the entire physical phenomenons observed. For example, as far as Axiom 1 is concerned, volatile memories such as RAMs regularly require a small amount of energy to refresh their values and this could be used to mount a side-channel attack. However, such leakages are significantly more difficult to exploit than computational leakages. Our expectation is therefore that these axioms approximates the physical reality to a sufficient degree.

From these axioms, an *abstract computer* is defined as a collection of special Turing machines, which invoke each other as subroutines and share a special common memory. Each member of the collection is denoted as an *abstract virtual-memory Turing machine* (abstract VTM or simply VTM for short). One writes $A := \{A_1, A_2, ..., A_n\}$ to mean that an abstract computer A consists of abstract VTMs $A_1, A_2, ..., A_n$. All VTM inputs and outputs are binary strings always residing in some virtual memory. Abstract computers and VTMs are not physical devices: they only represent logical computation and may have many different physical implementations.

Then, to model the physical leakage of any particular implementation, the notion of *physical VTM* is introduced. A physical VTM is a pair (L_i, A_i), where A_i is an abstract VTM and L_i is a leakage function. If $A := \{A_1, A_2, ..., A_n\}$ is an abstract computer then $P_i = (L_i, A_i)$ represents one physical implementation of A_i and $P := \{P_1, P_2, ..., P_n\}$ is defined as a physical implementation of the abstract computer A.

Finally, in these definitions, the relation between an abstract computing machine and a physical implementation is only determined by the leakage function that is defined as a function of three inputs, $L(C_A, M, R)$:

- The first input is the current internal configuration C_A of an abstract computer A, which incorporates anything that is in principle measurable.
- The second input M is the setting of the measuring apparatus (in essence, a specification of what the adversary chooses to measure).
- The third input R is a random string to model the randomness of the measurement process, *e.g.* typically, R models the noise that affect the useful leakage signal.

In the following section, we aim to describe a target circuit for physical attacks and relate this more intuitive view to the previous definitions.

3.2 Target circuit

Our target cryptographic implementation is schematized in Figure 1.

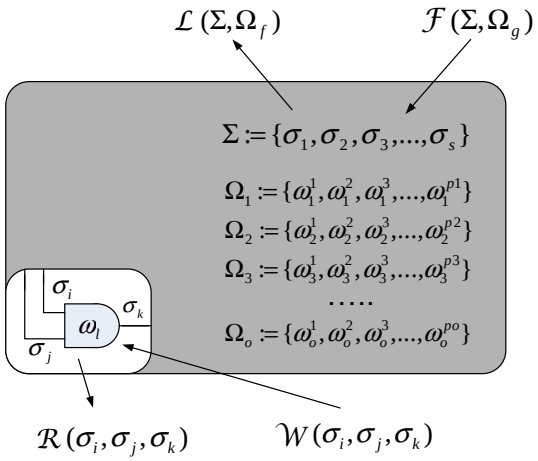

Fig. 1. Circuit model including physical threats.

It is defined as a combination of signals and operations. First, the set of all signals in the circuit is denoted as:

$$\Sigma := \{\sigma_1, \sigma_2, \sigma_3, ..., \sigma_s\},$$

where s is the total number of signals in the device. As physical signals are usually binary coded, we generally have $\sigma_i \in \mathbb{Z}_2$. In certain contexts, it may also be interesting to consider subsets of signals $\Theta_j := \{\sigma_l, \sigma_m, \sigma_n, ...\} \subset \Sigma$. In practice, the signal values are time-dependent and we have:

$$\Sigma(t) := \{\sigma_1(t), \sigma_2(t), \sigma_3(t), ..., \sigma_s(t)\}$$

Second, the cryptographic device can apply operations to the signals. A number of operations are actually included in the black box model. For example, if we consider a block cipher implementation, an attacker can query the block cipher and obtain plaintext/ciphertext pairs. As a circuit could contain several such operations, we define the set of oracles O as the set of operations that one can query in the black box model:

$$O := \{\Omega_1, \Omega_2, \Omega_3, ..., \Omega_o\}$$

Then, in actual implementations, these oracles are made of several elementary operations that cannot be queried by the black box attacker (but possibly by the side-channel one) because they apply to the circuit inner signals. For every oracle Ω_i, we have:

$$\Omega_i := \{\omega_i^1, \omega_i^2, \omega_i^3, ..., \omega_i^{p_i}\}$$

We mention that we make no hypothesis about the actual form of the elementary operations ω_i^j's, although Figure 1 suggests that they represent logic gates. For clarity purposes, we represented our cryptographic implementation as a hardware circuit where every ω_i^j is physically implemented. However, in practice, different operations could be performed by the same hardware resource. This is typically the case in software-programmed processors and in this latter context, the ω_i^j's represent instructions applied sequentially to the signals rather than physical resources.

Based on these definitions, we can consider different types of physical opponents. For example, an invasive probing attack gives read/write access to a limited subset of signals in the device (*i.e.* the functions $\mathcal{R}$ and $\mathcal{W}$ in the figure). A fault attack applies some probabilistic function $\mathcal{F}$ to the signals or operations (it is probabilistic in the sense that a signal or operation is affected by the fault function with a certain probability). Finally, side-channel attacks enhance the opponent with a leakage function. In the following, we only consider these side-channel opponents. More specifically, we will investigate the signal leakage function $\mathcal{L}(\Sigma)$. Attacks based on the operation leakage function $\mathcal{L}(\Omega)$ (*i.e.* typically timing attacks or simple power and electromagnetic analysis) are not considered in this paper. As a matter of fact, in the context of block ciphers that will be our running example, preventing the operation leakages is usually much easier than preventing the signal leakages.

It is important to observe that such a description can be efficiently translated into the formalism of [26]. Basically, our oracles Ω_i's can be simulated with abstract computers and the elementary operations ω_i^j's with VTMs. Also, our signals are simply the inputs and outputs of the VTMs. At this point, the only significant difference between the models relates to the specialization of the leakage function. While this function was kept totally general in [26], we introduce certain restrictions on its inputs C_A, M and R.

Formally, these restrictions are summarized in two first working assumptions:

> *WA*.1: *We only investigate the signal leakage function* $\mathcal{L}(\Sigma)$. That is, we consider an adversary that will not take advantage of the complete internal configuration of an abstract computer, but a part of it. Note that a signal leakage function can simply be defined as a surjective function from some secret signal space to a smaller leakage space.
>
> *WA*.2: *We assume perfect measurements.* That is, we consider that the measurements are not affected by the randomness modeled by the leakage function's R input and therefore are perfectly (deterministically) predicted with the leakage function.

On the other hand, we do not restrict leakage function's M input and assume that the adversary potentially measures any information available in a physical implementation. This will be re-stated later in the paper.

3.3 Target block cipher

A block cipher transforms a plaintext block P of a fixed bit length n_b into a ciphertext block C of the same length, under the influence of a cipher key K, of bit length n_k. We denote the forward operation of a block cipher as the encryption: $C = E_K(P)$ and the reverse operation as the decryption: $P = D_K(C)$.

In practice, modern block ciphers are usually composed of several identical transforms, denoted as the encryption (*resp.* decryption) rounds. If such a product cipher applies the same round function r times to the cipher state, it is necessary to expand the cipher key K into different round keys k_i. This is done by means of a key round. The round and key round functions are respectively denoted as:

$$\begin{aligned} p_{i+1} &= \mathrm{R}(p_i, k_i), \\ k_{j+1} &= \mathrm{KR}(k_j), \end{aligned}$$

where the p_i's represent the cipher state, with $p_0 = P$, $p_{r+1} = C$ and $k_0 = K$.

Finally, we model our round and key round functions as made of 3 different operations: a non-linear substitution layer, a linear diffusion layer and a bitwise XOR layer. Those are usual components of present block ciphers, *e.g.* the AES Rijndael [14].

More specifically, the substitution layer S consists of the parallel application of substitution boxes s to the b-bit blocks of the state:

$$\mathrm{S} : (\mathbb{Z}_{2^b})^{\frac{n_b}{b}} \rightarrow (\mathbb{Z}_{2^b})^{\frac{n_b}{b}} : x \rightarrow y = \mathrm{S}(x) \Leftrightarrow y^i = s\big(x^i\big), \quad 0 \leq i \leq \frac{n_b}{b} - 1,$$

where x^i is the ith b-bit block of the state vector x. The small S-boxes s are assumed to have good non-linearity, differential profile, non-linear order *etc.* As will be discussed later, we only address security against side-channel attacks, and thus need to assume that the given primitive is "good" in the traditional sense of security.

The linear diffusion layer D applies to the whole state and is assumed to have good diffusion properties (*e.g.* avalanche effect, high branch number, *etc.*):

$$\mathrm{D} : \mathbb{Z}_{2^{n_b}} \rightarrow \mathbb{Z}_{2^{n_b}} : x \rightarrow y = \mathrm{D}(x)$$

Finally, the bitwise XOR layer $\oplus$ is denoted as:

$$\oplus : \mathbb{Z}_2^{n_b} \times \mathbb{Z}_2^{n_b} \rightarrow \mathbb{Z}_2^{n_b} : x, y \rightarrow z \Leftrightarrow z(i) = x(i) \oplus y(i), \quad 0 \leq i \leq n_b - 1$$

where $x(i)$ is the ith bit of the state vector x. For example, a 3-round block cipher, is represented in Figure 2. With respect to the model of Section 3.2, the complete block cipher is an oracle Ω_1 and a possible division in elementary operations would be $\Omega_1 := \{\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3, \mathrm{KR}_1, \mathrm{KR}_2, \mathrm{KR}_3\}$. Another division (with smaller operations) is $\Omega_1 := \{\oplus_1, ..., \oplus_4, \mathrm{S}_\mathrm{A}, ..., \mathrm{S}_\mathrm{F}, \mathrm{D}_1, ..., \mathrm{D}_6\}$. As already mentioned, the choice of elementary operations is let open in our model and they can be as small as logic gates.

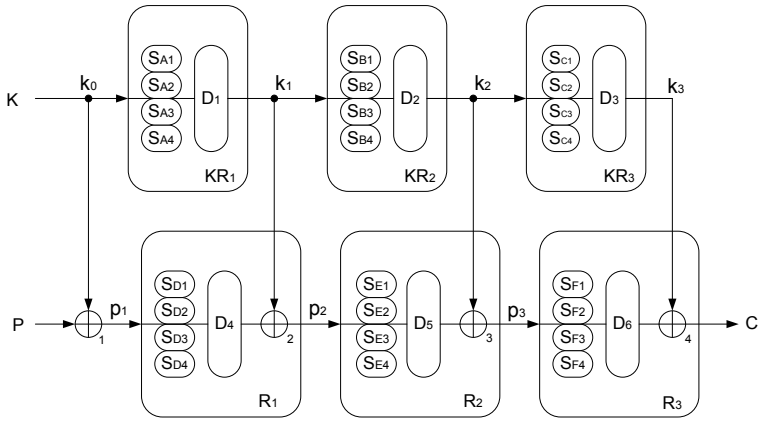

Fig. 2. A 3-round block cipher

3.4 Hypotheses

Our hypotheses are in two parts. First, we use the informal axioms of Micali and Reyzin, introduced in Section 3.1. Second, we add two new hypotheses that we detail more carefully in this section. Beforehand, we introduce a third working assumption in order to clearly specify the adversarial context of the side-channel attacks we consider:

> *WA.*3: *We consider a side-channel adversary that only perform non adaptive queries.* That is, if an adversary is allowed to query some oracle, the queries are non adaptive if they do not depend to the previous answers of the oracle; the queries are adaptive if they do depend on the previous answers of the oracle. In our context, it means that we consider an adversary that can observe an arbitrary number of side-channel leakages, but cannot choose its queries in function of these leakages.

A note on the definitions of entropy: In the following of the paper, we aim to evaluate the effect of physical leakages as entropy reductions on the secret data manipulated in a device. As a consequence, we start with a brief discussion on the definition of entropy to use in this context. Let S and L be two random variables in the discrete domains $\mathcal{S}$ and $\mathcal{L}$, respectively denoting a secret signal and its corresponding leakage. The *entropy* of S is defined as the expectation $\mathbf{H}[S] = \mathbf{E}_{s \leftarrow \mathcal{S}}[-\log_2 \mathbf{P}[S = s]]$. The *conditional entropy* of S given L is written $\mathbf{H}[S|L] = \mathbf{E}_{l \leftarrow \mathcal{L}} \mathbf{H}[S|L = l]$. The notion of entropy designates the "expected randomness" of a random variable.

To quantify the cryptographically more robust notion of "worst-case randomness", one generally considers the *minimum entropy* of S (see [7, 13] for recent uses), which is defined as $\mathbf{H}_\infty[S] = -\log_2 \max_{s \in \mathcal{S}} \mathbf{P}[S = s]$. It corresponds to the negative logarithm of the most likely element in a distribution. For conditional distributions, the notion of *average minimum entropy* is used, defined as $\overline{\mathbf{H}}_\infty[S|L] = -\log_2 \mathbf{E}_{l \leftarrow \mathcal{L}}[\max_{s \in \mathcal{S}} \mathbf{P}[S = s|L = l]]$. This is not the expected minimum entropy of S given L but rather the negative logarithm of the average probability of the most likely value of S given L. It is more pessimistic than the entropy since $\overline{\mathbf{H}}_\infty[S|L] \leq \mathbf{E}_{l \leftarrow \mathcal{L}} \mathbf{H}[S|L = l]$.

In addition to these common notions, we finally propose to define the *average secrecy* of a conditional distribution as follows:

$$\overline{\mathbf{H}}_{\infty}[S|L] = -\log_2 \mathbf{E}_{l \leftarrow \mathcal{L}}[\mathbf{E}_{s \leftarrow \mathcal{S}|L=l}\mathbf{P}[S = s|L = l]]$$

Informally, this is the negative logarithm of the average *predictability* (where predictability means average probability to predict) of S given L. We note that, for a fixed L value, if the random variable S has the uniform distribution, then the average minimum entropy and the average secrecy are the same. Also, the average secrecy only differs from the regular conditional entropy by the place of the negative logarithm. Therefore, thanks to Jensen's inequality, we have that $\overline{\mathbf{H}}_{\infty}[S|L] \leq \mathbf{E}_{l \leftarrow \mathcal{L}}\mathbf{H}[S|L = l]$.

Hypothesis 1: *The side-channel leakages obtained from non adaptive queries to a physically observable device reveal an average amount of information correctly measured by the definition of average secrecy.*

In other words, we assume that the effect of any leakage function is to reduce the space of the secret values in an implementation and we use the average secrecy to quantify the probability that an adversary can predict these secret values given the obtained leakages. This is a very important hypothesis with respect to our following results. Therefore, we discuss the consequences of the hypothesis in this section. It will be further illustrated with respect to a practical example, in Sections 4.1, 4.2.

Looking at the previous definitions, a first (known) observation is that Shannon entropy does measure the amount of information contained in a random variable and therefore says nothing about the security of this random variable. In general, cryptographers are rather interested in the a-posteriori probability to predict some secret data given some obtained information. For example, an encryption scheme $C = E_K(P)$ satisfies *perfect secrecy* [35] if $\mathbf{P}[P = p_i|C = c_i] = \mathbf{P}[P = p_i]$. Similarly, in side-channel attacks, we would like to quantify the probability that an adversary can predict some secret data given some obtained leakages. It yields the following question: "*given some leakage, should one evaluate the security of a secret parameter with its worst case probability of prediction or its average probability of prediction?*". That is, should we use average minimum entropy or average secrecy in our discussions?

As already mentioned, in case of uniform conditional distributions, these notions are identical. The difference between minimum entropy and secrecy is best understood through a simple example with non uniform distributions. Say we consider 3-bit signals and an obtained leakage L_1 such that a secret signal S_1 is distributed according to $S_1 \leftarrow \{1, 2, 2, 2, 3, 4, 5, 6\}$. Applying our definitions, we find:

$$\max \mathbf{P}[S_1|L_1] = \tfrac{3}{8} \simeq \tfrac{1}{2.67}$$

$$\mathbf{E}[\mathbf{P}[S_1|L_1]] = \tfrac{5}{8} \cdot \tfrac{1}{8} + \tfrac{3}{8} \cdot \tfrac{3}{8} \simeq \tfrac{1}{4.57}$$

Let us also consider a leakage L_2 such that $S_2 \leftarrow \{1, 2, 2, 2, 3, 3, 3, 6\}$. It now yields:

$$\max \mathbf{P}[S_2|L_2] = \tfrac{3}{8} \simeq \tfrac{1}{2.67}$$

$$\mathbf{E}[\mathbf{P}[S_2|L_2]] = \tfrac{2}{8} \cdot \tfrac{1}{8} + \tfrac{6}{8} \cdot \tfrac{3}{8} \simeq \tfrac{1}{3.2}$$

Clearly, the consequence of considering a worst case probability (*i.e.* minimum entropy) is the impossibility to distinguish between these two practically different situations. However, in a maximum likelihood recovery of a signal S given some leakage L (suggested *e.g.* in [9]), the probability of success not only depends on the worst case, but on the probability to distinguish the secret signal S from possible wrong candidates (*i.e.* false alarms). Similarly, our approach considers the *average* (and indeed, not worst case) security against side-channel attacks. Otherwise said, *the average secrecy models the average success rate of a maximum likelihood adversary* and this is the main motivation and meaning of our hypothesis. We note that, as will be detailed later, this notion of average security only makes sense in the context of non adaptive adversaries (involving the restriction of our hypothesis). We remark finally that this does not mean that worst case situations are out of the scope of our investigations. In fact, these worst cases can actually happen and will be included in our definition of security against side-channel adversaries, *i.e.* in Section 4.3.

Hypothesis 2: *There is sufficient read-only memory in the device.*

In other words, there is some data that can be stored in the circuit and cannot be overwritten by potential attackers (but can be read). A similar hypothesis has been made in the algorithmic tamper-proofness work of [17].

As already said, this hypothesis is mainly a requirement to circuit designers that is necessary for the construction of security proofs in Section 5. But the analysis of side-channel attacks in our model does not require the hypothesis.

3.5 Illustrations and consequences

We use the following definitions and symbols: (1) the Hamming weight of a n-bit vector $x \in \mathbb{Z}_2^n$ is denoted as $W_H(x)$, (2) the Hamming distance between two n-bit vectors $x, y \in \mathbb{Z}_2^n$ is denoted as $D_H(x, y) = W_H(x \oplus y)$, (3) we also define the notion of signed distance between two n-bit vectors $x, y \in \mathbb{Z}_2^n$ as $D_S(x, y) = \sum_{i=0}^{n-1}(y(i) - x(i))$, where $x(i)$ is the ith bit in the vector x and we have $-n \leq D_S(x, y) \leq n$.

Our two exemplary leakage functions are represented in Figure 3. They were selected both for their generality and because they correspond to usual assumptions made in practical side-channel attacks, as we briefly justify in the following.

These leakage functions are general since they can be explained by the physical origin of the leakages. As a matter of fact, physically observable phenomena always relate to data manipulations (*e.g.* energy or power is only consumed to alter a system state, electromagnetic radiation only happens if current flows in the circuit, *etc.*). Since most electronic devices work at the bit level, more bit modifications mean more power consumption or more electromagnetic radiation, thus more physically observable leakage. As for Section 3.1's first axiom, one may argue that small amounts of energy are also required to refresh the unmodified values during a computation. Again, we suppose that they are significantly harder to exploit. We therefore have to determine what are the possible bit modifications and how they affect the external leakages. Clearly, there are only two possible cases, denoted as the leakage functions F_0 and F_1. They differ by their ability (or lack thereof) to distinguish between $0 \rightarrow 1$ and $1 \rightarrow 0$ bit transitions.

Leakage function	Bit Modification	Externally observable leakage
F_0	$0 \rightarrow 0$ $1 \rightarrow 1$	0
	$0 \rightarrow 1$ $1 \rightarrow 0$	1
$F_1(\delta)$	$0 \rightarrow 0$ $1 \rightarrow 1$	0
	$0 \rightarrow 1$	1
	$1 \rightarrow 0$	$1\text{-}\delta$

Fig. 3. Leakage functions.

More important, these functions are of practical interest with respect to the present side-channel attacks described in the literature. The leakage function F_0 corresponds to a usual assumption in power analysis against CMOS circuits, where only the amplitude of the currents in a device is exploited. It is generally denoted as the Hamming weight or distance leakage model [5, 21, 25, 36]. The leakage function F_1 corresponds to more powerful opponents. For example, if $\delta = 2$, it actually relates to an electromagnetic analysis, where not only the amplitude of the currents is leaked, but also their direction [19, 28]. In this context, the signed distance is revealed. We remark that both functions are idealized and an attacker will rarely recover exact Hamming or signed distances in one single leakage measurement (although he can be very close to). Of course, those are not the only possible leakage functions to mount a side-channel attack. They only constitute a reasonable starting point with respect to practical adversaries. As mentioned in the preliminary remarks, more powerful attacks potentially exist, *e.g.* using space or time localization. These enhanced adversaries can be modeled in our setting with more severe secrecy losses. For simplicity, we only considered the leakage functions F_0 and F_1 with $\delta = 2$ (*i.e.* the signed distance model). As a matter of fact, the efficiency of a leakage function depends on the number of discrete values the leakages are distributed on. For example, the Hamming and signed distances of an n-bit data respectively yield $n+1$ and $2n+1$ possible discrete values for the leakage functions. Considering leakage function F_1 with $\delta \neq 0, 2$ would involve even stronger secrecy losses and is a scope for further research, from the theoretical and practical point of view.

Consequences: Let us now consider a computation (typically involving key bits) producing a value y with secrecy n (before any side-channel attack has been applied), meaning that the probability to predict y is 2^{-n}. Let us also assume that the computation result y overwrites some other value y_p previously stored in the circuit. That is, there is a signal in the circuit that switches from the value y_p to the value y. For simplicity, we take y, y_p to be uniform n-bit values, rather than longer strings with n bits of secrecy. Depending on the leakage function, the maximum leakage will be either $D_H(y_p, y)$ or $D_S(y_p, y)$. As the possible Hamming distances $h \in [0, n]$ and the possible signed distances $s \in [-n, n]$, we have:

$$\mathbf{P}[D_H(y_p, y) = h] = \frac{\binom{n}{h}}{2^n}$$

$$\mathbf{P}[D_S(y_p, y) = s] = \frac{\binom{2n}{s+n}}{2^{2n}}$$

According to the previous discussion, we can evaluate the security of this value y manipulated in a leaking device by computing its average secrecy. We derived the average secrecies in three different implementation contexts: known preliminary values, random preliminary values and deterministic preliminary values (*i.e.* when y and y_p are both unknown but relate to the same n bits of secrecy). For this purpose, we need an additional working assumption and definition.

> *WA*.4: *We assume a side-channel adversary that takes advantage of all the available information.* That is, if one obtains the leakage of an n-bit value, we consider the secrecy reduction on all the n bits. This is a strong assumption since actual adversaries usually only target some of these bits. Otherwise said, we consider an adversary that is not restricted by a key guess strategy. With respect to the model in [26], it means that we don't restrict the leakage function's M parameter.

Definition 1: A leakage table is the 2^{2n}-element table containing the leakages for all possible y and y_p values. A leakage sub-table is any 2^n-element part of the leakage table having fixed y_p values.

An example with $n = 2$ is given in Table 1.

Transition $y_p \rightarrow y$	D_H	D_S ($\delta = 2$)	Transition $y_p \rightarrow y$	D_H	D_S ($\delta = 2$)
00 → 00	0	0	10 → 00	1	-1
00 → 01	1	1	10 → 01	2	0
00 → 10	1	1	10 → 10	0	0
00 → 11	2	2	10 → 11	1	1
01 → 00	1	-1	11 → 00	2	-2
01 → 01	0	0	11 → 01	1	-1
01 → 10	2	0	11 → 10	1	-1
01 → 11	1	1	11 → 11	0	0

Table 1. Leakage table and sub-tables for $n = 2$.

The derivation of the average secrecy values is as follows:

1. *Known preliminary values* typically relate to the physical context of buses precharged with fixed values in microprocessors. In this context, the average secrecy is computed in a sub-table determined by the preliminary value y_p that is considered as a part of the leakage. Consequently, for the leakage function F_0, the adversary actually gets an information equivalent to the Hamming weight of y (since the Hamming distances are distributed identically in any subtable and if $y_p = 0$, we have $D_H(y, y_p) = W_H(y)$). It yields the following average secrecy:

$$\overline{\mathbf{H}}_\infty[y|D_H(y_p, y), y_p] = \overline{\mathbf{H}}_\infty[y|W_H(y)] = -\log_2 \sum_{h=0}^{n} \frac{\binom{n}{h}}{2^n} \cdot \frac{1}{\binom{n}{h}} = n - \log_2(n+1)$$

   In fact we obtain exactly the same expression when considering the leakage function F_1 with $\delta = 2$ since in any subtable, the Hamming and switching distances are distributed identically as well. Therefore: $\overline{\mathbf{H}}_\infty[y|D_S(y_p, y), y_p] = n - \log_2(n+1)$.

   > **Note:** It is interesting to observe the information theoretic analogy of this definition of average secrecy. Say we have a communication channel of which the input is an n-bit secret variable S and the output is a leakage variable L. The mutual information between these variables is:
   >
   > $$\begin{aligned} \mathbf{I}(S, L) &= \mathbf{H}[S] + \mathbf{H}[L] - \mathbf{H}[S, L], \\ &= \mathbf{H}[L] - \mathbf{H}[L|S], \\ &= \mathbf{H}[L], \end{aligned}$$
   >
   > since $\mathbf{H}[L|S] = 0$ (*i.e.* knowing the S, there is nothing to gain in knowing L).
   >
   > The channel capacity then corresponds to the maximum of the mutual information among all the possible probability distributions for L. It is obtained for the uniform distribution. As the number of possible leakage values is $n + 1$, we find $\mathbf{I}(S, L) = -\log_2 \frac{1}{n+1} = \log_2(n + 1)$. In our context, the mutual information also corresponds to the leakage (*i.e.* to the secrecy reduction) since we have $\mathbf{I}(S, L) = \mathbf{H}[S] - \mathbf{H}[S|L]$. We see that the channel maximizing the mutual information also yields a leakage of $\log_2(n + 1)$ bits. We stress that the Shannon entropy and average minimum entropy or secrecy are different (although sometimes close) notions. For example, they would be differently affected by noise addition. Therefore, this is just an analogy that can help the understanding of our model but only the definition of average secrecy is relevant in our context.

2. *Random preliminary values* typically correspond to a physical context where the buses are precharged with random values. That is, y_p is an unknown random data, independent of y. In this case, the combined secrecy of y and y_p is initially $2n$. It yields combined average secrecies for the vector (y_p, y):

$$\overline{\mathbf{H}}_\infty[y_p, y|D_H(y_p, y)] = -\log_2 \sum_{h=0}^{n} \frac{2^n \cdot \binom{n}{h}}{2^{2n}} \cdot \frac{1}{2^n \cdot \binom{n}{h}} = 2n - \log_2(n+1)$$

$$\overline{\mathbf{H}}_\infty[y_p, y|D_S(y_p, y)] = -\log_2 \sum_{s=-n}^{n} \frac{\binom{2n}{s+n}}{2^{2n}} \cdot \frac{1}{\binom{2n}{s+n}} = 2n - \log_2(2n+1)$$

Remark that in practice, an adversary will usually only be interested in the information on y (*i.e.* the secret signal) and not care about the knowledge of y_p (that is just a temporary random number). The leakage function F_0 would not be useful in this context because it only delivers combined information on y and y_p, but nothing on y alone. Otherwise said, the leakage cannot be predicted with it[2].

3. *Deterministic preliminary values* finally correspond to a general hardware context where both y and y_p are unknown but relate to the same n bits of secrecy, *e.g.* if $y = f(x, k)$, $y_p = f(x_p, k)$ where x, x_p denote known inputs, k denotes a secret key and f is the function used to combine x and k, *e.g.* a bitwise XOR . It means that the secrecy is computed from a 2^n-element set containing elements from every sub-table. Assuming that $f_X(k) = f(X, k)$ is a bijection (*i.e.* the knowledge on y is the same as the knowledge on k), it yields the bounds:

$$\begin{aligned}
\overline{\mathbf{H}}_\infty[k|D_H(y_p, y)] &\geq -\log_2 \textstyle\sum_{h=0}^{n} \frac{\#(k|D_H(y_p,y)=h)}{2^n} \cdot \frac{1}{\#(k|D_H(y_p,y)=h)} \\
&= n - \log_2(n+1) \\
\overline{\mathbf{H}}_\infty[k|D_S(y_p, y)] &\geq -\log_2 \textstyle\sum_{s=-n}^{n} \frac{\#(k|D_S(y_p,y)=s)}{2^n} \cdot \frac{1}{\#(k|D_S(y_p,y)=s)} \\
&= n - \log_2(2n+1)
\end{aligned}$$

Obviously, the second expression (with leakage function $F_1(2)$) is only meaningful for $n \geq 3$, since for smaller n's, there are more possible leakage values (*i.e.* $2n+1$) than elements in the set in which we compute the average secrecy (*i.e.* 2^n). It is important to remark that in this physical context, the leakage function $F_1(2)$ may deliver more information than F_0. However, this evaluation is pessimistic since we consider that the sets in which we compute the average secrecy contains all the possible $n+1$ or $2n+1$ leakage values and that these leakages depend on the key. It may not be the case in practice, depending on the function used to combine x, x_p and k. For example, it is well known that the Hamming distance of a key addition does not reveal anything on the key. Indeed, assuming $x_p \oplus k \rightarrow x \oplus k$, the observation of $W_H(x_p \oplus k \oplus x \oplus k) = W_H(x_p \oplus x)$ does not leak any key information.

[2] For example, looking at Table 1, the observation of $D_H(y_p, y) = 0$ or 2 does not allow to reject any candidate for y. On the other hand, the observation of $D_S(y_p, y) = -2$ or 2 results in only one possible candidate for y. See [28] for more details.

As the data bit-size n is an important parameter of the leakage function, it is finally worth noting that attackers not always have to observe all the n bits at once. They may also repeatedly observe smaller sets of n_{obs} bits. There are two physical factors to take into account with this respect.

1. The implementation type and size of data buses. For example, an 8-bit processor for which $n_{obs} = 8$ potentially leaks much more information than a hardware architecture dealing with 128-bit data in parallel for which $n_{obs} = 128$. Pipelining may yield even lower leakages, *e.g* $n_{obs} = n_p \times 128$ where n_p is the number of pipeline stages. This is a first part of the *bit-chunk design principle*.
2. The attacker capabilities. For example, while power analysis generally leaks an image of the whole device state, certain improved (*e.g.* electromagnetic) attacks allow focusing the measurement onto specific parts of the computation. This reduces n_{obs} to less than the implementation type specifies.

Notice that small n_{obs} values nearly correspond to probing attacks rejected by Hypothesis (*e.g.* if $n_{obs} = 1$, we have $W_H(y) = y$). Note also that in the three investigated contexts, the definitions of average minimum entropy and average secrecy result in the same expressions, because the conditional distributions are uniform.

4 Analysis

In this section, we present our tools for the analysis of cryptographic implementations against side-channel attacks. We start by considering single point leakages. These investigations give rise to two exemplary lemmas about the average secrecy of one-input and two-input functions affected by some leakages. That is, typically, we will consider the two following questions:

1. Assuming $y_1 = f_1(x)$ and $y_2 = f_2(x)$ to be two leaking computations such that, *e.g.* one knows $W_H(y_1)$ and $W_H(y_2)$, what is the resulting knowledge on x?
2. Assuming $z = f(x, y)$ to be a leaking computation such that, *e.g.* one knows $W_H(x)$ and $W_H(y)$, what is the resulting knowledge on z?

We show in the section that such questions can be formally answered by applying our definition of average secrecy. On the other hand, the resulting analysis is typically dependent on the cryptographic functions and leakage functions considered. Therefore, for simplicity and generality reasons, we also introduce two additional working assumptions that corresponds to simple bounds to approximate these formal lemmas, when assuming that cryptographic functions behave as random functions. Then, we study the application of these principles to real constructions, considering multiple points/traces leakages. We finally show that, under certain conditions on the behavior of the block cipher components, it is possible to integrate DPA in our framework and to derive (first and indeed not tight) bounds for the actual complexity of the attack. From these additional assumptions, we define the context of "exclusive side-channel attacks" and finally provide a formal definition of security against such adversaries.

4.1 Analysis of single point leakages

We start with a few definitions.

Definition 2: The *single point leakage* of a n-bit vector $y \in \mathbb{Z}_2^n$ with an attacker capability of n_{obs} bits, denoted as $Ł(y, n, n_{obs})$ (or l for short) is the average number of bits leaked by one single point of computation in an implementation, if the attacker is able to observe sets of n_{obs} bits.

In the following sections, we will only illustrate our analysis with the leakage function F_0 and assume the context of known preliminary values, *i.e.* we consider a Hamming weight leakage function. The previous working assumptions involve that $n_{obs} \leq n$ since we assume noise-free measurements. Typically, $n_{obs} > n$ would mean that some of the bits are not targeted by the adversary and therefore produce what is usually referred to as algorithmic noise. According to the previous discussions, it yields:

$$Ł(y, n, n_{obs}) = \log_2(n_{obs} + 1) \times \frac{n}{n_{obs}}$$

The multiplicative law holds because the different sets of n_{obs} bits are observed independently. Again, it is important to have in mind that considering other leakage functions would be straightforward. Indeed, any leakage assumption can be considered in our framework as long as it is expressed in such information theoretic terms.

Definition 3: A *1-input b-wise function* is a bijection of n-bit inputs acting on b-bit blocks $f : (\mathbb{Z}_{2^b})^{\frac{n}{b}} \rightarrow (\mathbb{Z}_{2^b})^{\frac{n}{b}} : x \rightarrow y = f(x) \Leftrightarrow y^i = f^i(x^i)$, $0 \leq i \leq \frac{n}{b} - 1$, where any $f^i(x^i)$ is a bijection.

Definition 4: A *2-input b-wise function* is a function of n-bit inputs acting on b-bit blocks $f : (\mathbb{Z}_{2^b})^{\frac{n}{b}} \times (\mathbb{Z}_{2^b})^{\frac{n}{b}} \rightarrow (\mathbb{Z}_{2^b})^{\frac{n}{b}} : (x, y) \rightarrow z = f(x, y) \Leftrightarrow z^i = f^i(x^i, y^i)$, $0 \leq i \leq \frac{n}{b} - 1$, where $f_X(y) = f(X, y)$, $f_Y(x)$ and any $f^i_{X^i}(y^i)$'s, $f^i_{Y^i}(x^i)$'s are bijections.

The substitution layer or diffusion layer defined in Section 3.3 are examples of 1-input b-wise functions. The bitwise XOR or addition modulo 2^n are examples of 2-input b-wise functions. Intuitively, the significance of the b parameter relates to the maximum diffusion (or algebraic complexity) of a function. Using a b-wise function implies that changing one input bit of the function will result in the change of at most b output bits. We note that the b parameter is mainly important with respect to key guess strategies since it determines the computational complexity of a key guess. Because of our fourth working assumption, most of our following analysis will be independent of b.

Based on these definitions, we now investigate the situation in which an adversary obtains several leakages related to the same secret value x. For this purpose, we first state an exemplary formal lemma allowing to derive the remaining secrecy of x given two Hamming weight-based leakages.

Lemma 1: Let f_1, f_2 be two 1-input n-bit functions. Let $y_1 = f_1(x)$, $y_2 = f_2(x)$ be two computations performed within a leaking device with $n_{obs} = n$. Assuming that a Hamming weight leakage function that only affects the function's outputs (*i.e.* y_1 and y_2), the average secrecy of x equals[3]:

$$\overline{\mathbf{H}}_{\infty}[x|W_H(f_1(x)), W_H(f_2(x))] = -\log_2 \sum_{h_1=0}^{n} \sum_{h_2=0}^{n} \frac{\binom{n}{h_1}}{2^n} \frac{\binom{n}{h_2}}{2^n} \cdot \underset{x|h_1,h_2}{\mathbf{E}} \mathbf{P}[x|h_1, h_2]$$

□

In this expression, the predictability $\underset{x|h_1,h_2}{\mathbf{E}} \mathbf{P}[x|h_1, h_2]$ can be computed for any pair of functions f_1, f_2. Basically, the lemma just applies our definition of average secrecy. It can be easily extended to various numbers of obtained leakages, each of those adding a new sum in the expression of x's average secrecy. However, as already mentioned, this computation is typically dependent on the cryptographic function considered. Therefore, for simplicity purposes, we first modeled f_1 and f_2 as random[4] functions and stated a naive bound for the secrecy reduction on x as follows:

> *WA*.5: *Attack against 1-input random functions.* Let f_1, f_2 be two 1-input random functions and $y_1 = f_1(x), y_2 = f_2(x)$ be computations performed within a device with a single point leakage on y_1, y_2 of respectively l_1, l_2 bits. Then, the average secrecy reduction on x is bounded by $l_1 + l_2$ bits.

It is obvious that such an assumption defines the worst possible adversary. Indeed, it involves that a combination of leakages delivers new information, no matter what the adversary already knows. This means that the secrecy reductions due to multiple leakages will be linear in the number of leakage points observed. However, in practice, after an adversary gets several leakages, an additional one will always contain some redundant information. Also, typically, the secrecy of x cannot be lower than zero.

In fact, the theoretical meaning of this working assumption is straightforward. We just consider that the leakages give rise to sets of candidates of respective size $2^{n-l_1}, 2^{n-l_2}$. If $f_1(x)$ and $f_2(x)$ are random functions, then these sets of candidates are independent and their intersection has size $\frac{2^{n-l_1} \cdot 2^{n-l_2}}{2^n} = 2^{n-(l_1+l_2)}$. That is, the following approximations are hidden in *WA*.5:

1. It considers the average number of x candidates rather than the average predictability of x. These notions are strongly correlated since increasing the number of x candidates reduces its predictability.
2. It neglects the impossibility of an empty intersection between the sets of x candidates generated by the leakages. In practice, there always remains one candidate in this intersection, corresponding to the secret signal used to generate the leakages. Therefore, *WA*.5 upper bounds the efficiency of the leakage combination.

[3] Since f_1 and f_2 are bijections, the secrecy of x, y_1 and y_2 are the same.
[4] This is an important assumption that will be discussed further for practical attacks.

We mention again that we considered this assumption for simplicity reasons and because it gives a good intuition of how a side-channel attack proceeds. As an illustration, in Figure 4 we compared the behavior of $WA.5$ with an evaluation of the average secrecy of an 8-bit value x, assuming that the different functions $y_i = f_i(x)$ were random. It clearly suggests that an actual combination of leakages will be less efficient than what is bounded by our working assumption.

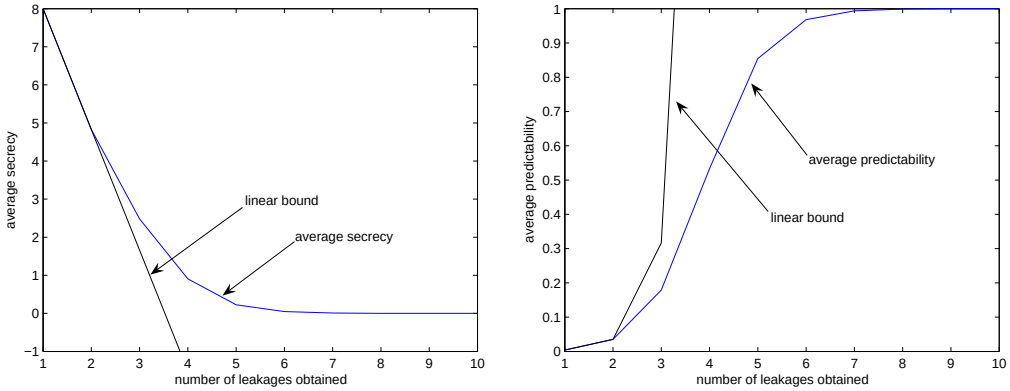

Fig. 4. Comparison between Lemma 1 and $WA.5$ for of an 8-bit value x.

> **Note:** The information theoretic analogy of the working assumption is straightforward: we just consider a (side) channel with two independent outputs for which we have the additivity of the mutual information.

We now investigate the case of 2-input functions. For this purpose, we start with the following example. Let $z = f(x, y)$ be the result of such a function with inputs x and y. Let N_x be the number of possible x values and N_y be the number of possible y values after some leakage has been obtained by an adversary. Say we consider a 2-input 3-wise function for which we have $N_x = 4$, $N_y = 2$ and we find the following possible z values: $z \leftarrow \{0, 1, 1, 2, 4, 5, 5, 6\}$. This is a typical example where average minimum entropy and average secrecy differ, since the distribution of z is not uniform. The minimum entropy of z is worth the negative logarithm of $\frac{2}{8} = \frac{1}{4}$ while the secrecy of z is the negative logarithm of its predictability:

$$\mathbf{E}_{z|N_x,N_y}[\mathbf{P}[z|z = f(x, y), N_x, N_y]] = \frac{4}{8} \times \frac{1}{8} + \frac{4}{8} \times \frac{2}{8} = \frac{1}{5.33}$$

As for Lemma 1, it is interesting to get an intuitive understanding of a 2-input function's behavior by thinking about number of candidates (*i.e.* 6 in our example) rather than predictabilities and assuming random functions. In this context, one can evaluate an average expression for the number of possible z values as follows.

Fact: Let f be a 2-input n-bit random function. Let x and y be two values computed in a leaking device such that the respective number of x, y candidates[5] is N_x, N_y. If $z = f(x, y)$, then the average number of possible z values is worth:

$$N_z = C(z = f(x, y), N_x, N_y, 2^n) = -2^n \cdot \left(1 - \frac{N_x}{2^n}\right)^{N_y} + 2^n, \qquad \text{with } N_x \geq N_y$$

Proof. Initially, we fix the number of possible z values to zero: $C(z, N_x, 0, 2^n) = 0$. Then, each candidate for y will add a list of N_x new candidates to the possible values of z minus the ones already found before: $C(z, N_x, i+1, 2^n) = C(z, N_x, i, 2^n) + N_x - \frac{C(z, N_x, i, 2^n).N_x}{2^n}$. This holds because $f_X(y)$, $f_Y(x)$ are random permutations generating independent sets of candidates. Also, since $f_X(y)$ and $f_Y(x)$ are bijections, the best estimation holds for $N_x \geq N_y$ (*i.e.* the only collisions are between the lists of N_x candidates, not within them). It yields the following recurrence equation: $C(z, N_x, i+1, 2^n) = C(z, N_x, i, 2^n).(1 - \frac{N_x}{2^n}) + N_x$ that has solution:

$$C(z, N_x, i, 2^n) = -2^n \cdot \left(1 - \frac{N_x}{2^n}\right)^i + 2^n$$

The number of possible z values N_z therefore equals $C(z, N_x, N_y, 2^n)$. □

Note that the same result can be obtained by solving the sum:

$$C(z, N_x, N_y, 2^n) = \sum_{i=1}^{N_y} (-1)^{i-1} \binom{N_y}{i} \frac{N_x^i}{2^{n(i-1)}}$$

From this expression, an interesting quantity to measure is the logarithm of the average number of z candidates per bit, *i.e.* $\log_2(N_z)/n$. It is illustrated in Figure 5 for different bit sizes $n \in [5, 20]$ and different input leakages per bit on x, y: $\frac{l}{n} \in [\frac{1}{2}, \frac{1}{5}]$. One observation is that for fixed input leakages per bit, $\log_2(N_z)/n$ is an increasing function of n. Since lower N_z values implies z candidates with higher multiplicities, it involves the important consequence that, for fixed input leakages per bit, the average secrecy of z is also an increasing function of n. This is a good news since present block cipher work on large bit sizes, *e.g.* $n = 128$. However, as already mentioned, the actual size we are interested in is n_{obs} rather than n, *e.g.* in an 8-bit processor implementation, the size of the 2-input functions will be 8, regardless the actual block cipher size. This yields the second part of the *bit-chunk design principle*.

In appendix A, the average minimum entropy of a 2-input random function's output affected by some Hamming weight-based leakage on its inputs is analyzed. It illustrates why it is preferable to use average secrecy in the context of side-channel attacks, as assumed by Hypothesis 1. We now derive a second exemplary formal lemma allowing to derive the average secrecy of a 2-input function's output.

[5] Note that the leakages give to uniform lists of candidates for x, y because of the surjective nature of the signal leakage function.

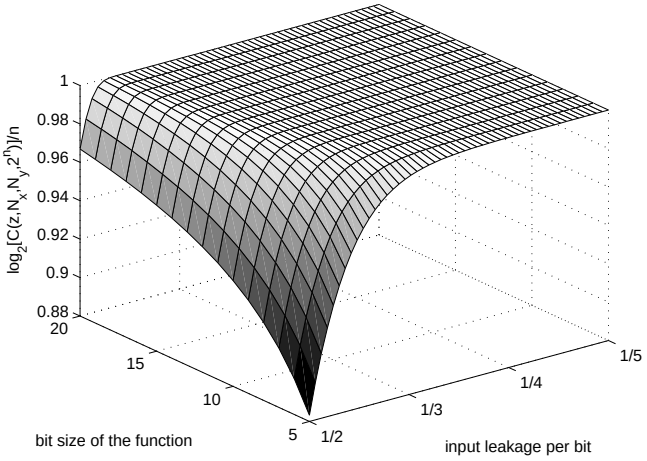

Fig. 5. Average number of output values of a non-leaking 2-input random function (in log scale, per bit), considering different input leakages per bit on x, y and different bit sizes n.

Lemma 2: Let f be a 2-input n-bit function. Let x and y be two values computed in a leaking device with $n_{obs} = n$ and the leakages on x, y be measured with the Hamming weight function. Let $z = f(x, y)$ be a *non-leaking* computation (meaning that there is no direct leakage on z but only on x, y). The average secrecy of z equals:

$$\overline{\mathbf{H}}_{\infty}[z|z = f(x,y), W_H(x), W_H(y)] = -\log_2 \sum_{h_x=0}^{n} \sum_{h_y=0}^{n} \frac{\binom{n}{h_x}}{2^n} \frac{\binom{n}{h_y}}{2^n} \cdot \mathop{\mathbf{E}}_{z|h_x,h_y} \mathbf{P}[z|h_x, h_y]$$

□

Again, we simply apply our previous definition of average secrecy and the predictability $\mathop{\mathbf{E}}_{z|h_x,h_y} \mathbf{P}[z|h_x, h_y]$ can be computed for any particular function f.

We remark that this lemma considers $z = f(x, y)$ to be a non-leaking computation. This is just an abstraction allowing to evaluate the additional leakage on z from a partial knowledge of x and y. In practice, z will be manipulated in a leaking device as well. The combination of the leakages resulting from the computation of z itself and from the partial knowledge of x and y is obtained from Lemma 1. We remark also that if one input is known, *e.g.* x, the function $z = f(X, y) = f_X(y)$ acts like a 1-input function and can be analyzed using Lemma 1.

> **Note:** This latter point suggests that the secrecy of a 2-input function also has an information theoretic analogy. We are actually in the context of a cascade of two channels, corresponding to the two inputs of the function. If an input is known, the corresponding channel is noiseless and the output is only determined by the knowledge of the second input.

As for Lemma 1, the computation of Lemma 2 depends on the 2-input function f. Therefore, we similarly assume f to be random in order to provide a simple and intuitive bound for the efficiency of this formal lemma.

WA.6: *Attack against 2-input random functions.* Let $z = f(x, y)$ be a 2-input random function computation performed within a device with a single point leakage on x, y of l_x, l_y bits. Then, the average secrecy reduction on z due to the knowledge on x, y is bounded by $\frac{l_x \cdot l_y}{n}$ bits.

The assumption considers a too-powerful adversary since it supposes the knowledge on x, y to be physical bits of which the indices are randomly distributed among their n possible positions rather than bits of secrecy. Again, the only aim of this working assumption is to provide a simple to manipulate and intuitive view of the behavior of 2-input functions affected by side-channel leakages. The important consequence of Lemma 2 (that is also understood through *WA*.6) is that, for reasonable input leakages (*e.g.* less than half the bits have been leaked), the output leakage of a 2-input function will be even smaller. Note that the assumption is specially pessimistic for large n values since it considers that the average secrecy per bit of z is independent of the bit size.

To conclude this section on the analysis of single point leakages, we illustrate the difference between Lemma 1 and Lemma 2 with a simple example. Let us first consider Lemma 1. That is, we have two 1-input 3-bit functions $y_1 = f_1(x)$ and $y_2 = f_2(x)$ that respectively give rise to the following sets of possible x values: $x \leftarrow \{0, 2, 4, 6\}$ and $x \leftarrow \{2, 3, 4, 5\}$. When combining the two leakages, one finds $x \leftarrow \{2, 4\}$. So, basically, the combination of different leakages results in a reduction of secrecy by *rejecting* certain of the key candidates. Let us now consider Lemma 2, with a function $z = f(x, y)$ and input leakages such that $x \leftarrow \{0, 2, 4, 6\}$ and $y \in \{2, 3, 4, 5\}$. Let us also assume that the 2-input function is such that we obtain $z \leftarrow \{0, 1, 1, 1, 2, 3, 3, 4, 5, 5, 5, 5, 6, 6, 7, 7\}$. In this sequence, the number of possible z values is 8, *i.e.* none of the key candidates can be rejected at this point. However, the secrecy has been reduced because certain of the key candidates are *less likely*.

This example stresses that Lemma 1 basically corresponds to the behavior of a first order side-channel attack like the DPA in which (if we assume perfect measurements, *i.e.* no noise) the secrecy reduction is deterministic. On the other hand, if the input leakages are reasonable (*i.e.* typically, less than half the bits have been leaked), Lemma 2 corresponds to the behavior of a higher order side-channel attack in which (even if the measurements are perfect), the secrecy reduction is probabilistic [24, 29, 38].

In the following sections, we apply the previous working assumptions to discuss the security of an idealized implementation against DPA-like attacks. In this context, it is important to note that obtaining tighter bounds for Lemmas 1 and 2, possibly dedicated to specific classes of functions, would allow to refine our analysis and is therefore a scope for further research. As underlined in this section, *WA*.5 and *WA*.6 both correspond to bounds of efficiency for side-channel attacks. Therefore, the following analysis is conservative and our main purpose is to initiate a formal treatment of physical security within our model, from a general point of view. As for [26], further specialization is still required to close the gap between theory and practice.

4.2 Towards the analysis of real constructions

The previous lemmas provide tools for the analysis of a large number of encryption schemes. In this section, we use the target block cipher described in Section 3.3 to illustrate how DPA can be integrated in our framework. For this purpose, we need some last definitions that gives rise to the *"table lookup design principle"*.

Definition 5: Let a *side-channel oracle* be an operation of which only the inputs/outputs are physically observable (*i.e.* it has no observable intermediate values).

From this definition, one could for example assume that the block cipher operations (*i.e.* substitution, diffusion and bitwise XOR) are side-channel oracles. That is: "only the leakages due to the inputs/outputs of the block cipher operations can be used by a potential attacker". Although it may not always be the case in practice (*e.g.* gate-level description of S-boxes have intermediate values), this assumption reasonably corresponds to the limitations of actual adversaries for at least two reasons:

- Most present attacks use one single leakage point per measurement, *e.g.* a substitution box output. This is notably because it makes the key guess strategy easy and the non-linearity of the S-box provides good separation between key candidates.
- Not all the leakage points are as easily observable. Actual attacks target values stored in a register or loaded on a data bus (because of their large capacitances and easy synchronization). Targeting intermediate values is significantly harder.

Anyway, the number of leaking points in a device is as a parameter in our framework. Looking back at the model of Section 3.2, this actually relates to the smallest operations that are considered to be observable by an opponent. Clearly, the most powerful adversary is even able to distinguish the outputs of any logic gate. Again, such outputs usually exhibit much lower correlations with the leakages than, *e.g.* data loaded on a bus. Based on these observations, we now define two classes of adversaries that we will consider in the following: limited adversaries and unlimited adversaries.

Definition 6: Let Ω_i be an oracle (as defined in Section 3.2), physically realized by side-channel oracles Φ_i^j's: $\Omega_i := \{\Phi_i^1, \Phi_i^2, \Phi_i^3, ..., \Phi_i^q\}$. That is, we re-define Ω_i, with the set of operations that are actually observable through side-channel measurements. We say that an adversary is $\{n_{obs}, t\}$-*limited* if: (1) he can observe the leakages of at most t side-channel oracles between any two 2-input functions in the oracle Ω_i; (2) the smaller sets of bits he can observe have size n_{obs}; (3) the leakage on the output of a 2-input function due to some partial knowledge on its inputs is considered negligible. We say that an adversary is $\{n_{obs}, t\}$-*unlimited* if he is not limited by the third condition.

Informally, the limited adversary is simply the one that can only take advantage of Lemma 1, *i.e.* the one for which the leakages on a 2-input function's output due to partial knowledge of its inputs is not exploited. As a matter of fact, any actual adversary is unlimited and we only use the abstraction of limited adversaries for the purposes of our analysis. The consequences of this abstraction will be detailed in Section 5.4.

Definition 7: The *single-trace leakage* of an $\{n_{obs}, t\}$-adversary with single point leakage l is the leakage obtained from the combination of t leaking points.

In practice, the parameters n_{obs} and t can be viewed respectively as the size granularity and the time granularity of the adversary. Clearly, a powerful attacker has small n_{obs} and large t values. We mention that in the context of block ciphers, t represents the number of leaking points per round. As an illustration of these definitions, implementations using precomputed memory tables are particularly interesting, if we assume these tables to be side-channel oracles. It directly gives rise to the "*table lookup design principle*". Note that this is not in contradiction with efficiency constraints as table lookup based designs are commonly used in cryptographic hardware and software implementations [30, 33]. We now analyze two basic side-channel attacks against the block cipher of Figure 2 assuming three side-channel oracles per round (*i.e.* the substitution layer, diffusion layer and key addition layer). That is, we consider a $\{n_{obs}, 3\}$-limited adversary with the following division in side-channel oracles: $\Omega_1 := \{\oplus_1, ..., \oplus_4, \mathrm{S_A}, ..., \mathrm{S_F}, \mathrm{D}_1, ..., \mathrm{D}_6\}$. For the purposes of our analysis, we assume that the working assumptions of the previous section apply to real block cipher components, *i.e.* that these components behave like random functions.

Attacking the key scheduling: Let us consider a single point leakage of l bits. Using *WA*.5 and the previously defined adversary, the execution of the key scheduling yields a secrecy reduction bounded to $6l$ bits (as there are only two side-channel oracles per key round and three key rounds are implemented).

Attacking the rounds: Under the same hypotheses as for the key scheduling, we can attack the round functions. Let us for example target the first round of Figure 2 in a known plaintext attack. The encryption device is feeded with plaintexts P's, and produces first round outputs p_1's. For each pair (P, p_1), we have full knowledge of P and a secrecy reduction on k_0 of $3l$ bits. This is because the 2-input function with a known input $f_P(k_0) = p_1$ is analyzed as a 1-input function. Assuming an initial secrecy $\overline{\mathbf{H}}_{\infty}^{init}(k_0) \geq n - 6l$ or $\overline{\mathbf{H}}_{\infty}^{init}(k_0) = n$ (*i.e.* with or without the key scheduling attack), we find $\overline{\mathbf{H}}_{\infty}(k_0) \geq \overline{\mathbf{H}}_{\infty}^{init}(k_0) - i \times 3l$, where i is the number of plaintexts used in the attack.

Remark 1: It is important to note that the non adaptive nature of the queries (assumed in *WA*.3) is fundamental in our analysis of multiple trace leakages. Looking at the previous section's first lemma, it is assumed that an adversary is provided with two leakages corresponding to computations $y_1 = f_1(x)$ and $y_2 = f_2(x)$. However, in the more practical case investigated above, the adversary would rather obtain leakages corresponding to two different plaintexts, *i.e.* $y_1 = f_{P_1}(k)$ and $y_2 = f_{P_2}(k)$. As the adversary cannot control the functions f_1 and f_2 in Lemma 1, it should not be able to do it when attacking the block cipher rounds. That is, in the context of block ciphers, our average analysis is only meaningful if the input messages are selected before the observation of the leakages. As a matter of fact, an adversary capable of adaptive queries would be more efficient than what is measured by the average secrecy. We note that we are not aware of side-channel attacks taking advantage of adaptive queries in the present literature and such a context is a scope for further context

Remark 2: As a matter of fact, the data complexity of the presented attack is (bounded to) linear in the number of side-channel measurements. However, it is important to observe that we only consider information theoretic arguments and that our assumptions are weak compared to what an actual adversary can achieve. Therefore, deriving practical attacks from actual leakages may not be as efficient and the time complexity may be a concern as well (*e.g.* typically, practical side-channel attacks only target the first and last encryption rounds). In other words, constructions could be proven insecure in our model and still achieve a relatively high level of actual security. Positively, proving the security of a construction in this model would not relate to any specific attack.

4.3 Definition of security against side-channel attacks

As already mentioned, an important feature of Section 4.1's working assumptions and Section 4.2's idealized block cipher attack is the randomness of the investigated functions or side-channel oracles. More precisely, the required property in our average secrecy bounds is that, given a number of computations involving a value x, for example $y_i = f_i(x)$'s, the obtained leakages give rise to independent sets of candidates for x. It consequently requires that the block cipher components (or more generally the side-channel oracles that are considered in an attack) fulfill this condition. In practice, this is what was meant in Section 3.3, when requiring that the underlying primitive is "good" in the traditional sense of security.

It is interesting to observe that in Lemma 1 (*i.e.* when considering 1-input functions), the smaller the intersection between the sets of candidates is, the more leakage we have (since the remaining secrecy corresponds to an intersection of sets generated by these functions). On the other hand, in Lemma 2 (*i.e.* when considering 2-input functions), the smaller these intersections are, the more secrecy we have (since the remaining secrecy corresponds to an union of sets generated by the 2-input function). Looking at present block ciphers, the boolean functions used for *e.g.* the S-boxes generally provide less independence than a random function. This is notably known in the literature as the "ghost peaks" problem. Since the analysis of the DPA is only based on our fifth working assumption (because the only 2-input function considered in the scheme has a known input and is consequently analyzed as a 1-input function), it again corresponds to a worst case prediction. Note that the parameter t depends on this randomness assumption as well. Indeed for taking advantage of multiple leakage points, it is necessary that these leakages give rise to independent information. In practice, this is another reason why t is usually limited to small values.

This discussion leads to the following informal definition:

Definition 8: An exclusive side-channel attack (**xsca**) against a cryptographic implementation is an idealized side-channel attack in which we assume the side-channel oracles to behave like perfect random functions.

This is actually the side-channel counterpart of the random oracle model. Otherwise said, exclusive side-channel attacks can be seen as side-channel attacks in the random side-channel oracle model. Such a context has two important consequences:

1. It allows us to use the previous working assumptions to evaluate the security of idealized implementations against side-channel attacks.
2. It does not include black box attacks nor any possible combination between side-channel and black box attacks.

The second point is fundamental since the objective of our framework is to analyze the security of a construction with respect to side-channel attacks *only*. Of course, for carefully designed algorithms, the expectation is that exclusive side-channel attacks and side-channel attacks will be roughly as powerful. As already mentioned, the algebraic nature of actual block cipher components even involves that the efficiency of exclusive side-channel attacks is an upper bound on what can be obtained in practice as long as the target block cipher does not contain any obvious black box weakness.

Comparing exclusive side-channel attacks with real attacks based on actual statistical tools and applied to concrete block ciphers is another scope for further research.

We next present our definition of security against side-channel attacks.

Definition 9: Let f_K be a cryptographic primitive implementation embedding a secret key K with security parameter k, *i.e.* $K \in \{0,1\}^k$. Let $A^{\tau,q}_{f_K,Ł}(F, L)$ be a PPT algorithm representing a side-channel distinguisher with time complexity τ, having access to q non adaptive black box queries to the primitive f_K and their respective side-channel queries to a leakage function Ł (the black box and side-channel queries being respectively stored in the sets B and S). We consider two experiments:

Experiment $\mathbf{Exp}_0$	Experiment $\mathbf{Exp}_1$	
$K_0 \xleftarrow{R} \{0,1\}^k$	$K_1, K_2 \xleftarrow{R} \{0,1\}^k$	
$B_0 \leftarrow f_{K_0}(.,.,.)$	$B_1 \leftarrow f_{K_1}(.,.,.)$	; black box queries
	$B_2 \leftarrow f_{K_2}(.,.,.)$	; black box queries
$S_0 \leftarrow Ł(B_0)$	$S_2 \leftarrow Ł(B_2)$	; side-channel queries
$d \leftarrow A^{\tau,q}_{f_K,Ł}(B_0, S_0)$	$d \leftarrow A^{\tau,q}_{f_K,Ł}(B_1, S_2)$	
return d	return d	

The advantage of the side-channel distinguisher $A^{\tau,q}_{f_K,Ł}$ is defined as:

$$\mathbf{Adv}^{A^{\tau,q}}_{f_K,Ł} = \mid P[\mathbf{Exp}_0 = 1] - P[\mathbf{Exp}_1 = 1] \mid$$

For any t, q, we define the side-channel advantage of a cryptographic implementation f_K against non adaptive adversaries[6] enhanced with a leakage function Ł as:

$$\mathbf{Adv}^{na-sca}_{f_K,Ł}(\tau, q) = \max_A \{\mathbf{Adv}^{A^{\tau,q}}_{f_K,Ł}\}$$

Informally, the first experiment provides the distinguisher with a set of black box queries and the corresponding set of leakages. The second experiment provides the distinguisher with a set of black box queries generated with a random key K_1 and a set of leakages generated with another random key K_2. The advantage of the adversary is the probability that it can distinguish between these experiments.

[6] Here meaning that the adversaries only perform non adaptive queries.

As an illustration, the previously described attack against the rounds of the block cipher in Figure 2 results in a side-channel advantage: $\mathbf{Adv}^{na-sca}_{f_K,\mathrm{L}}(\tau, q) \leq c_1 \cdot \frac{\tau}{2^k} + c_2 \cdot \frac{1}{2^{k-q\lambda}} + c_3 \cdot \frac{q}{2^{n_{obs}}}$. In this expression, the first term corresponds to an exhaustive key search. In the second term, $\lambda \leq t \cdot l$ is a single trace leakage and the denominator $2^{k-q\lambda}$ relates to the linear bound for the secrecy reduction during the attack. Finally, the third term suggests the possibility of a worst case leakage happening in some query, *e.g.* an "*all zeroes*" Hamming weight that would reveal a secret key in one trace. This situation is actually the side-channel counterpart of an exhaustive key search. It is an important point with respect to our definition of average secrecy since it introduces the possibility of a worst case situation in our analysis, as required in Section 3.4. This term typically indicates that our average analysis is only meaningful as long as the side-channel adversary observes large n_{obs} values.

Definition 10: A cryptographic implementation f_K with security parameter k is $(k - \lambda)$-leakage proof with time complexity τ and q side-channel queries if its side-channel advantage $\mathbf{Adv}^{na-sca}_{f_K,\mathrm{L}}(\tau, q)$ is negligible in $k - \lambda$.

An important point of this definition is that is takes into account the fact that side-channels will *always* involve a reduction of the complexity of a distinguishing attack. A secure implementation is therefore the one for which this reduction of complexity *only* comes from a single trace leakage λ, *i.e.* when having access to multiple traces does not improve the efficiency of the attack (or does it with a negligible effect). Remark that this notion of security is only meaningful with respect to the definition of an adversary's capabilities (*i.e.* for fixed n_{obs} and t values).

As already mentioned, we will investigate the security against exclusive side-channel attacks rather than side-channel attacks. For this purpose, a similar definition can be obtained by just replacing the side-channel advantage function $\mathbf{Adv}^{na-sca}_{f_K,\mathrm{L}}(\tau, q)$ by an exclusive side-channel advantage function $\mathbf{Adv}^{na-xsca}_{f_K,\mathrm{L}}(\tau, q)$. Again, the question of modeling adaptive adversaries is a scope for further research.

We remark that this definition says nothing about the black box security of a cryptographic primitive and therefore corresponds to the notion of *durable function* given by Micali and Reyzin in [26]. According to our definition, proving an implementation to be leakage proof only means that the leakage "does not significantly help" to break it, although the underlying primitive can still be weak with other respects. The reason why we used this notion of security relates to the assumptions that we need to analyze actual primitives like block ciphers. Indeed, what we propose in this paper is:

1. To assume a very strong black box security (*i.e.* exclusive side-channel attacks).
2. To analyze the leakage proofness in this context.

The exclusive side-channel context is therefore required for the purposes of our theoretical analysis. In practice, however, there obviously exist implementations both insecure in the black box model and leakage proof, *e.g.* based on the identity function (*i.e.* where the leakages do not provide any more information than the black box outputs).

Again, it is important to remember that with respect to side-channel attacks only, this exclusive context is conservative since it involves that every leaking point in an implementation and trace in an attack deliver independent information. Otherwise said, *we make a strong black box assumption corresponding to a strong side-channel adversary.* To sum up, in order to analyze side-channel attacks against real world primitives, we had to clearly separate the notion of black box security from the one of leakage proofness. This is what allows us to obtain more positive results, as the next section will emphasize. Therefore, the better a block cipher (and its component functions) compares to its ideal black box counterpart, the more realistic our model is. Note finally that anyway, in the context of block ciphers, the black box security cannot be proven with present theory and therefore has to be assumed.

5 Design

In this section, we apply our tools and definitions in order to investigate the security of two basic primitives against exclusive side-channel adversaries. Specifically, from the block cipher described in Section 3.3, we will construct a pseudo random number generator (PRNG) and a publicly initialized pseudo random number generator (PIPRNG), both leakage proof against (a class of) side-channel attacks. We start by giving some explanations about our proof strategy. Then, we show that our two primitives can be proven secure against $\{n_{obs}, t\}$-limited exclusive side-channel adversaries. Finally, we discuss the practically important case of unlimited adversaries.

5.1 Proof strategy

In the previously defined exclusive side-channel context, it is assumed that there is no possible black box weakness in the side-channel oracles. It involves that the only possible attack allowing to solve the challenge in Definition 9 is a prediction attack: according to one of the leakage functions (*i.e.* F_0 or $F_1(\delta)$), the adversary tries to predict the measured leakage in function of the known input/output pairs that it gets thanks to its black box queries. Then it compares these predictions with the received leakage set and decides which prediction is relevant.

The quality of its decision is directly related to the quality of its predictions and consequently, on the knowledge of the circuit signals. The better an adversary knows about the values manipulated by a device, the better it can predict the leakage. By opposition, if the adversary has only a very limited knowledge of the circuit signals, it cannot predict anything precisely, nor make any decision. As a consequence, if we can prove that, given the leakage function described in this paper, the average secrecy of the inner signals in a circuit cannot be decreased more than by a single trace leakage (plus some negligible leakage), then we have leakage proofness.

Note that, since in practice predicting any intermediate value inside an implementation requires the knowledge of the key, the leakage proofness against exclusive side-channel adversaries actually corresponds to security against a key recovery attack, assuming that there are no other possible attacks.

5.2 A PRNG provably secure against limited side-channel adversaries

We start with the following observation about the (in)security of 2-input b-wise functions. Let $y = f(k, x)$ be such a function with k the secret parameter, x a controllable input and y a leaking output. If an attacker can obtain an arbitrary number of leakages, with different x's and the same k, it can discover the secret parameter. Therefore, a secure encryption scheme requires that any such 2-input function in the scheme never uses twice the same secret value k. We apply this principle, denoted as the *"re-keying design principle"*, to build a PRNG provably secure against side-channel attacks.

Description: The PRNG is illustrated in Figure 6. It is basically a block cipher in CBC mode, without key scheduling, where the bitwise XOR is replaced by a 2-input b-wise function f. The initial input value x_0 is public and fixed for every encrypted block $i \in [0, N]$ (*i.e.* cannot be modified, thanks to Hypothesis 2) and there are r different initial secret round keys denoted as $k_j^0 \in \{0,1\}^n$, $j \in [0, r]$, where $n = n_b = n_k$ is the block size, equals to the key size for simplicity purposes. We also define the middle point as the value $M^i = x^i_{\lfloor r/2 \rfloor + 2}$. The key update procedure is as follows:

$$k_j^{i+1} = f(k_j^i, M^i)$$

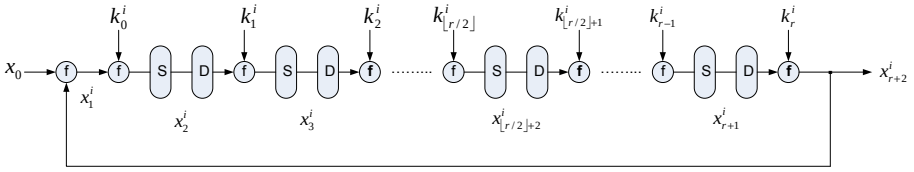

Fig. 6. A PRNG secure against side-channel attacks.

Theorem 1: The PRNG of Figure 6 is $(n-(t+1)\cdot l)$-leakage proof[7] against $\{n_{obs}, t\}$-limited exclusive side-channel adversaries with single point leakage l.

Proof. Initially, the average secrecy of the different parameters is as follows: (1) $\overline{\mathbf{H}}_\propto(x_0^0) = \overline{\mathbf{H}}_\propto(x_1^0) = \overline{\mathbf{H}}_\propto(x_{r+2}^0) = 0$, because the input and output values are known, (2) $\overline{\mathbf{H}}_\propto(k_0^0) = \overline{\mathbf{H}}_\propto(k_1^0) = ... = \overline{\mathbf{H}}_\propto(k_r^0) = n - l$, because we assume that loading the keys is a computation, (3) $\overline{\mathbf{H}}_\propto(x_2^0) = \overline{\mathbf{H}}_\propto(x_3^0) = ... = \overline{\mathbf{H}}_\propto(x_{r+1}^0) \geq n - t \cdot l$, because there are t side-channel oracles per round (*i.e.* between any two 2-input function).

From these leakages, there is an additional average secrecy reduction on k_0^0 and k_r^0 of $t \cdot l$ bits because the corresponding 2-input function has one known input. It yields $\overline{\mathbf{H}}_\propto(k_0^0) = \overline{\mathbf{H}}_\propto(k_0^r) \geq n - (t+1) \cdot l$. For the other round keys, the 2-input function has two partially known inputs of secrecy $n - t \cdot l$ bits each. Therefore, the additional average secrecy reduction is negligible and cannot be used by a limited adversary.

[7] Since the PRNG's input values cannot be controlled by an adversary, it is interesting to note that the PRNG is by nature secure against adaptive adversaries as well.

After the first encryption cycle, the round keys are updated with $k_j^{i+1} = f(k_j^i, M^i)$. In the worst case, we know $t \cdot l$ bits of M^i and $(t+1) \cdot l$ bits of k_j^i (this happens for the first and last round keys). Again, the additional leakage on k_j^{i+1} cannot be exploited by the limited adversary. Therefore, the remaining average secrecy of the updated keys is $n-l-\eta$, where the η is the neglected leakage due to the limited adversary assumption.

Starting the encryption of the second block, the situation is the same as initially, *i.e.* known inputs and the secrecy of each updated key equals to $n-l$, excepted for an additional negligible leakage η. This completes the proof. □

Remark: It is important to remember again that our proofs only consider average secrecy values, which do not prevent from having a worst case happening in some query, *e.g.* the previously mentioned "*all zeroes*" leakage. As the black box security relies on computational assumptions, our definition of leakage proofness relies on "measuremental" assumptions. This means that the PRNG of Figure 6 remains secure as long as the number of side-channel queries q is small with respect to $2^{n_{obs}}$. Therefore, although we use entropy measurements to evaluate the amount of information leaked by an implementation, our proofs provide no information theoretic security overall.

5.3 How to initialize a PRNG securely with a public seed?

In this section, we study the possibility to initialize our PRNG with a public seed in the side-channel context, as usually required in higher-level protocols. Looking at Figure 6, the main constraint is then that the public random seed may be under the control of an adversary. This puts a number of obvious solutions out of the game. For example, it is clear that XORing a random seed to the initial value x_0 is not acceptable. Indeed, it would allow one (by applying repeated reset signals) to obtain an arbitrary number of known inputs of the scheme and therefore to reveal k_0 (and all the other round keys).

Description: A possible solution, illustrated in Figure 7, is to use two initial values x_0 and y_0 (again, fixed $\forall i \in [0, N]$). Then, a n-bit random number r is selected of which we denote the different bits as $r(i)$. This solution holds in two steps:

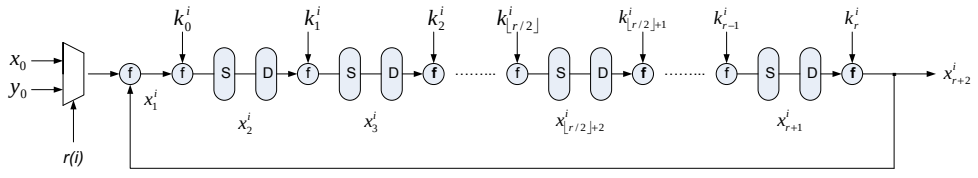

Fig. 7. A PIPRNG secure against side-channel attacks.

1. Initialization: during the initialization, we encrypt n blocks with the initial value selected as follows: $x_1^i = f(x_0, x_{r+2}^{i-1})$ if $r(i) = 0$, else $x_1^i = f(y_0, x_{r+2}^{i-1})$. Nothing is outputted during initialization.
2. Generation: after the initial n encrypted blocks, the PIPRNG goes on encrypting new blocks, but the x_{r+2}^i's are now outputted.

Theorem 2: The PIPRNG of Figure 7 is $(n - (t+1) \cdot 2 \cdot l)$-leakage proof against $\{n_{obs}, t\}$-limited exclusive side-channel adversaries with single point leakage l.

Proof. For every possible round key k_j^i, and function $x_{j+2}^j = f(k_j^i, x_{j+1}^i)$, there are only two possible values for the x_j^i's, corresponding to $r(j) = 0$ or 1. As a consequence, the previous proof (for the secure PRNG) still holds with a single point leakage of $2 \cdot l$ in place of l. We mention that the PIPRNG is correctly randomized at the end of the initialization period, as there are 2^n possible values for x_{r+2}^n at this time. □

Implementation issues: Different variants of the previously described primitives can be considered in order to allow efficient sequential or parallel implementations. The only constraint for keeping a valid proof is that any 2-input function in the scheme is never used more than once (for the secure PRNG) or twice (for the PIPRNG) with the same key. Note that for pipelined PRNGs, it is additionally required that the feedback is only activated once the pipeline is fulfilled (*i.e.* once the middle point is significant).

5.4 Unlimited adversaries

The previous section proved the security of two primitives against limited exclusive side-channel adversaries. It is shown that multiple trace leakages can be kept as small as a single trace leakage plus an additional negligible leakage. However, as already discussed, practical adversaries are unlimited. As a consequence, this section finally investigates the context of unlimited adversaries, for which the leakages of 2-input functions are not considered to be negligible anymore. For this purpose, we put forward the two assumptions allowing to expect the practical security of our PRNGs in this context.

Measuremental assumptions and the security of 2-input functions: Looking back at Figure 5, a first observation is that there are basically two ways to reduce the additional leakage caused by a two-input function with partially known inputs: either by increasing the block size n or by reducing the inputs leakage per bit l/n. Therefore, an interesting question is to know how the security of an implementation against unlimited side-channel adversaries depends on the single point leakages, for a fixed n value.

For this purpose, we go back to the proof of Theorem 1. More specifically, let us review again the key update procedure $k_j^{i+1} = f(k_j^i, M^i)$ for the key k_0^i (*i.e.* the worst case). At this point the respective leakages on the inputs of the 2-input function are $t \cdot l$ and $(t+1) \cdot l$. Clearly, if these actual input leakages are sufficiently small, then the additional leakage on the 2-input function's output will be even smaller. Otherwise said, rather than being really negligible, it may happen that in practice, this additional leakage is sufficiently small for being practically un-exploitable.

The question is then obviously to know what are the acceptable input leakages that can be considered as un-exploitable. For example, using our sixth working assumption (see Section 4.1), we have that a 2-input function $z = f(x, y)$ with respective leakages on x, y of l_x, l_y has an average secrecy reduction on z bounded to $\frac{l_x \cdot l_y}{n}$ bits. It clearly suggests that as long as we have $l_x \cdot l_y << n$, the additional leakage on the output of a 2-input function with partially known inputs will be less than one bit on average. In general, this additional leakage can be further decreased by increasing the size of an implementation, *i.e.* applying the bit-chunk design principle.

One more time, let us emphasize that the fundamental factor to obtain the leakage proofness in our implementations is that, during the key update procedure $k_j^{i+1} = f(k_j^i, M^i)$, the leakage only comes from the computation of k_j^{i+1} and not from partial knowledge on k_j^i, M^i. With respect to the previous discussion, this is obviously not formally correct. Indeed, the average leakage on an updated key is always increased by the partial knowledge of its generators. Assuming these leakages to be negligible is equivalent as to say that higher-order attacks are not applicable in our context. In fact, our expectation is that under the measuremental assumption that the single point leakages are small, the information delivered by a 2-input function will be hardly exploitable, as suggested by the limited adversary abstraction and therefore improve practical security (typically as countermeasures such as making do). As a matter of fact, this is only a qualitative argument and its quantitative evaluation is an additional scope for further research. The next assumption is therefore of bigger importance to justify the security against unlimited adversaries.

Computational assumptions and the security of the middle point: In addition to the previous measuremental assumptions, one may finally observe that considering the additional leakage of a 2-input function with partially known inputs to be negligible also relates to a reasonable computational assumption. In practice, the strategy of most side-channel attacks is based on the key guess strategy. That is, the useful leakages are the ones for which the related data only depends on a limited number of key bits. This is typically the case when applying a DPA-like attack as in Section 3.2. Because only the first block cipher round is targeted, the diffusion is not yet complete and *e.g.* the S-boxes outputs only depend on 8 key bits. Therefore, their leakage can be predicted. On the other hand, when using a 2-input function for the key update procedure in our PRNGs, the inputs are either depending on the whole key because of the diffusion within the cipher (*i.e.* for M^i) or a priori unknown (*i.e.* for the k_j^i's). Therefore, the possible additional leakage on k_j^{i+1} is not easily exploitable, for computational reasons. Exploiting these leakages actually corresponds to a DPA-like attack where the adversary directly targets, *e.g.* the middle round of a block cipher. Otherwise said, by finally removing our fourth working assumption (see Section 3.5) and reasonably restricting the adversaries measurement capabilities (*i.e.* the second input of Micali and Reyzin's leakage function) to computable key guess strategies, it is expected that one can obtain leakage proofness in practice. Note finally that this last discussion adds an actual design requirement for the middle point in our PRNGs. That is, the diffusion within the cipher has to be complete at this point.

5.5 A short note on template attacks

In template attacks [9], an adversary initially tries to build a model for the leakage of a device and then uses this model to target similar implementations. The leakage proof implementations described in this paper are the ones for which the leakage is not sufficient to significantly reduce the secrecy of its secret parameter. Therefore, in the context of template attacks, a leakage proof implementation is the one for which the adversary does not have sufficient information to build a precise template for any (part) of the secret parameters in a target device.

6 Concluding remarks

A new framework in accordance with the physical reality of side-channel attacks is proposed. It is based on average evaluations of certain entropy measurements on the inner signals in a cryptographic implementation. The model allowed us to:

1. Derive practical analysis tools for side-channel attacks,
2. Evaluate existing attacks such as the Differential Power Analysis,
3. State a formal definition of security against side-channel adversaries,
4. Prove the security of two cryptographic primitives against such adversaries.

Specifically, our analysis considers a model of "measuremental complexity", which is the side-channel counterpart of the computational complexity. We first show that, under certain hypotheses detailed in the paper, previously investigated side-channel attacks, such as the Differential Power Analysis, have a linearly bounded complexity in the number of side-channel queries obtained by an adversary. Then we formally define the notion of leakage proof implementations (*i.e.* intuitively, implementations for which the physical leakages are not exploitable by an adversary). Finally, we prove two new constructions to be leakage proof against the so-called "limited side-channel adversaries". Under a reasonable combination of measuremental and computational assumptions, it is argued that such proofs provide real-world security against actual side-channel attacks.

More fundamentally, we consider the black box security and the physical security as two separate worlds and try to protect against attacks taking exclusively advantage of physical leakages, not of possible black box weaknesses. For carefully designed block ciphers, it is expected that the hypothesis is reasonable. Positively, if a side-channel attack targets our provably secure designs, it means either a "black box weakness" in the underlying primitive, or that the security parameter (roughly depending on the amount of information leaked by a single measurement) chosen was too low.

Finally, from a practical point of view, this work determines how the size of an implementation relates to the amount of information leaked during a computation. In this information theoretic framework, large parallel implementations will be preferred to small devices, *e.g.* using 8-bit data buses. This also implies that adversaries's capabilities should, as far as possible, be limited to the observation of global leakages (*e.g.* the power consumption) rather than space-localized emanations (*e.g.* near field EM radiations) or time-localized emanations (*e.g.* using the leakages phases to discriminate the effect of single bits in an implementation). As a matter of fact, in these latter contexts, it is likely that the proposed designs would not provide enough security anymore, because too much entropy would be leaked.

7 Open research problems

A first (and main) direction would be to study the practical consequences of the *hypotheses* and *assumptions* made in the paper. This includes the following questions:

1. How to relax our various working assumptions? A straightforward problem would be to further specialize our model to specific instances of cryptographic functions and derive tighter bounds of security. That is, to compute exact expressions for Lemmas 1 and 2 rather than bounding them with $WA.5$ and $WA.6$. Another exemplary issue is to introduce noise in the model.
2. Can an adversary be more effective than what is measured by our definition of average secrecy? This involves the investigation of particular attack contexts, *e.g.* the use of adaptive chosen plaintexts.
3. How meaningful is the exclusive context?
   (a) Comparisons between random side-channel oracles and actual block cipher components must be performed. This involves to compare our information theoretic bounds of security with practical attacks based on actual statistical tools.
   (b) Recent publications (*e.g.* in [27]) suggest the combination of side-channel and black box attacks that are typically not considered in the exclusive context and therefore require further analysis.
4. How meaningful is a limited adversary? Side-channel attacks against the proposed PRNG and PIPRNG for unlimited adversaries have to be evaluated in order to determine precisely how small must the leakage be to obtain actual security.

A second direction would be to further investigate the *construction* of leakage proof implementations and primitives. This includes the following problems:

1. Construction of an actual instance of PRNG, based on present block cipher components and using the design principles proposed in this paper, with cryptanalysis and other verifications of the black box properties (*e.g.* statistical tests).
2. Construction of other leakage proof cryptographic primitives. A first step would be to build an encryption function. Longer term research should (if possible) evaluate the possibility to build leakage proof asymmetric primitives (for which operation leakages have to be considered).

A third direction would be to *expand the model* as far as it is possible. This notably includes the following possibilities:

1. Considering other leakage functions than the simple CMOS-based models proposed.
2. Considering all the physical threats pictured in Figure 1, *i.e.* operation leakages, fault attacks, probing attacks and possibly others.

Finally, practical *implementations* should be considered and evaluated.

Acknowledgements: The authors would like to acknowledge Leonid Reyzin for his valuable contributions to the development and understanding of the issues discussed in the paper. We also would like to thank Cédric Archambeau, Nenad Dedic, Marc Joye and Eric Peeters for their helpful comments about preliminary versions of this work. François-Xavier Standaert is a post doctoral researcher funded by the FNRS (Funds for National Scientific Research, Belgium).

References

1. D. Agrawal, B. Archambeault, J. Rao, P. Rohatgi, *The EM Side-Channel(s)*, in the proceedings of CHES 2002, Lecture Notes in Computer Science, vol 2523, pp 29-45, Redwood City, California, USA, August 2002.
2. D. Agrawal, J. Rao, P. Rohatgi, *Multi-channel Attacks*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 2-16, Cologne, Germany, Sept. 2003.
3. M.L. Akkar, C. Giraud, *An Implementation of DES and AES Secure againts Some Attacks*, in the proceedings of CHES 2001, Lecture Notes in Computer Sciences, vol 2162, pp 309-318, Paris, France, May 2001.
4. R. Anderson, M. Kuhn, *Tamper Resistance - a Cautionary Note*, in the proceedings of the USENIX Workshop on Electronic Commerce, pp 1-11, Oakland, USA, Nov. 1996.
5. E. Brier, C. Clavier, F. Olivier, *Correlation Power Analysis with a Leakage Model*, in the proceedings of CHES 2004, Lecture Notes in Computer Science, vol 3156, pp 16-29, Boston, USA, August 2004.
6. D. Boneh, R. DeMillo, R. Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*, proceedings of Eurocrypt 1997, Lecture Notes in Computer Science, vol 1233, pp 37-51.
7. X. Boyen, *Reusable Cryptographic Fuzzy Extractors*, in the proceedings of the ACM Conference on Computer and Communications Security, pp 82-91, Washington DC, USA, October 2004.
8. S. Chari C. Jutla, J. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power-Analysis Attacks*, in the proceedings of Crypto 1999, Lecture Notes in Computer Science, vol 1666, pp 398-412, Santa Barbara, California, USA, August 1999.
9. S. Chari, J. Rao, P. Rohatgi, *Template Attacks*, in the proceedings of CHES 2002, Lecture Notes in Computer Science, vol 2523, pp 13-28, Redwood City, CA, USA, August 2002.
10. J.S. Coron, P. Kocher, D. Naccache, *Statistics and Secret Leakage*, in the proceedings of Financial Crypto 2000, Lecture Notes in Computer Science, vol 1972, pp 157-173, Anguilla, British West Indies, February 2000.
11. C. Clavier, J.S. Coron, N. Dabbous, *Differential Power Analysis in the Presence of Hardware Countermeasures*, in the proceedings of CHES 2000, Lecture Notes in Computer Sciences, vol 1965, pp 252-263, Worcester, Massachusetts, USA, August 2000.
12. J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestre, J.-J. Quisquater, J.-L. Willems, *A Practical Implementation of the Timing Attack*, in the proceedings of CARDIS 1998, Lecture Notes in Computer Science, vol 1820, pp 167-182, Louvain-la-Neuve, Belgium, 1998.
13. Y. Dodis, L. Reyzin, A. Smith, *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*, in the proceedings of Eurocrypt 2004, Lecture Notes in Computer Science, vol 3027, pp 523-540, Interlaken, Switzerland, May 2004.
14. FIPS 197, *"Advanced Encryption Standard,"* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
15. J.A. Fournier, S. Moore, H. Li, R.D. Mullins, G.S. Taylor, *Security Evaluation of Asynchronous Circuits*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 137-151, Cologne, Germany, September 2003.
16. K. Gandolfi, C. Mourtel, F. Olivier, *Electromagnetic Analysis: Concrete Results*, in the proceedings of CHES 2001, Lecture Notes in Computer Science, vol 2162, pp 251-261, Paris, France, May 2001.
17. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, *Algorithmic Tamper-Proof Security: Theoretical Foundations for Security Against Hardware Tampering*, in the proceedings of TCC 2004, Lecture Notes in Computer Science, vol 2951, pp 258-277, Cambridge, MA, USA, February 2004.
18. L. Goubin, J. Patarin, *DES and Differential Power Analysis*, in the proceedings of CHES 1999, Lecture Notes in Computer Science, vol 1717, pp 158-172, Worcester, Massachussets, USA, August 1999.

19. S. Guilley, P. Hoogvorst, R. Pacalet, *Differential Power Analysis Model and Some Results*, in the proceedings of CARDIS 2004, pp 127-142, Toulouse, France, Kluwer 2004.
20. P. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems*, in the proceedings of Crypto 1996, Lecture Notes in Computer Science, vol 1109, pp 104-113, Santa Barbara, California, USA, August 1996.
21. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, in the proceedings of Crypto 1999, Lecture Notes in Computer Science, vol 1666, pp 398-412, Santa-Barbara, USA, August 1999.
22. S. Mangard, *Hardware Countermeasures against DPA - A Statistical Analysis of Their Effectiveness*, in the proceedings of CT-RSA 2004, Lecture Notes in Computer Science, vol 2964, pp 222-235, San Francisco, USA, February 2004.
23. D. May, H. Muller, N. Smart, *Randomized Register Renaming to Foil DPA*, in the proceedings of CHES 2001, Lecture Notes in Computer Sciences, vol 2162, pp 28-38, Paris, France, May 2001, Springer-Verlag.
24. T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant Software*, in the proceedings of CHES 2000, Lecture Notes in Computer Sciences, vol 1965, pp 71-77, Worcester, Massachusetts, USA, August 2000.
25. T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Examining Smart-Card Security under the Threat of Power Analysis Attacks*, IEEE Transactions on Computers, vol 51, num 5, pp 541-552, May 2002.
26. S. Micali, L. Reyzin, *Physically Observable Cryptography*, in the proceedings of TCC 2004, Lecture Notes in Computer Science, vol 2951, pp 278-296, Cambridge, MA, USA, February 2004.
27. F. Muller, *Analyse d'Algorithmes en Cryptographie Symtrique*, PhD Thesis, Sept. 2005.
28. E. Peeters, F.-X. Standaert, J.-J. Quisquater, *Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons*, to appear in Integration, the VLSI Journal, Spring 2006, Elsevier.
29. E. Peeters, F.-X. Standaert, N. Donckers, J.-J. Quisquater, *Improved Higher-Order Side-Channel Attacks With FPGA Experiments*, in the proceedings of CHES 2005, Lecture Notes in Computer Science, vol 3659, pp 309-323, Edinburgh, Scotland, September 2005.
30. A. Pfitzmann, R. Aßmann *More Efficient Software Implementations of (Generalized) DES*, Internal Report 18/90, University of Karlsruhe, May 1990.
31. J.-J. Quisquater, D. Samyde, *ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards*, in the proceedings of E-smart 2001, Lecture Notes in Computer Science, vol 2140, pp 200-210, Cannes, France, September 2001.
32. J.M. Rabaey, *Digital Integrated Circuits*, Prentice Hall International, 1996.
33. G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat, *Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications*, in the proceedings of ITCC 2004, Las Vegas, USA, April 2004.
34. A. Shamir, *Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies*, in the proceedings of CHES 2000, Lecture Notes in Computer Sciences, vol 1965, pp 238-251, Worcester, Massachusetts, USA, August 2000.
35. C. E. Shannon, *Communication theory of secrecy systems*, Bell System Technical Journal, vol 28, pp 656-715, October 1949.
36. F.-X. Standaert, S.B. Ors, B. Preneel, *Power Analysis of an FPGA Implementation of Rijndael: is Pipelining a DPA Countermeasure?*, in the proceedings of CHES 2004, Lecture Notes in Computer Science, vol 3156, pp 30-44, Boston, USA, August 2004.
37. K. Tiri, M. Akmal, I. Verbauwhede, *A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*, in the proceedings of ESSCIRC 2003.
38. J. Waddle, D. Wagner, *Towards Efficient Second-Order Power Analysis*, in the proceedings of CHES 2004, Lecture Notes in Computer Science, vol 3156, pp 1-15, Boston, USA, August 2004.

A Average minimum entropy of a 2-input function

We first need the following result. If $z = f(x, y)$ is an n-bit 2-input function and N_x, N_y respectively denote the number of possible x and y values, then the probability of the most likely z value is worth $P_\infty(z = f(x, y), N_x, N_y) = \frac{\min(N_x, N_y)}{N_x N_y}$. This expression just corresponds to the probability of the z candidate with the highest multiplicity, since $f_X(y)$ and $f_Y(x)$ are both bijections. It yields the following lemma.

Lemma 2bis: Let f be a 2-input n-bit function. Let x and y be two values computed in a leaking device with $n_{obs} = n$ and the leakages on x, y be measured with the Hamming weight function. Let $z = f(x, y)$ be a *non-leaking* computation (meaning that there is no direct leakage on z but only on x, y). The avg. min. entropy of z is:

$$\overline{\mathbf{H}}_\infty[z|z = f(x, y), W_H(x), W_H(y)] = -\log_2 \sum_{h_x=0}^{n} \sum_{h_y=0}^{n} \frac{\binom{n}{h_x}}{2^n} \frac{\binom{n}{h_y}}{2^n} \cdot \frac{1}{\max\left(\binom{n}{h_x}, \binom{n}{h_y}\right)}$$

□

This expression simply derives from the definition of minimum entropy and the previous expression for the probability of the most likely z value.

In fact, this Lemma is too pessimistic with respect to what an actual side-channel adversary can achieve, as the following observations underline:

1. Lemma 2bis corresponds to a strategy where the adversary would select the most likely key *only*. However, in practice, the strategy of a side-channel attack is to reduce the secrecy of the cryptographic keys by combining multiple leakages. That is, in our example where the possible z values are $\{0, 1, 1, 2, 4, 5, 5, 6\}$, a side-channel adversary would not reject the candidates 0,2,4,6 because they are less likely.
2. In a side-channel attack, there are no a priori means (before any leakage has been obtained) allowing an adversary to force a worst case such that, in our example, $z = 1$ or 5. This is because 2-input functions are generally used to combine the cipher state with (secret) key material. Therefore, it is natural to consider the average probability to predict z rather than a worst-case scenario as the average minimum entropy involves. Since we only consider non adaptive adversaries, the same observation remains true once a number of leakages have been observed.
3. A practical consequence of Lemma 2bis is that the minimum entropy of z is determined by only one of the function's inputs. However, it is intuitively clear that in practice, both the uncertainty on x and y should increase the uncertainty on z.

The definition of average secrecy therefore better corresponds to the actual behavior of such an adversary. This justifies our hypothesis in Section 3.4.