Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions

François-Xavier Standaert[1], Olivier Pereira[1], and Yu Yu[2]

[1] ICTEAM/ELEN/Crypto Group, Université Catholique de Louvain, Belgium
[2] East China Normal University and Tsinghua University, China

Abstract. Leakage-resilient cryptography aims at formally proving the security of cryptographic implementations against large classes of side-channel adversaries. One important challenge for such an approach to be relevant is to adequately connect the formal models used in the proofs with the practice of side-channel attacks. It raises the fundamental problem of finding reasonable restrictions of the leakage functions that can be empirically verified by evaluation laboratories. In this paper, we first argue that the previous "bounded leakage" requirements used in leakage-resilient cryptography are hard to fulfill by hardware engineers. We then introduce a new, more realistic and empirically verifiable assumption of simulatable leakage, under which security proofs in the standard model can be obtained. We finally illustrate our claims by analyzing the physical security of an efficient pseudorandom generator (for which security could only be proven under a random oracle based assumption so far). These positive results come at the cost of (algorithm-level) specialization, as our new assumption is specifically defined for block ciphers. Nevertheless, since block ciphers are the main building block of many leakage-resilient cryptographic primitives, our results also open the way towards more realistic constructions and proofs for other pseudorandom objects.

Introduction

Physical cryptanalysis is an important concern for cryptographic implementations. By allowing to circumvent the models in which standard security proofs are obtained, it can lead to powerful attacks (e.g. key recoveries) against large classes of devices. Following the publications of papers about side-channel [24], fault [6] or cold boot attacks [16], a large body of research has investigated solutions to mitigate these security breaches. For this purpose, a natural solution is to add protection mechanisms directly at the hardware level (i.e. independent of the algorithm implemented). Examples of such approaches include masking and hiding against side-channel attacks [26], error-detection codes against fault attacks [20], and their formal extensions as compilers (e.g. [18,19]) - leading to contrasted observations. On the one hand, these countermeasures are useful as they reduce the amount of information leakage provided by physical implementations. On the other hand, they usually imply significant performance overheads, and the security they provide is highly dependent on technological assumptions (that may turn out to be contradicted in practice). Over the years, and starting

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 335–352, 2013.

with the seminal work of Micali and Reyzin [27], the question whether a complementary approach exploiting the formalism of modern cryptography could be used in order to improve physical security consequently triggered the interest of many researchers. In other words, can we design new cryptographic constructions and security models in which the guarantees of provable security can be extended from mathematical objects towards physical ones? And are the results obtained in these models practically relevant (in terms of performance and security)?

Related Work. A look at the recent literature suggests that a wide variety of tools aiming at reflecting different classes of physical attacks exist, ranging from quite abstract to more realistic, and for various types of cryptographic primitives. For example, the bounded retrieval model captures an hypothetical situation in which the total amount of information leaked through the execution of a cryptographic primitive is bounded [1,3]. One important drawback of this abstraction is that quantifying an "overall amount of leakage" is hard for hardware engineers. Besides, if a system is being used continually for a sufficiently long period of time, the amount of leakage observed by the attacker may exceed any a-priori determined leakage bound. As a result, alternative models have been proposed, assuming that the leakage rate is bounded and leaving the overall leakage arbitrarily large, e.g. Dziembowski and Pietrzak's leakage-resilient cryptography [12]. These models have been applied for analyzing different cryptographic primitives, including PRGs and stream ciphers [13,30,38,39], PRFs and PRPs [11,13,38], signature schemes (e.g. [7,21]) and public-key encryption (e.g. [2,22]).

Are We Done? Not Really. Despite significant progresses and many clever design ideas, the fundamental problem of formal approaches to physical security remains to determine reasonable restrictions of the leakage function. Even taking the simple(st) example of leakage-resilient PRGs (that will be our main concern in this work), obtaining security proofs in the standard cryptographic setting turns out to be surprisingly difficult [12,13,30,38,39]. Intuitively, the proofs obtained so far require a combination of seemingly too weak assumptions (e.g. that the leakage may come from any polynomial time function) and seemingly too strong assumptions (e.g. that the information leakage is bounded in a somewhat unrealistic manner) [35]. Consequences of this imperfect modeling are threefold. First, it implies design tweaks that seem motivated by proof artifacts more than physical intuition, and consequently harm performances. Second, obtaining the proofs requires intricate (though sometimes of independent interest) mathematical tools, usually leading to loose security bounds. Third and most importantly, it leaves the question of how to connect the results in leakage-resilient cryptography with the practice of side-channel attacks essentially open.

Our Contribution. In this paper, we start by investigating the relevance of different bounded leakage assumptions. In particular, we confront the notion of HILL pseudoentropy used to prove the leakage-resilience of previous PRGs, PRFs and PRPs to the operation of actual side-channel attacks, and argue that it is hard to verify empirically. We then tackle our main problem, i.e. the construction of a leakage-resilient PRG based exclusively on empirically verifiable

assumptions. For this purpose, our central ingredient is the introduction of a specialized assumption of simulatable leakage. We first show that this requirement is easier to guarantee than maintaining a high pseudoentropy in a leaking device, and detail how it can be tested in actual security laboratories. Next, we show the security of an efficient leakage-resilient PRG under simulatable leakage. Eventually, we put forward that our new modeling allows mitigating the three issues listed in the previous paragraph. In particular, it allows major simplifications of the proofs, with reductions directly connected to our physical assumption (i.e. the quality of the simulator). From a methodological point of view, the idea of specialized assumption that we introduce can be seen as an intermediate path, between fully generic requirements (e.g. bounded leakage that applies to any algorithm) and implementation-specific ones (e.g. as used in hardware-level countermeasures). More precisely, our simulatable leakage assumption is specialized at the algorithm level, and applies to any block cipher. We believe this intermediate path is interesting, as it allows a better connection between the theory and practice of side-channel attacks. It is also general enough for being potentially applicable to the many other symmetric cryptographic primitives.

1 Previous Leakage Assumptions

In this section, we analyze different assumptions that have been introduced in previous works, in order to bound the informativeness of a leakage function. For this purpose, we start by providing a description of leakage traces, as they are obtained from the power consumption or electromagnetic radiation of actual cryptographic devices. We then argue that assuming a leakage function with bounded range, or assuming that the secrets manipulated by a leaking device have high pseudoentropy, is hardly realistic. By contrast, we list a few alternative assumptions that are more in line with what hardware designers try to guarantee, based on unpredictable cipher outputs or hard-to-invert leakage functions.

1.1 Actual Leakage Traces

We will consider the AES Rijndael as a case study. Note however that the observations in this section hold for any block cipher. In this context, let us denote the encryption of a plaintext x under a key k giving rise to a leakage trace $\mathbf{l}$ as $y = \mathsf{AES}_k(x) \rightsquigarrow \mathbf{l}$. Since the AES is made of ten rounds, we further denote the application of these rounds and their corresponding subtraces as $x_{i+1} = \mathsf{R}_{k_i}(x_i) \rightsquigarrow l_{i+1}$, where the initial state is given by the plaintext $x_0 = x$ and the ciphertext is given by the final state $y = x_{10}$. That is, a full leakage trace is a vector containing all the rounds subtraces: $\mathbf{l} = [l_0, l_1, l_2, \ldots, l_9, l_{10}]$. For illustration, we provide a leakage trace obtained from a hardware implementation of the AES in Figure 1, where each sample can be written as $l_i(t) = \mathsf{L}_{i,t}(k, x, \rho)$, with ρ a parameter representing the physical randomness (aka noise) in the measurements [27]. In practice, it frequently turns out that the leakage produced when generating an intermediate state x_i can be approximated by the sum of a polynomial function of the bits of x_i (denoted as $x_i[j]$) and some noise [32]:

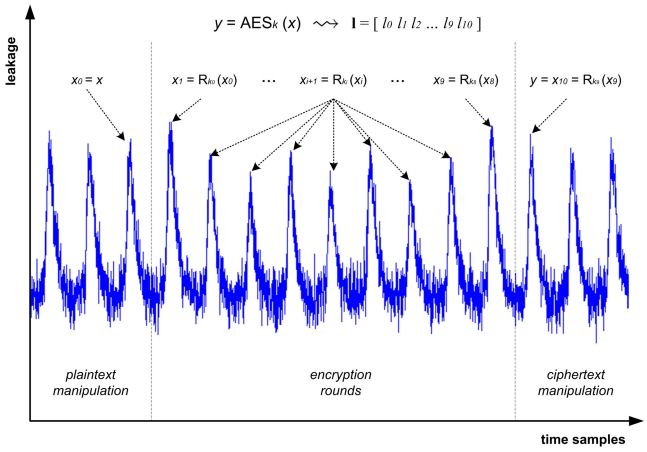

Fig. 1. Exemplary leakage trace of an AES encryption

$$l_i(t) = \mathsf{L}_{i,t}(k, x, \rho) \approx \sum_j \alpha_j \; x_i[j] + \sum_{j_1 \neq j_2} \beta_{j_1,j_2} \; x_i[j_1] \; x_i[j_2] + \ldots + \rho_{i,t}. \qquad (1)$$

In the following, and in order to simplify our discussions, we will further assume that each subtrace is made of a single sample (pointed by the arrows in the figure) that can be written as $l_i(k, x) = \mathsf{HW}(x_i) + r_i^{\mathrm{ph}}$, with HW the Hamming weight function and r_i^{ph} a Gaussian distributed physical noise. Note that these are usual assumptions is side-channel attacks [26]. Yet, also keep in mind that we can only loose information by doing this, and that actual adversaries may be more powerful. Summarizing, we will consider illustrative leakage traces defined as:

$$\mathbf{l} = [\mathsf{HW}(x_0) \; \mathsf{HW}(x_1) \ldots \mathsf{HW}(x_{10})] + [r_0^{\mathrm{ph}} \; r_1^{\mathrm{ph}} \ldots r_{10}^{\mathrm{ph}}]. \qquad (2)$$

1.2 Bounded Range and HILL Pseudoentropy

One of the most demanding assumption regarding the informativeness of the leakage function is the requirement that its range is bounded to $\{0,1\}^\lambda$, for some parameter λ. Taking the example of a leaking block cipher implementation as in the previous subsection, it is easy to observe that a bounded range is hardly obtained. Starting with our simplifying Hamming weight assumption and considering an n-bit key, we already have that each of the N_r Hamming weights in the trace has range $\approx \log(n)$, leading to an output range proportional to $N_r \cdot \log(n)$ (with N_r the number of block cipher rounds). Then, keeping in mind that the number of samples monitored by an oscilloscope in actual attacks is much larger than N_r, it turns out that the range of the leakage function is frequently larger than $\{0,1\}^n$. In practice, this large range is directly reflected by the memory requirements needed to store the measurements. For example, in a recent work from Eurocrypt 2012, Moradi acquired 200 000 traces, each of

them corresponding to 1μs of power consumption leakage sampled at roughly 10^9 samples per second, i.e. leading to more than 1.5 Gigabits of data storage [28].

Previous works in leakage-resilient cryptography (e.g. [11,12,13,30,39]), it is argued that the bounded range assumption can be relaxed. Loosely speaking, these previous proofs only require that for every key update $k_{j+1} = \mathsf{AES}_{k_j}(x) \rightsquigarrow \mathbf{l}$, the leakage $\mathbf{l}$ does not decrease the HILL pseudoentropy of the updated state k_{j+1} by more than a bounded amount of bits. It is further claimed in [22] that such a requirement seems much more realistic in practice. Unfortunately, a look at our example suggests the opposite. Having a pseudoentropy of $n - \lambda$ bits for k_{j+1} requires that there exist a dense set of $2^{n-\lambda}$ keys $\tilde{k}$ that no efficient distinguisher is able to tell apart from k_{j+1} given $\mathbf{l}$. But again considering that the leakage trace contains a sequence of (pseudorandom) Hamming weights, the number of keys $\tilde{k}$ that give rise to the correct sequence of Hamming weights rapidly vanishes, roughly decreasing the pseudoentropy of k_{j+1} according to $n - N_r \cdot \log(n)$. Of course, the high pseudoentropy requirement is weaker than the bounded range assumption. For example, having multiple correlated samples in the traces would not significantly decrease the pseudoentropy of k_{j+1}, while it would increase the output range of the leakage function. Yet, falsifying the pseudoentropy assumption simply requires that an adversary can check whether the trace $\mathbf{l}$ is consistent with the actual k_{j+1}, allowing him to efficiently distinguish it from most fake $\tilde{k}$'s.

Summarizing, while these simple examples exclude the additional randomness due to physical noise, they clearly suggest that maintaining high pseudoentropy in a leaking device is challenging. Interestingly, this observation nicely connects with the conclusions of Micali and Reyzin, who showed the non-equivalence between unpredictability and indistinguishability in physically observable cryptography [27]. We argue in the next subsections that also in terms of practical assumptions, unpredictability is arguably easier to guarantee.

1.3 Side-Channel Attacks

Most distinguishers published in the literature are based on a divide-and-conquer strategy, where independent pieces of a masker key (denoted as subkeys) are recovered independently. Examples include Kocher et al.'s differential power analysis [24], Gandolfi et al.'s electromagnetic analysis [14], Chari et al.'s template attacks [9], Brier et al.'s correlation power analysis [8], Schindler et al.'s stochastic approach [32], Gierlichs et al.'s mutual information analysis [15] and many variations. These attacks are "standard DPAs" in the sense defined by Mangard et al. [25], and operate according to the three following steps:

1. *Prediction.* The adversary predicts subkey-dependent intermediate values manipulated during the encryption process (e.g. a 1^{st}-round S-box output).
2. *Modeling.* The adversary models the leakage corresponding to these intermediate values (e.g. assuming it depends on the HW of the manipulated data).
3. *Comparison.* The adversary compares the subkey-dependent models with actual measurements (e.g. with Pearson's correlation coefficient). If the attack is successful, the best comparison holds for the correct subkey candidate.

The result of a standard DPA attack against the AES usually corresponds to 16 lists of 256 scores (typically proportional to subkey likelihoods), that are then recombined to obtain a master key candidate, e.g. using key enumeration [36].

One consequence of this description is that actual adversaries are usually not able to exploit all the leakage samples in a trace. In practice, only the intermediate computations that can be guessed will be useful. Taking our AES example again, it means that out of a vector $\mathbf{l} = [l_0, l_1, l_2, \ldots, l_9, l_{10}]$, only the external rounds are exploited (i.e. before the diffusion is complete). Furthermore, considering an attack exploiting the first-round leakage l_1, and under our current assumption that the AES is implemented in 10 clock cycles, we have:

$$l_1 = \mathsf{HW}(x_1) + r_1^{\mathrm{ph}} = \mathsf{HW}(x_1[0]) + \mathsf{HW}(x_1[1]) + \ldots + \mathsf{HW}(x_1[15]) + r_1^{\mathrm{ph}}, \quad (3)$$

where $x_1[i]$ denotes the ith byte of x_1. But actual adversaries are not able to guess all the 16 bytes of x_1 at once either. As a result, a part of this information is usually considered as "algorithmic noise". That is, in a (usual) attack where the 16 AES key bytes are targeted independently, the leakage sample l_1 as seen by the adversary can be rewritten as:

$$l_1^{\mathrm{adv}}[0] = \mathsf{HW}(x_1[0]) + \underbrace{\mathsf{HW}(r_1) + \ldots + \mathsf{HW}(r_{15})}_{\text{algorithmic noise}} + r_1^{\mathrm{ph}}, \quad (4)$$

when targeting the first key byte, whith the r_i's uniformly random unknown bytes (a similar equation holds for all the key bytes). In other words, only a single (or at most a couple of) byte Hamming weight(s) is (are) actually considered as useful signal at a time in this computationally bounded setting.

1.4 One-Way and Seed-Preserving Leakage Functions, Unpredictability

The previous description allows shedding another light on why ensuring high pseudoentropy for cryptographic keys in leaking devices is challenging. The main issue is that it requires that these keys remain difficult to distinguish in front of an adversary who can predict the whole device state (hence, exploit the full vector of Equation 2 rather than the noisy samples of Equation 4). In fact, this task is arguably more difficult than the (already difficult) task of securing an implementation against standard DPA attacks. Therefore, we can at least claim that constructions that strictly need this assumption to hold for being secure are not going to "help hardware designers", as usually advertised by leakage-resilient cryptography. This observation naturally provides a strong incentive to look at alternative assumptions that could be easier to fulfill and evaluate.

In general, a weaker assumption than the high HILL pseudoentropy requirement is that the leakage function is hard-to-invert, or that the key/seed is computationally infeasible to predict given the leakage (see [4,17] for relations between several forms of pseudoentropy). This is easily seen the minimal assumption

since no security is possible if the adversary can recover the key/seed. It is also directly connected to the practice of side-channel attacks that usually aim to predict keys/seeds. Unfortunately, how to build leakage-resilient symmetric cryptographic primitives under such assumptions remains an open problem. So far, only some weaker forms of security results have been obtained in this case, such as the encryption schemes of [10] in the auxiliary-input setting (based on a non-standard lattice problem), and the leakage-resilient stream cipher in [39] (assuming PRGs to behave as random oracles that the leakage functions cannot access). In view of this state-of-the-art, another natural solution would be to use the simulation paradigm. Namely, to argue that some information reveals nothing substantial, it suffices to show that it can be efficiently simulated from some other information that is already part of the adversary's knowledge. This approach is empirically verifiable since it challenges the designer to build such a simulator, and the adversary to break the indistinguishability game. In the next sections, we argue that in the context of block ciphers, simulatable leakage is at least easier to guarantee than high pseudoentropy - and that efficient leakage-resilient PRGs can be proven secure under this assumption.

2 Simulatable Leakage

Concretely, we will study the physical security of a "natural" (i.e. conform to engineering intuition) PRG relying on the iterative application of a length-doubling 2PRG, represented in the left part of Figure 2 (the iterative application of length-$q$pling generator qPRG would allow improved efficiency at the cost of more physical information leakage, and relies on similar security proofs). Furthermore, we will focus on the block cipher based instantiation of 2PRG represented in the right part of the figure, where p_0 and p_1 are public constants (larger expansion factors q's are directly obtained by encrypting more p_i's). The (leakage-free) security of this PRG is easily seen by a hybrid argument. It enjoys many advantages such as simplicity, efficiency and forward security (see more discussions in [5]). From a physical security point of view, it also avoids the alternating structure and large randomness requirements of previously published proposals [13,30,38]. However, it turns out to be extremely difficult to prove the leakage-resilience of this construction in a standard setting (independent of its instantiation).

In order to obtain practically-relevant proofs of leakage-resilience, we want our assumption to be local (i.e. focusing on a single iteration), and re-usable. The second condition suggests to consider block cipher implementations for this purpose. On one hand, they are among the work horses of today's secure communications [23]. On the other hand, they are frequent targets of side-channel analysis, with a vast literature on attacks and countermeasures - making them natural candidates for mitigating the instantiation issues raised in [33]. In the rest of this section, we will consequently define the simulatable leakage assumption for block ciphers (denoted as $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ from now on), and argue about its empirical verifiability. The next section will then show how to use this assumption to prove the leakage-resilience of the PRG from Figure 2.

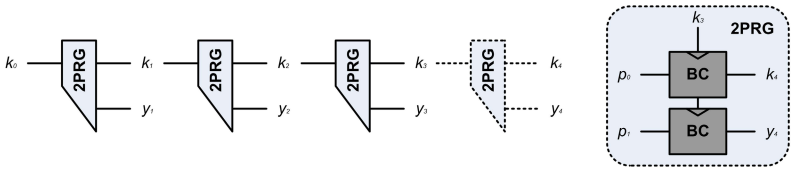

Fig. 2. Left: leakage-resilient PRG. Right: 2PRG instantiation with block ciphers.

2.1 Formal Definition

As discussed in Section 1.1, actual leakage traces are made of multiple samples, each of them being the output of a leakage function. Yet, since our goal is to define our assumptions in general terms, this section will take advantage of a slightly more concise notation that is independent of the actual representation of these traces. That is, we will denote the probabilistic leakage corresponding to a block cipher execution as: $y = \mathsf{BC}_k(x) \rightsquigarrow \mathbf{l} \stackrel{\mathsf{def}}{=} \mathsf{L}(k, x)$, with L the (global) leakage function (i.e. including all the samples).

In practice, we do not know how to express this function as a circuit or a program that a computer could evaluate: we can only sample it by taking leakage measurements from the target circuit on given inputs. Leakages resulting from a complex physical process, it is even unclear how efficiently a Turing machine could compute them. For this reason, they will be available through queries to a public oracle in our model, and our complexity measures will take the number of these queries into account: an (s, t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$ will do at most s queries to L and run in time at most t. Note that we define the leakage oracle as stateless, to capture the usual situation in side-channel attacks where leakages only depend on the current state of the target device and some independent randomness. Using this notation, the requirement we make on a block cipher implementation is that the leakages are *simulatable*. That is, we require that a (stateless) leakage simulator oracle $\mathcal{S}^{\mathsf{L}}(\cdot, \cdot, \cdot)$ can be built, possibly relying on accessing the implementation and measuring equipment producing the real leakages. It must be able to return a simulated leakage corresponding to any (possibly inconsistent) key, plaintext and ciphertext, and its responses must be such that no efficient adversary $\mathcal{A}$ can guess the bit b in the following q-sim game except with a small advantage.

In this game, the adversary can query the device for the encryption of q values of his choice. If $b = 0$, he receives the encryption of his queries and the corresponding real leakages. If $b = 1$, he receives the encryption of his queries and simulated leakages, based on the plaintext and ciphertext, but ignoring the (real) key k that was used to compute them, which is replaced by another random k^*

Game q-$\mathsf{sim}(\mathcal{A}, \mathsf{BC}, \mathsf{L}, \mathcal{S}, b)$.		
The challenger selects two random keys k and k^* in $\{0,1\}^n$. The output of the game is a bit b' computed by $\mathcal{A}^{\mathsf{L}}$ based on the challenger responses to a total of at most q adversarial queries of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Enc}(x)$	$\mathsf{BC}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{BC}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{BC}_k(x))$
and one query of the following type:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}(z, x)$	$\mathcal{S}^{\mathsf{L}}(z, x, k)$	$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$

in an invocation of $\mathcal{S}^{\mathsf{L}}(\cdot, \cdot, \cdot)$. Independently of these encryption queries, $\mathcal{A}$ gets one more chance of winning the game by being able to query $\mathcal{S}^{\mathsf{L}}(\cdot, \cdot, \cdot)$ on inputs of his choice, the ciphertext being the real or random key depending on b. This extra query captures the case where the key used in a block cipher was itself a ciphertext from a previous iteration. Note that it departs from the real world/ideal world paradigm, as $\mathcal{S}^{\mathsf{L}}$ is invoked for both values of b. This aspect plays a central role in our further developments. Additional types of (e.g. decryption) queries could be added to the game. However, the two proposed ones capture the usual situation where a block cipher is used to produce a key, which is then used to encrypt multiple plaintexts. It can be observed that we do not use the fact that BC is a block cipher so far. Its invertibility will however be used in the next subsection, when proposing our instance of leakage simulator.

Definition 1 (q-simulatable Leakage). *A block cipher BC with leakage function L has $(s_{\mathcal{S}}, t_{\mathcal{S}}, s_{\mathcal{A}}, t_{\mathcal{A}}, \varepsilon)$ q-simulatable leakages if there is an $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded simulator $\mathcal{S}^{\mathsf{L}}$ such that, for every $(s_{\mathcal{A}}, t_{\mathcal{A}})$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, we have:*

$$|\Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{BC}, \mathsf{L}, \mathcal{S}, 1) = 1] - \Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{BC}, \mathsf{L}, \mathcal{S}, 0) = 1]| \leq \epsilon.$$

Note that $\mathcal{A}^{\mathsf{L}(\cdot,\cdot)}$ can query the leakage function $s_{\mathcal{A}}$ times, independently of the q queries to the target implementation in the q-sim game. In practice, these $s_{\mathcal{A}}$ queries could correspond to profiling efforts to build a leakage model (e.g. as in step 2 of the attack in Section 1.3). They will also be useful to generate simulated leakages in our security proofs. As previously mentioned, we will keep small constant values for q in any practical instantiation of the q-simulatability game. This choice connects with the observation that 1-simulatability does not imply q-simulatability without severe security degradation. For example, it is easy to see that there might be block cipher implementations that offer perfect $q - 1$ simulatability but not q-simulatability. Consider a block cipher BC' built from a block cipher BC as follows: $\mathsf{BC}'_{k_1 \cdots k_q}(x) := \mathsf{BC}_{k_1 \oplus \cdots \oplus k_q}(x)$ (for a constant q), and a leakage function that leaks one of the k_i's every time the device computes. Clearly, $q - 1$ leakages will not provide any information about the cipher key, while the q-th leakage will fully disclose this key, making it trivial to detect the simulation in our game. In fact, this example also matches the usual intuition in side-channel attacks that security degrades almost exponentially with the number of queries, as will be illustrated experimentally in the next subsection.

2.2 Empirical Verifiability

To show that the previous simulatability assumption is realistic, we will first instantiate an efficient simulator $\mathcal{S}^{\mathsf{L}}_{\mathrm{s\&c}}(\cdot,\cdot,\cdot)$ to be used in the Enc and Gen queries of the q-sim game, based on a block cipher implementation. We will then discuss the interpretation of this assumption with respect to actual side-channel attacks.

As suggested by the acronym $\mathcal{S}^{\mathsf{L}}_{\mathrm{s\&c}}(\cdot,\cdot,\cdot)$, our proposal of simulator is based on the splitting and concatenation of leakage traces. For this purpose, and as we now consider concrete instantiation issues, we again need the specific notations of Section 1.1, and take the case of the AES for illustration. Namely, we will use $y = \mathsf{AES}_k(x) \rightsquigarrow \mathbf{l} = \mathbf{l}^{\mathsf{p}} \| \mathbf{l}^{\mathsf{c}}$, with $\mathbf{l}^{\mathsf{p}} = [l_0, l_1, \ldots, l_5]$ (resp. $\mathbf{l}^{\mathsf{c}} = [l_6, l_7, \ldots, l_{10}]$) denoting the first (resp. second) half of the traces, and $||$ the concatenation operator. Next, we want to build a simulator for such traces using only the knowledge of the public values x and y. In this context, a central observation already made in Section 1.2 is that any known intermediate value during a cryptographic computation can be exploited to check its consistency with the leakage. That is, taking the example of (noiseless) Hamming weight samples for illustration, it is quite easy to check whether the triple $(\mathbf{l}, x, y)$ is consistent by checking whether $l_0 = \mathsf{HW}(x)$ and $l_{10} = \mathsf{HW}(y)$. Yet, we still have that the "middle samples" $l_1, l_2, \ldots, l_8, l_9$ may not be as easy to exploit since the intermediate values $x_1, x_2, \ldots, x_8, x_9$ are not given to the adversary. As a result, the goal of the simulator will be to build traces that are at least consistent with the input/output pair (x, y). This is where the specialization to (invertible) block ciphers turns out to be useful, leading to the following proposal:

Leakage simulator instantiation $\mathcal{S}^{\mathsf{L}}_{\mathrm{s\&c}}(k^*, x, y)$.
1. Run $y' = \mathsf{AES}_{k^*}(x) \rightsquigarrow \mathbf{l}^{\mathsf{p}} \| \alpha$;
2. Compute $x' = \mathsf{AES}^{-1}_{k^*}(y)$;
3. Run $y = \mathsf{AES}_{k^*}(x') \rightsquigarrow \beta \| \mathbf{l}^{\mathsf{c}}$;
4. Output $\mathbf{l}^{\mathsf{p}} \| \mathbf{l}^{\mathsf{c}}$;

It is easy to verify that this simulator instance generates leakages that are consistent with the public values x and y, since in practice it does nothing else than generating traces from these values with a randomly generated key and concatenating them. Hence, it can be implemented using the same hardware as the target device containing the correct key. Note also that the same instance can directly be used in the Gen queries by adapting its inputs.

Interpretation. The assumption in this section suggests that there exists situations in which the leakage of a cryptographic implementation can be simulated without knowing all its secrets. For this purpose, our instance of simulator essentially relies on the possibility to use the same hardware as the one manipulating the actual cipher key. We believe this fact nicely captures the idea that the only secret in a cryptographic implementation should indeed be this key (not the device manipulating it). The assumption is also expressed as a game that can be tested by evaluation laboratories, since they could control both keys k and k^*. In practice, the main question naturally is whether the probability to win the q-sim

game can remain sufficiently low in front of actual side-channel distinguishers. There are two natural strategies that could be considered to answer it:

1. Performing standard DPA attacks exploiting the first and last encryption round leakages, e.g. trying to find an inconsistency between the x's and y's.
2. Targeting the middle rounds where concatenation occurs to find a direct inconsistency in the trace, possibly based on the key information gathered.

Starting with the first type of distinguishers, an important observation is that resisting them is at least easier than guaranteeing high HILL pseudoentropy for a block cipher key. This relates to the previously observed fact that attacks checking the consistency between the traces and the device state are not possible in the q-sim game, since the key is not given to the adversary. In other words, the device state is not known and can only be guessed, just as usually considered in side-channel analysis. Of course, being more realistic than the HILL pseudoentropy assumption does not imply empirical verifiability yet. Typically, there is little hope to ensure any security for small and unprotected devices (e.g. 8-bit microcontrollers), as key recovery is usually possible with very limited data complexities in these cases [34]. Under certain hypotheses, it is even possible to exploit the middle round leakages against such devices [31]. By contrast, a reasoning in the lines of Section 1.2 suggests that the simulatable leakage assumption could be realistic for (unprotected but parallel) hardware implementations.

For example, Figure 3 depicts the security evaluation of the best attack performed against such an implementation during the DPA Contest V2 (after two years of public investigations) [29,37]. It indicates that as long as the number of queries q remains limited (e.g. below 10), the success probability in recovering the key (hence, in finding inconsistencies between x's and y's) remains close to 2^{-128} in this case. Say that an adversary would try to exploit the q first- and last-round leakages corresponding to his Enc queries, together with the last-round leakage of his additional Gen query, and would be able to combine this information efficiently (which is unlikely in view of the large number of key candidates that remain possible after attacks with low data complexity). Then the amount of information leakage would at most be multiplied by three, still leaving comfortable security margins. Therefore, as long as our leakage-resilient PRG iterates qPRG's with small enough q's, we can conclude that this first strategy will not succeed against this hardware implementation[1]. Note that the linearity of the min/max bounds on Figure 3 typically illustrates the exponential security degradation (in time) that was mentioned in the previous subsection.

Although the second strategy is admittedly less investigated, we argue that it can also be verified for a wide variety of implementations based on the following

[1] Leakage-resilient constructions as proposed in this (and previous) works are naturally interesting in the context of small embedded devices as well, in combination with other hardware level countermeasures. In particular, they simplify the goal of protecting an implementation against arbitrary number of queries into the easier goal of protecting it against a bounded number of queries. We gave the example of the DPA Contest V2 for illustration, and because it is publicly available.

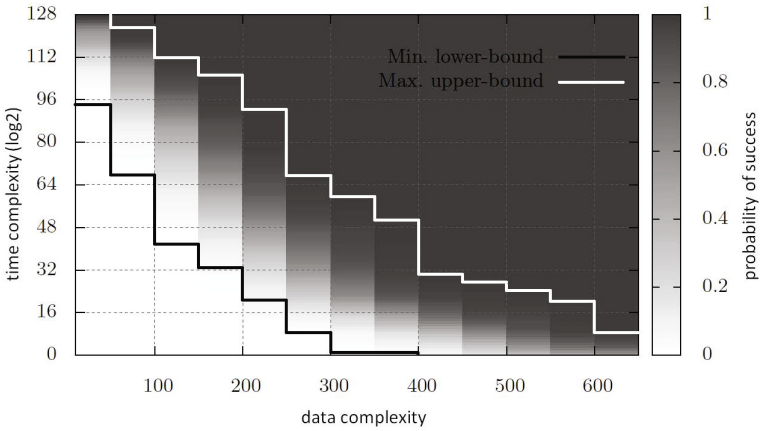

Fig. 3. Security evaluation for best attack of the DPA contest v2

reasoning. What is needed for this strategy to fail is an efficient method for concatenating side-channel traces in an indistinguishable manner. For this purpose, the key observation is that most current microelectronic devices are based on sequential logic circuits. As illustrated in Figure 4 for a couple of rounds of an AES implementation, such circuits essentially update some memory elements (i.e. the registers in dark gray on the figure) every clock cycle. And the length of these clock cycles is selected according to the longest delay needed to perform a round (aka the critical path), with some security margin. One consequence of this setup is that the circuit activity (hence, its leakage) is maximum at the beginning of each cycle (when the round computation actually takes place), and rapidly decreases afterwards. As indicated in the figure, the fact that each clock cycle should anyway be longer than the critical path guarantees that there exist samples with little or no activity. Interestingly, these points where no activity occurs usually contain no exploitable information. This observation actually

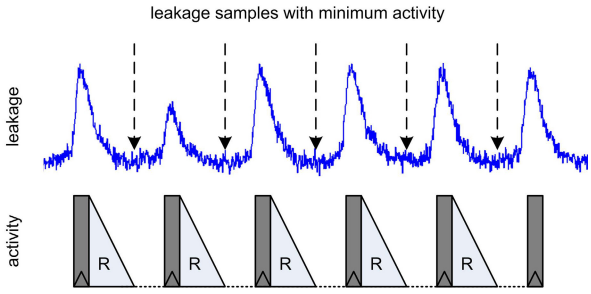

Fig. 4. Selection of samples for the concatenation of leakage traces

connects with the intuition from side-channel attacks that only a few samples in the measurements contain useful signal, i.e. the so-called "points of interest".

For example, the Signal-to-Noise Ratio (SNR) curves in [26] (Section 4.3.1) illustrate this fact. In general, concatenating traces exactly at their non-informative points can be done without risk of being distinguishable, since both the actual traces and the simulated ones would exhibit a noise following the same distribution at these points. Hence, our assumption for this second strategy to fail boils down to the existence of a couple of points *without* interest in the traces (which we believe is generally verified) and the ability to detect them. The latter task is relatively easy since (*i*) any side-channel distinguisher (e.g. the ones in Section 1.3) can be used for this purpose and (*ii*) the simulator can predict the full state corresponding to his fake inputs, hence allowing it to easily plot SNR curves. For illustration we performed such concatenations in the context of actual power traces and compared their spectrum with the one of original traces, without being able to detect any significant bias. As a result, we conclude that security against this second type of distinguishers can sometimes be ensured too.

Challenges. As for any cryptographic assumption, the claim that the simulatable leakage requirement is empirically verifiable will take strength with further investigations by physical cryptanalysts. In this respect, we believe that a central benefit of our security game is that it can be challenged using the techniques developed by the cryptographic hardware community. In order to stimulate research in this direction, we conclude this section by stating three challenges:

C1 (constructive). Design alternative instances of simulators. For example, the proposal in this section relies on the splitting and concatenation of leakage traces, based on the ability to detect "points without interest". But more sophisticated techniques for mixing the traces could be investigated.

C2 (constructive). Given any instance of simulator, design efficient block cipher implementations with q-simulatable leakages, for the largest possible q's. This challenge concurs with the one of securing these implementations against side-channel key recoveries with data complexity bounded to q.

C3 (destructive). Given a block cipher implementation and an instance of simulator, break the q-sim game with non-negligible advantage.

Regarding this last challenge, we finally note that falsifying the simulatable leakage assumption for one given instance of block cipher implementation and simulator does not imply that it cannot be verified at all. Our hope and belief is that it will be verified for a sufficiently wide range of realistic implementations.

3 Security Analysis and Proofs under Simulatable Leakages

We now want to show that the PRG of Figure 2 is secure when implemented with a secure block cipher that has 2-simulatable leakages (as previously mentioned, the proof would be similar for any constant value of q). The property we require from BC is to be a PRF. Our PRF adversary is a regular interactive Turing machine, augmented with an access to a leakage oracle $\mathsf{L}(\cdot, \cdot)$. While this oracle

is independent of the PRF challenger, nothing theoretically precludes that it might do some cryptanalytic work, and we therefore include the number of times it is queried in the adversary's total computational power.

Definition 2 (Pseudorandom Function). *A block cipher* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is a* (s,t,ε) pseudorandom function *(PRF) in the presence of leakage function* L *if, for every* (s,t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}(\cdot,\cdot)}$*, we have that:*

$$|\Pr[\mathcal{A}^{\mathsf{L}(\cdot,\cdot),\mathsf{BC}_k(\cdot)} = 1] - \Pr[\mathcal{A}^{\mathsf{L}(\cdot,\cdot)\mathsf{F}(\cdot)} = 1]| \leq \varepsilon,$$

where k *is a random key in* $\{0,1\}^n$ *and* F *is a random function.*

Note that if the leakage function was polynomial time, this definition would be strictly equivalent to the standard definition of a PRF. However, it remains an open problem to determine the exact complexity of such physical functions (which essentially corresponds to the cost of solving Maxwell's equations for a complex circuit). Therefore, and despite it is unlikely that actual leakage functions perform any cryptanalytic work, we believe it is conceptually cleaner to keep track of their queries separately, as specified in Definition 2.

The first step towards showing the security of our stream cipher consists in proving that one call of the 2PRG construction remains secure when it leaks and when the computation of its seed also leaks, as expressed in the next lemma. All bounds include the number of calls to the leakage function and the running time.

Lemma 1 (Single Iteration). *Let* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *with leakage function* L *be an* $(s,t,\varepsilon_{\mathsf{prf}})$ *PRF having* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \varepsilon_{\mathsf{sim}})$ *2-simulatable leakages, and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate* $(s_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded leakage simulator. Then, for every* k^-, p_0, p_1 *in* $\{0,1\}^n$ *and every* $(s - 3s_{\mathcal{S}}, t - \max(t_{\mathsf{prf}}, t_{\mathsf{sim}}))$*-bounded distinguisher* $\mathcal{D}^{\mathsf{L}}$*, the following inequation holds:*

$$|\Pr[\mathcal{D}^{\mathsf{L}}(y^+, k^+, \mathsf{L}(k,p_0), \mathsf{L}(k,p_1), \mathcal{S}^{\mathsf{L}}(k^-, p_1, k)) = 1] -$$
$$\Pr[\mathcal{D}^{\mathsf{L}}(y^{+*}, k^{+*}, \mathcal{S}^{\mathsf{L}}(k,p_0,y^{+*}), \mathcal{S}^{\mathsf{L}}(k,p_1,k^{+*}), \mathcal{S}^{\mathsf{L}}(k^-,p_1,k)) = 1]| \leq \varepsilon_{\mathsf{prf}} + \varepsilon_{\mathsf{sim}},$$

with $k, y^{+*}, k^{+*} \xleftarrow{R} \{0,1\}^n$*,* $y^+ = \mathsf{BC}(k,p_0)$*,* $k^+ = \mathsf{BC}(k,p_1)$*,* t_{prf} *being equal to* $3t_{\mathcal{S}}$ *augmented with the time needed to make 2 oracle queries to the PRF challenger and select a random key uniformly in* $\{0,1\}^n$*, and* t_{sim} *being the time to relay the content of two* Enc *and one* Gen *queries from and to a* q*-sim challenger.*

Proof. The proof makes use of an intermediary distribution that provides round outputs computed with one key and leakages simulated with another key. Details appear in the long version of the paper on the IACR ePrint archive.

Based on this Lemma, we show that the last output after l iterations of 2PRG remains pseudorandom even in the presence of the public outputs and leakages for all the previous iterations. For this purpose, we first specify our PRG instance:

Definition 3 (PRG Instance). *We denote as* $\mathsf{PRG}(k_0)$ *the pseudorandom generator in Figure 2 with n-bit initial state* k_0*. Each iteration of* PRG *expands the current state by running a length-doubling PRG (2PRG*: $\{0,1\}^n \rightarrow \{0,1\}^{2n}$*) following the recurrence* $(y_i, k_i) = \mathsf{2PRG}(k_{i-1})$*, and produces* y_1*,* y_2*, ... as output.*

Next, we define our notion of leakage-resilient stream cipher as follows:

Definition 4 (Leakage-Resilient Stream Cipher). *Let* PRG *be the stream cipher of Definition 3 and let* $\mathsf{L}(k_i) = \mathsf{L}(k_i, p_0)||\mathsf{L}(k_i, p_1)$ *be the leakages from its* i^{th} *iteration. The implementation of this* PRG *is* (l, s, t, ε)*-LR-pseudorandom if, for every* (s, t) *bounded distinguisher* $\mathcal{D}^{\mathsf{L}}$*, the following inequation holds:*

$$|\Pr[\mathcal{D}^{\mathsf{L}}(y_1, \cdots, y_l, \mathsf{L}(k_0), \cdots, \mathsf{L}(k_{l-1})) = 1] \; - \\ \Pr[\mathcal{D}^{\mathsf{L}}(y_1, \cdots, y_{l-1}, U_n, \mathsf{L}(k_0), \cdots, \mathsf{L}(k_{l-1})) = 1])| \leq \epsilon,$$

with k_0 *and* U_n *uniformly random values chosen in* $\{0,1\}^n$*.*

We can now state our main theorem, which shows the leakage-resilience of the stream cipher above and offers tight bounds: we only require 2-simulatable leakages, and the security degrades linearly with the number of rounds.

Theorem 1. *Let* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ *be a block cipher that is an* $(s, t, \varepsilon_{\mathsf{prf}})$ *PRF with a leakage function* L *and* $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \varepsilon_{\mathsf{sim}})$ *2-simulatable leakages. Then, the implementation of* PRG *instantiated with* BC *is* $(s', t', \varepsilon', l)$*-LR-pseudo-random, where* $s' = s - (2l-1)(s_{\mathcal{S}} + 1)$*,* $\varepsilon' = 2l(\varepsilon_{\mathsf{prf}} + \varepsilon_{\mathsf{sim}})$*, and* $t' = t - t_{12}$ *where* t_{12} *is* $2lt_{\mathcal{S}}$ *augmented with the time needed to sample* $2l$ *random n-bit strings and evaluate* BC $2l$ *times, plus the time needed to relay these block cipher inputs, outputs and leakages from and to oracles*[2]*.*

Proof. We rely on Lemma 1 and on a hybrid argument. The full proof appears in the long version of the paper on the IACR ePrint archive.

We may observe that this proof, like the one of Lemma 1, does not make full use of the power of the adversary in the q-sim game. They could indeed accommodate a non-interactive variant of game in which the plaintexts of the Enc and Gen queries are fixed, and the key of the Gen query is chosen randomly.

Conclusion

This paper suggests that the specification of realistic leakage assumptions may allow simplifying the proofs of natural constructions (such as the stream cipher in Figure 2), for which one intuitively expects an improved resistance against

[2] We do not include these relay times in the operation counts, because we assume them to be small compared to the time needed for the block cipher evaluations.

practical side-channel attacks. While the simulatable leakage requirement introduced in this work naturally raises open questions regarding the implementation scenarios in which it can be fulfilled (e.g. the challenges in Section 2.2), we can at least claim that it is more realistic than requirements such as the bounded range or high HILL pseudoentropy used in previous proofs for similar (symmetric cryptographic) constructions. Interesting scopes for further investigations include the application of simulatability to other primitives, and the quest for more generic yet empirically verifiable assumptions that could be exploited to analyze the leakage of cryptographic implementations.

Acknowledgements. We thank Ran Canetti and Martijn Stam for interesting discussions and useful suggestions. This work has been funded in parts by the European Commission through the ERC project 280141 (acronym CRASH) and the European ISEC action grant HOME/2010/ISEC/AG/INT-011 B-CCENTRE project. François-Xavier Standaert is an associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Yu Yu was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61172085, 61061130540, 61073174, 61103221, 11061130539, 61021004 and 61133014.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
4. Barak, B., Shaltiel, R., Wigderson, A.: Computational analogues of entropy. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 200–215. Springer, Heidelberg (2003)
5. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003)
6. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
7. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
8. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
9. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)

10. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) STOC, pp. 621–630. ACM (2009)
11. Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010)
12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE Computer Society (2008)
13. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 213–232. Springer, Heidelberg (2012)
14. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
15. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
16. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 45–60. USENIX Association (2008)
17. Hsiao, C.-Y., Lu, C.-J., Reyzin, L.: Conditional computational entropy, or toward separating pseudoentropy from compressibility. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 169–186. Springer, Heidelberg (2007)
18. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
19. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
20. Joye, M., Tunstall, M.: Fault Analysis in Cryptography. Springer (2012)
21. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
22. Kiltz, E., Pietrzak, K.: Leakage resilient elGamal encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010)
23. Knudsen, L.R., Robshaw, M.: The Block Cipher Companion. In: Information Security and Cryptography. Springer (2011)
24. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
25. Mangard, S., Oswald, E., François-Xavier: One for all – all for one: unifying standard differential power analysis attacks. IET Information Security 5(2), 100–110 (2011)
26. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
27. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
28. Moradi, A.: Statistical tools flavor side-channel collision attacks. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 428–445. Springer, Heidelberg (2012)

29. Telecom ParisTech, http://www.dpacontest.org/ (retrieved on August 1, 2012)
30. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
31. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic side-channel attacks on the AES: Why time also matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
32. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
33. Standaert, F.-X.: How leaky is an extractor? In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 294–304. Springer, Heidelberg (2010)
34. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)
35. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, Information Security and Cryptography, pp. 99–134. Springer, Heidelberg (2010)
36. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 390–406. Springer, Heidelberg (2013)
37. Veyrat-Charvillon, N., Gérard, B., Standaert, F.-X.: Security evaluations beyond computing power. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 126–141. Springer, Heidelberg (2013)
38. Yu, Y., Standaert, F.-X.: Practical leakage-resilient pseudorandom objects with minimum public randomness. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 223–238. Springer, Heidelberg (2013)
39. Yu, Y., Standaert, F.-X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: ACM Conference on Computer and Communications Security, pp. 141–151. ACM (2010)