Leakage-Resilient Symmetric Cryptography Under Empirically Verifiable Assumptions

François-Xavier Standaert[1], Olivier Pereira[1], 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 [26], fault [6] or cold boot attacks [18], 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 [28], error-detection codes against fault attacks [22], and their formal extensions as compilers (e.g. [20, 21]) - 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 with the seminal work of Micali and Reyzin [29], 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 formal results obtained in these models practically relevant (both 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 [14][1]. Following their introduction, these models have been applied for analyzing different modern cryptographic primitives, including PRGs and stream ciphers [15, 33, 45, 46], PRFs and PRPs [13, 15, 45], signature schemes (e.g. [7, 23]) and public-key encryption (e.g. [2, 24]).

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 [14, 15, 33, 45, 46]. 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) [40]. 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 (e.g. how to quantify bounded leakage from actual measurements?).

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

[1] Intermediate solutions exist, additionally assuming a secure refreshing of the device [8, 11].

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 relying on block ciphers.

Notations. We use upper cases X, Y for random variables, lower cases x, y for their samples, sans serif fonts X, Y for functions and calligraphic letters $\mathcal{X}, \mathcal{Y}$ for Turing machines.

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 key alternating 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: $\mathbf{l} = [l_0, l_1, l_2, \dots, l_9, l_{10}]$. 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 [29]. 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 (deterministic) polynomial function of the bits of x_i (denoted as $x_i[j]$) and some noise [37]:

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

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 [28]. 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 side-channel attacks where leakage traces can be written as:

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

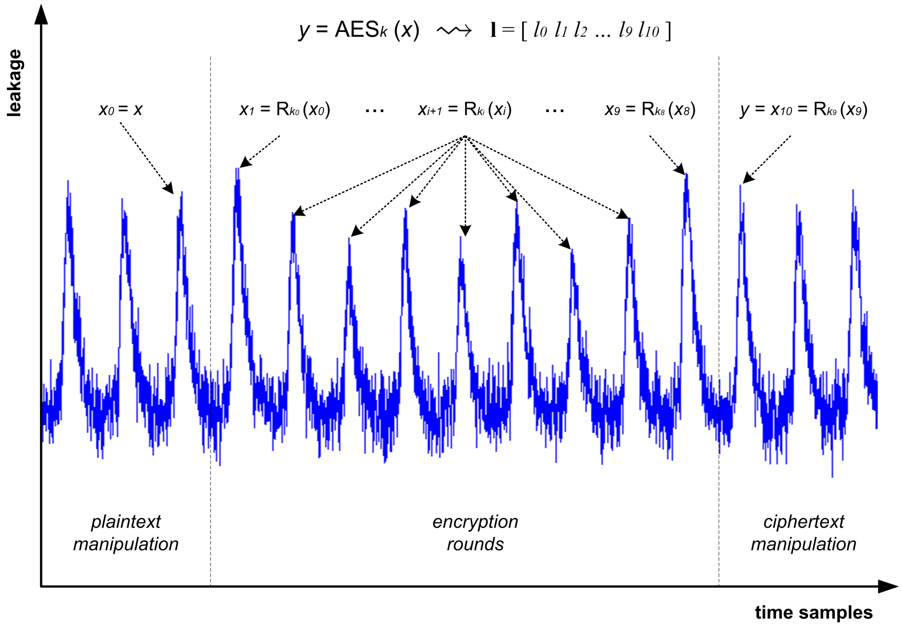

Fig. 1. Exemplary leakage trace of an AES encryption.

1.2 Bounded range & 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 (e.g. thousands of samples per encryption trace are usual), it turns out that the range of the leakage function is frequently larger than $\{0,1\}^n$. In practice, this large range is 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 [30].

In different works on leakage-resilient symmetric cryptography (e.g. [13–15, 33, 46]), 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 [24] 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} essentially 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, it remains that falsifying the pseudoentropy assumption simply requires that an adversary can check whether the (full) 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 [29]. We argue next that also in terms of practical assumptions, unpredictability is arguably easier to guarantee. For this purpose, we start by describing the operation of actual side-channel attacks.

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 [26], Gandolfi et al.'s electromagnetic analysis [16], Chari et al.'s template attacks [10], Brier et al.'s correlation power analysis [9], Schindler et al.'s stochastic approach [37], Gierlichs et al.'s mutual information analysis [17] and many variations. These attacks are "standard DPAs" in the sense defined by Mangard et al. [27], and operate according to three steps:

1. *Prediction.* The adversary predicts subkey-dependent intermediate values manipulated during the encryption process (e.g. the output of a first-round AES S-box).
2. *Modeling.* The adversary models the leakage corresponding to these intermediate values (e.g. assuming that it depends on the Hamming weight 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 should be observed 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 in order to obtain a master key candidate, e.g. using a key enumeration algorithm [42].

One important consequence of this attack description is that actual adversaries are usually not able to exploit all the leakage samples in a side-channel trace. In practice, only the intermediate computations that can be guessed will be useful. Taking our AES example again, it typically means that out of a side-channel trace $\mathbf{l} = [l_0, l_1, l_2, \ldots, l_9, l_{10}]$, 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:

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

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:

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

when targeting the first key byte, whith the r_i's uniformly random (unknown) bytes, and:

$$l_1^{\mathrm{adv}}[1] = \mathsf{HW}(x_1[1]) + \underbrace{\mathsf{HW}(r_0) + \ldots + \mathsf{HW}(r_{15})}_{\text{algorithmic noise}} + r_1^{\mathrm{ph}}, \tag{5}$$

when targeting the second key byte (and so on for the other bytes). In other words, only a single byte Hamming weight is considered as useful signal at a time in this setting[2].

1.4 One-way & 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 Equations 4, 5). 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 equivalently that the key/seed is computationally infeasible to predict given the leakage (see [4, 19] 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 [12] in the auxiliary-input setting (based on a non-standard lattice problem), and the leakage-resilient stream cipher in [46] (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

[2] Of course, it is possible to predict more than one byte of x_1. Yet, it remains that not all its bytes can be predicted at once by a computationally bounded adversary. The unpredicted bytes consequently behave as algorithmic noise (if the plaintexts are random).

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 [15, 33, 45]. However, it turns out to be extremely difficult to prove the leakage-resilience of this construction in a standard setting (independent of its instantiation). So far, it is only known to be physically secure under a non-standard random oracle assumption [46].

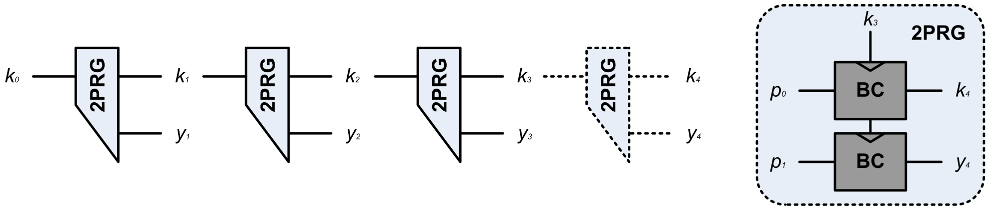

Fig. 2. Left: leakage-resilient PRG. Right: 2PRG instantiation with block ciphers.

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 naturally suggests to consider block cipher implementations for this purpose. On one hand, they are among the work horses of today's secure communications [25]. 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 [38]. 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 depicted in Figure 2.

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: $y = \mathsf{BC}_k(x) \rightsquigarrow \mathbf{l} \overset{\mathsf{def}}{=} \mathsf{L}(k, x)$, 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[3].

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. This leakage simulator 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.

<table>
<tr><td colspan="3">Game q-$\mathsf{sim}(\mathcal{A}, \mathsf{BC}, \mathsf{L}, \mathcal{S}, b)$.</td></tr>
<tr><td colspan="3"><i>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:</i></td></tr>
<tr><td>Query</td><td>Response if $b = 0$</td><td>Response if $b = 1$</td></tr>
<tr><td>$\mathsf{Enc}(x)$</td><td>$\mathsf{BC}_k(x)$, $\mathsf{L}(k, x)$</td><td>$\mathsf{BC}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{BC}_k(x))$</td></tr>
<tr><td colspan="3"><i>and one query of the following type:</i></td></tr>
<tr><td>Query</td><td>Response if $b = 0$</td><td>Response if $b = 1$</td></tr>
<tr><td>$\mathsf{Gen}(z, x)$</td><td>$\mathcal{S}^{\mathsf{L}}(z, x, k)$</td><td>$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$</td></tr>
</table>

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 key 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

[3] This assumes that the leakage function does not retain information about its past states. As discussed in [36], such "memory effects" may be observed at the gate level in submicron technologies, but are unlikely to be exploitable at the block cipher level, especially if designers ensure a sufficient (time or space) separation between different block cipher executions.

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 (e.g. in leakage-resilient cryptography) 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:*

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

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. As a result, and as most previous symmetric primitives in leakage-resilient cryptography, our PRG construction relies on frequent re-keying.

2.2 Empirical verifiability

In order to show that the previous simulatability assumption is actually realistic, we will first instantiate an efficient simulator $\mathcal{S}^{\mathsf{L}}_{\text{s\&c}}(\cdot,\cdot,\cdot)$ 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 $\mathcal{S}^{\mathsf{L}}_{\text{s\&c}}(\cdot,\cdot,\cdot)$, 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 be used in the Gen queries by adapting its inputs (i.e. running it on (z, x, k) if $b = 0$, or (z, x, k^*) if $b = 1$, rather than on (k^*, x, y)).

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 taking advantage of the first and last encryption round leakages, e.g. trying to find an inconsistency between the x's and the y's.
2. Targeting the middle rounds where the concatenation occurs, in order 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 (e.g. 8-bit) and unprotected devices, as key recovery is usually possible with very limited data complexities in these cases [39]. Under certain hypotheses, it is even possible to exploit the middle round leakages against such devices [35]. 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) [31, 43]. 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}. 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. So 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[4]. Note that the linearity of the min/max bounds on Figure 3 illustrates the exponential security degradation (in time) that was mentioned in the previous subsection.

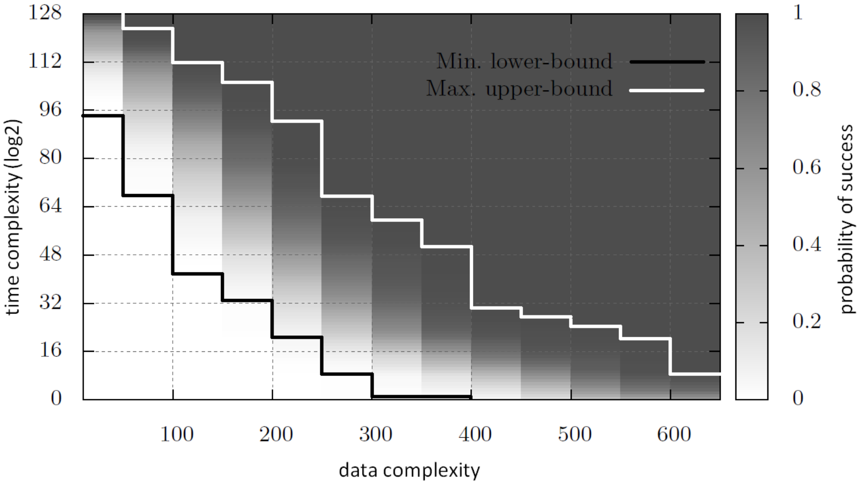

Fig. 3. Security evaluation for best attack of the DPA contest v2.

We additionally remark that the q-sim game considers adversaries who can choose their challenges in the Enc and Gen queries. This leads to the question whether improved attacks could take advantage of chosen-plaintexts or weak keys. However, it has been shown that there are no weak keys in standard DPA attacks [27], and that the adaptive selection of plaintexts only leads to marginal improvements in this context [44]. Hence, we do not expect these concerns to have a strong impact on the security of our proposal.

[4] 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 countermeasures. 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.

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 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.

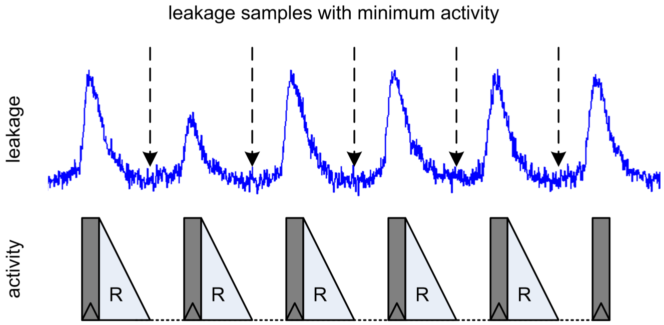

Fig. 4. Selection of samples for the concatenation of leakage traces.

Interestingly, these points where no activity occurs usually contain no exploitable information. This observation actually 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 [28] (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 listed 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 (FFT) 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[5].

[5] Even in situations where security against this second type of distinguishers would not always be fulfilled, our proposal would keep on ensuring security against all side-channel attacks excluding the middle-round leakages, already making a quite positive result in practice.

Challenges. Eventually, and just as for any cryptographic assumption, the claim that the simulatable leakage requirement is empirically verifiable will only 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 easily challenged using the techniques developed by the cryptographic hardware community. In order to stimulate further research in this direction, we conclude this section by explicitly 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 in an indistinguishible manner 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. As previously discussed, this challenge largely concurs with the one of securing these implementations against side-channel key recoveries with data complexity bounded to q.

C3 (destructive). Given the combination of a block cipher implementation and an instance of simulator, try to 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 this assumption will remain hard to falsify for a sufficiently wide range of realistic implementations.

3 Security analysis and proofs under simulatable leakages

Our goal is 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). Hence, 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 in Definition 2.

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

Lemma 1 (single iteration). *Let* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *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 needed to relay the content of two* Enc *and one* Gen *queries from and to a q-sim challenger.*

Proof. Let P_0 and P_2 denote the first and second probabilities from the inequality above, and P_1 denote the following intermediate probability:

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

with $k^* \xleftarrow{R} \{0,1\}^n$. From any $(s, t - t_{\mathsf{sim}})$ $\mathcal{D}^{\mathsf{L}}$ such that $|\mathrm{P}_0 - \mathrm{P}_1| = \varepsilon_{\mathsf{sim}}$, we build an (s, t) adversary $\mathcal{D}^{\mathsf{L}}_{\mathsf{sim}}$ against the 2-simulatability of the leakages of BC with the same distinguishing probability. $\mathcal{D}^{\mathsf{L}}_{\mathsf{sim}}$ simply issues $\mathsf{Enc}(p_0)$, $\mathsf{Enc}(p_1)$ and $\mathsf{Gen}(k^-, p_1)$ queries in the q-sim game instantiated with an appropriate $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator, and relays the responses to $\mathcal{D}^{\mathsf{L}}$. Depending on the value of b in the q-sim game, the inputs of $\mathcal{D}^{\mathsf{L}}$ are distributed as in P_0 or P_1. Furthermore, from any $(s - 3s_{\mathcal{S}}, t - t_{\mathsf{prf}})$ distinguisher $\mathcal{D}^{\mathsf{L}}$ such that $|\mathrm{P}_1 - \mathrm{P}_2| = \varepsilon_{\mathsf{prf}}$, we build an $(s, t, \varepsilon_{\mathsf{prf}})$ distinguisher $\mathcal{D}^{\mathsf{L}}_{\mathsf{prf}}$ against the pseudorandomness of BC. $\mathcal{D}^{\mathsf{L}}_{\mathsf{prf}}$ simply queries the pseudorandomness challenger on p_0 and p_1, obtaining $\tilde{y}$ and $\tilde{k}$, then selects a random value k', and runs the $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator in order to get $\mathcal{S}^{\mathsf{L}}(k', p_0, \tilde{y})$, $\mathcal{S}^{\mathsf{L}}(k', p_1, \tilde{k})$, $\mathcal{S}^{\mathsf{L}}(k^-, p_1, k')$. He finally relays the results to $\mathcal{D}^{\mathsf{L}}$. Depending on whether $\tilde{y}$ and $\tilde{k}$ are pseudorandom or random, the inputs of $\mathcal{D}^{\mathsf{L}}$ are distributed as in P_1 or P_2. This shows that, for every $(s - 3s_{\mathcal{S}}, t - \max(t_{\mathsf{prf}}, t_{\mathsf{sim}}))$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, we have $|\mathrm{P}_0 - \mathrm{P}_2| \leq |\mathrm{P}_0 - \mathrm{P}_1| + |\mathrm{P}_1 - \mathrm{P}_2| \leq \varepsilon_{\mathsf{prf}} + \varepsilon_{\mathsf{sim}}$. □

Based on this Lemma, we show that the last output after l iterations of $\mathsf{2PRG}$ 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 as follows:

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

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. We say that 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:*

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

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, as in the absence of leakages.

Theorem 1. *Let* $\mathsf{BC} : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}^n$ *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*[6].

Proof. Let $\mathsf{view}_{\mathsf{2PRG}}(k) := \langle y^+, \mathsf{L}(k, p_0), \mathsf{L}(k, p_1)\rangle$ denote the distribution of a real view for one PRG round, with $y^+ = \mathsf{BC}_k(p_0)$ and $k^+ = \mathsf{BC}_k(p_1)$. And given an appropriate $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator $\mathcal{S}^{\mathsf{L}}$, let $\mathsf{view}^*_{\mathsf{2PRG}}(k) := \langle y^{+*}, \mathcal{S}^{\mathsf{L}}(k, p_0, y^{+*}), \mathcal{S}^{\mathsf{L}}(k, p_1, k^{+*})\rangle$ with y^{+*} and k^{+*} selected at random. We extend these views to multiple rounds in the natural way: we write $\mathsf{view}_{\mathsf{2PRG}}(k, l)$ for the view of l real PRG rounds in which the output k^+ is used as seed for the next round, and similarly, we write $\mathsf{view}^*_{\mathsf{2PRG}}(k, l)$ for the view of l simulated rounds computed with the same chaining rule (i.e. using k^{+*} as next round key).

We define a set of $l + 1$ hybrid distributions, starting from the view of an l-round leaking execution of PRG and leading to a view in which all round keys and outputs are selected uniformly at random and leakages are simulated. More precisely, we define $H_0, \ldots, H_l$, where: $H_i := \mathsf{view}^*_{\mathsf{2PRG}}(k_0, i)||\mathsf{view}_{\mathsf{2PRG}}(k_i, l - i)$, with k_0 selected uniformly at random, and k_i taken as the key part of the output of the i-th round in $\mathsf{view}^*_{\mathsf{2PRG}}(k_0, i)$, or a uniform value for $i = 0$. From these notations, we see that H_0 is the full view for l rounds of PRG, while H_l is a completely random view, as expected.

Suppose now that there exists an $(s - s_1, t - t_1)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ such that $|\Pr[\mathcal{D}^{\mathsf{L}}(H_i) = 1] - \Pr[\mathcal{D}^{\mathsf{L}}(H_{i+1}) = 1]| = \delta$. We then build an (s, t)-bounded distinguisher $\mathcal{D}^{\mathsf{L}}_1$ between the two distributions in Lemma 1 that succeeds with the same probability δ (t_1 will be defined below). $\mathcal{D}^{\mathsf{L}}_1$ proceeds as follows. It first samples $\mathsf{view}^*_{\mathsf{2PRG}}(k_0, i)$ from a random key k_0 and gets its inputs $d_1, \ldots, d_5$ from one of the two distributions in Lemma 1 using k_{i-1} for k^-. It then replaces $\mathcal{S}^{\mathsf{L}}(p_1, k_{i-1}, k_i)$ with d_5 and builds the view for the $(i+1)$-th round as $\langle d_1, d_3, d_4\rangle$. The view for the $l - i - 1$ remaining rounds is sampled as $\mathsf{view}_{\mathsf{2PRG}}(d_2, l - i - 1)$. The l-round view produced is then relayed to $\mathcal{D}^{\mathsf{L}}$, and its output is given as output of $\mathcal{D}^{\mathsf{L}}_1$. Depending on which distribution the values $d_1, \cdots, d_5$ were sampled from, $\mathcal{D}^{\mathsf{L}}$ received inputs sampled as in H_i or H_{i+1}. We see that the running

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

time of $\mathcal{D}_1^{\mathsf{L}}$ is the one of $\mathcal{D}^{\mathsf{L}}$ to which we add the time t_1 to select $2i$ random n-bit strings, perform $2(l-i-1)$ evaluations of BC, run $\mathcal{S}^{\mathsf{L}}$ $2i-1$ times, read $d_1, \cdots, d_5$ and submit its input to $\mathcal{D}^{\mathsf{L}}$. We also see that $\mathcal{D}_1^{\mathsf{L}}$ performs $s_1 := (2i-1)s_{\mathcal{S}} + 2(l-i-1)$ calls to L.

H_l is made of l rounds with random outputs. However, we need to prove the pseudorandomness of the last round output in front of an adversary who first saw $l-1$ real rounds of $\mathsf{2PRG}$. We reach that view with a second sequence of hybrid distributions $H_l, \ldots, H_{2l-1}$, where: $H_{l+i} := \mathsf{view}^*_{\mathsf{2PRG}}(k_0, l-i-1) \| \mathsf{view}_{\mathsf{2PRG}}(k_{l-i-1}, i) \| \mathsf{view}^*_{\mathsf{2PRG}}(k_{l-1}, 1)$, for a uniformly random k_0. The definition of H_l here is consistent with the previous one, while H_{2l-1} is a (fully) real view except for the last round that remains random.

Suppose now that there exists an $(s-s_2, t-t_2)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ such that $|\Pr[\mathcal{D}^{\mathsf{L}}(H_{l+i}) = 1] - \Pr[\mathcal{D}^{\mathsf{L}}(H_{l+i+1}) = 1]| = \delta$. We then build an (s,t)-bounded distinguisher $\mathcal{D}_2^{\mathsf{L}}$ between the two distributions in Lemma 1 that succeeds with the same probability δ (t_2 will be defined below). $\mathcal{D}_2^{\mathsf{L}}$ proceeds as follows. It first samples $\mathsf{view}^*_{\mathsf{2PRG}}(k_0, l-i-2)$ from a random key k_0, then gets its inputs $d_1, \cdots, d_5$ from one of the two distributions in Lemma 1 using k_{l-i-2} as k^- and replaces $\mathcal{S}^{\mathsf{L}}(p_1, k_{l-i-2}, k_{l-i-1})$ with d_5. The view for the $(l-i-1)$-th round is then built as $\langle d_1, d_3, d_4 \rangle$, and the view for the last $i+1$ remaining rounds is sampled as $\mathsf{view}_{\mathsf{2PRG}}(d_2, i) \| \mathsf{view}^*_{\mathsf{2PRG}}(k_{l-1})$. Again, the l-round view produced is relayed to $\mathcal{D}^{\mathsf{L}}$, and its output is given as output of $\mathcal{D}_2^{\mathsf{L}}$. Depending on which distribution the values $d_1, \cdots, d_5$ were sampled from, $\mathcal{D}^{\mathsf{L}}$ received inputs sampled as in H_{l+i} or H_{l+i+1}. We see that the running time of $\mathcal{D}_2^{\mathsf{L}}$ is the one of $\mathcal{D}^{\mathsf{L}}$ to which we add the time t_2 to select $2(l-i-1)$ random n-bit strings, perform $2i$ evaluations of BC, run the leakage simulator $2(l-i-1)$ times, read $d_1, \cdots, d_5$ and submit its input to $\mathcal{D}^{\mathsf{L}}$. We also see that $\mathcal{D}_2^{\mathsf{L}}$ performs $s_2 := 2(l-i-1)s_{\mathcal{S}} + 2i - 1$ calls to L.

From the triangle equality, it comes that any (s,t,ε) distinguisher between H_0 and H_{2l-1} must be an $(s,t,\varepsilon/2l)$ distinguisher between one of the (H_i, H_{i+1}) pairs. We eventually observe that the distributions H_0 and H_{2l-1} both sample $\mathsf{view}_{\mathsf{2PRG}}(k_0, l-1)$ from a random k_0, but differ through $\langle y_l, k_l, \mathcal{S}^{\mathsf{L}}(k_{l-1}, p_0, y_l), \mathcal{S}^{\mathsf{L}}(k_{l-1}, p_1, k_l) \rangle$. Indeed, H_0 computes y_l and k_l as outputs of the block cipher and has real leakages, while H_{2l-1} selects them at random and has simulated leakages. This implies that, from an (s,t)-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ such that $|\Pr[\mathcal{D}^{\mathsf{L}}(H_0) = 1] - \Pr[\Pr[\mathcal{D}^{\mathsf{L}}(H_{2l-1}) = 1]| = \delta$, we can also build an (s,t)-bounded distinguisher $\mathcal{D}_3^{\mathsf{L}}$ between $\mathsf{view}_{\mathsf{2PRG}}(k_0, l-1) \| y_l$ and $\mathsf{view}_{\mathsf{2PRG}}(k_0, l-1) \| U_n$ succeeding with the same probability: it just needs to run $\mathcal{D}^{\mathsf{L}}$ while ignoring k_l and the last two leakages. The proof follows from the observation that t_1 and t_2 (resp. s_1 and s_2) are always smaller than the t_{12} (resp. $(2l-1)(s_{\mathcal{S}}+1))$ bound from the theorem statement, and the fact that the two distributions from Lemma 1 cannot be separated with probability better than $\varepsilon_{\mathsf{prf}} + \varepsilon_{qsim}$ by an (s,t) bounded distinguisher. □

Note that this proof and the one of Lemma 1 do not make full use of the adversary's power in the q-sim game. They could accommodate a non-interactive game variant with fixed plaintexts in the Enc and Gen queries, and a randomly chosen key in the Gen query.

Conclusion

The analysis in 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 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. as suggested by 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 other 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.).

References

1. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
2. Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134. Springer, 2010.
3. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, 2009.
4. Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM 2003)*, pages 200–215, 2003.
5. Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.
6. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.
7. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Paterson [32], pages 89–108.
8. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In Trevisan [41], pages 501–510.
9. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
10. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
11. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In Trevisan [41], pages 511–520.
12. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *STOC*, pages 621–630. ACM, 2009.

13. Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
14. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.
15. Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Prouff and Schaumont [34], pages 213–232.
16. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
17. Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
18. J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.
19. Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2007.
20. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.
21. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
22. Marc Joye and Michael Tunstall. *Fault Analysis in Cryptography*. Springer, 2012.
23. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.
24. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.
25. Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information security and cryptography. Springer, 2011.
26. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
27. Stefan Mangard, Elisabeth Oswald, and François-Xavier. One for all – all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.
28. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
29. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
30. Amir Moradi. Statistical tools flavor side-channel collision attacks. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012.
31. Telecom ParisTech. http://www.dpacontest.org/, retrieved on aug. 1, 2012.
32. Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.

33. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
34. Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012.
35. Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: Why time also matters in DPA. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.
36. Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Paterson [32], pages 109–128.
37. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
38. François-Xavier Standaert. How leaky is an extractor? In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 294–304. Springer, 2010.
39. François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2008.
40. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer Berlin Heidelberg, 2010.
41. Luca Trevisan, editor. *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010.
42. Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.
43. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2013.
44. Nicolas Veyrat-Charvillon and François-Xavier Standaert. Adaptive chosen-message side-channel attacks. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 186–199, 2010.
45. Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.
46. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 141–151. ACM, 2010.