Leakage Resilient Cryptography in Practice

François-Xavier Standaert[1⋆], Olivier Pereira[1⋆], Yu Yu[1],
Jean-Jacques Quisquater[1], Moti Yung[2,3], Elisabeth Oswald[4]

[1] Université catholique de Louvain, Crypto Group, Belgium.
[2] Columbia University, Dept. of Computer Science, USA.
[3] Google Inc, USA.
[4] University of Bristol, Department of Computer Science, UK.

e-mails: `fstandae,olivier.pereira,yu.yu,jjq@uclouvain.be`
`moti@cs.columbia.edu; elisabeth.oswald@bristol.ac.uk`

Abstract. In this report, we are concerned with models to analyze the security of cryptographic algorithms against side-channel attacks. Our objectives are threefold. In a first part of the paper, we aim to survey a number of well known intuitions related to physical security and to connect them with more formal results in this area. For this purpose, we study the definition of leakage function introduced by Micali and Reyzin in 2004 and its relation to practical power consumption traces. Then, we discuss the non equivalence between the unpredictability and indistinguishability of pseudorandom generators in physically observable cryptography. Eventually, we examine the assumption of bounded leakage per iteration that has been used recently to prove the security of different constructions against side-channel attacks. We show that approximated leakage bounds can be obtained using the framework for the analysis of side-channel key recovery attacks published at Eurocrypt 2009.

In a second part of the paper, we aim to investigate two recent leakage resilient pseudorandom generators, both from a theoretical and practical point of view. On the one hand, we consider a forward secure generator from ASIACCS 2008 and its similarities with a previous construction by Bellare and Yee. On the other hand, we analyze Pietrzak's block cipher based construction from Eurocrypt 2009. Doing this, we put forward the difficulty of meaningfully restricting the physical leakages and show that this difficulty leads to different drawbacks. It allows us to emphasize the differences between these two designs. First, one construction that we analyze requires strong black box assumptions (*i.e.* random oracles) - the other one considers unrealistic leakages leading to (possibly useless) performance overheads. Second, one construction considers an adversary able to adaptively choose a leakage function while the second one does not permit this adaptivity. Third, the security proof of the Eurocrypt 2009 construction relies on the assumption that "only computation leaks" (or relaxed but related hypotheses) while this assumption is not necessary for the ASIACCS construction. We then discuss the impact of these hypotheses with respect to recent technological advances.

⋆ Research Associate of the Belgian Fund for Scientific Research (FNRS - F.R.S.).

In the third part of the paper, we show that Pietrzak's leakage resilient mode of operation from Eurocrypt 2009 can be broken with a standard DPA if it is re-initialized without sharing new keys. Then, we propose solutions to fix this issue and extend the initial proposal from ASIACCS 2008 in order to rely on more standard cryptographic constructions. We use these alternative designs to illustrate the incompatibility between a fully adaptive selection of the leakage function and the secure initialization of a pseudorandom generator. We also argue that simple pseudorandom functions (*e.g.* the one of Goldreich, Goldwasser, Micali) can be shown leakage resilient, using the random oracle methodology. We additionally discuss the security *vs.* performance tradeoff that is inherent to these different schemes. Eventually, we show that the security of the forward secure pseudorandom number generator of Bellare and Yee against side-channel attacks cannot be directly generalized in the standard model. It is an open problem to determine the minimum black box assumptions and restrictions of the leakage function for this purpose.

1 Introduction

Theoretical treatments of physical attacks have recently attracted the attention of the cryptographic community, as witnessed by various publications, *e.g.* [1, 17, 22, 24, 29, 31, 33, 34, 42]. These works consider adversaries enhanced with abilities such as inserting faults during a computation or monitoring side-channel leakages. They generally aim to move from the ad hoc analysis of the attacks towards stronger and more systematic security arguments or proofs. Quite naturally, these more general approaches also have limitations that are mainly caused by the versatility of physical phenomenons. Namely, since it is impossible to prove the security against an all powerful physical adversary, one has to find ways to meaningfully restrict them. This is in fact similar to the situation in classical cryptography, where we need to rely on computational assumptions. That is, when moving to a physical setting, we need to determine what are the physical limits of the adversary. Therefore, the question arises of how relevant these physical models are and to which extent they capture the engineering experience. In order to tackle this question, it is useful to first introduce some usual terminology, *e.g.* from the side-channel cryptanalysis lounge [14]:

1. Invasive *vs.* non-invasive attacks. Invasive attacks require depackaging the chip to get direct access to its inside components. A typical example of this is the connection of a wire on a data bus to see the data transfers. A non-invasive attack only exploits externally available information (the emission of which is however often unintentional) such as running time, power consumption, ...

One can go further along this axis by distinguishing local and distant attacks: a local attack requires close but external (*i.e.* non-invasive) proximity to the device under concern, for example by a direct connection to its power supply. As opposed, a distant attack can operate at a larger distance, for example by measuring an electromagnetic field several meters (or hundreds of meters) away.

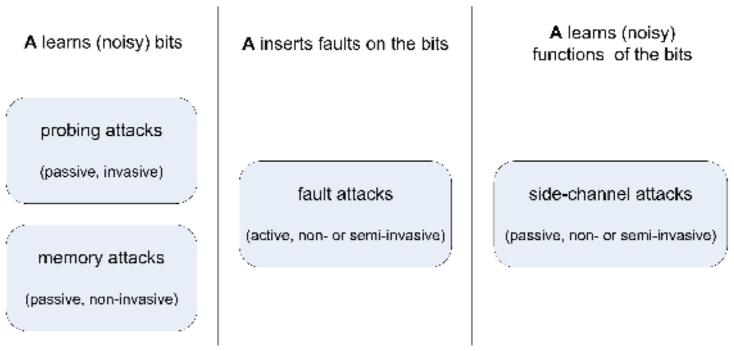

A learns (noisy) bits
A inserts faults on the bits
A learns (noisy) functions of the bits
probing attacks
(passive, invasive)
memory attacks
(passive, non-invasive)
fault attacks
(active, non- or semi-invasive)
side-channel attacks
(passive, non- or semi-invasive)

Fig. 1: Informal classification of physical attacks.

2. Active *vs.* passive attacks. Active attacks try to tamper with the devices proper functioning. For example, fault induction attacks will try to introduce errors in the computation. By contrast, passive attacks will simply observe the devices behavior during its processing, without disturbing it.

As an illustration, Figure 1 classifies different physical attacks according to these two axes. With this respect, it is important to remark that seemingly similar abilities can have very different costs in practice. For example, probing attacks such as described by Anderson and Kuhn [3] and the recent memory attacks based on data remanence issues [20] both allow the adversary to learn the value of certain bits in a cryptographic device. But the first one can only target depackaged chips and may require very expensive tools to probe circuits, *e.g.* when realized in advanced (65 nanometer or smaller) technologies. By contrast, memory remanence based attacks can take advantage of cheap "cold boot" techniques to read memory. In addition, the cost of an attack is not the only factor to consider when discussing its applicability. The likelihood to find its scenario in real life conditions is obviously as important. As a result, side-channel attacks are usually considered as the most dangerous type of physical attacks. They are at the same time low cost and realistic, *e.g.* when applied against small embedded devices, as in the Keeloq case study presented at Crypto 2008 [13].

In this report, we consequently investigate the relation between theoretical models and practical engineering works in the area of side-channel attacks. In particular, we consider the pseudorandom generators (PRG) proposed at ASIACCS 2008 [33] and Eurocrypt 2009 [34], respectively. These constructions are based on the same core ideas. First, they assume a bounded leakage for each iteration of the PRG. Second, they rely on frequent key updates in order to avoid the application of standard DPA attacks. In fact, these ideas were not new. Directly after the publication of the first power analysis attack [26], Paul Kocher listed possible countermeasure in which key updates combined with bounded leakage were explicitly mentioned [27, 28]. Hence, the novelty in the previous PRGs is not really in the design principles but rather in the advanced techniques for their analysis, leading to a better confidence in their security levels.

Both papers have pros and cons. Summarizing, the ASIACCS PRG was the first one to provide a systematic analysis of a block cipher based construction in a physically observable setting. It initiated a study of forward secure cryptographic primitives with their relation to side-channel issues. The underlying model in this work [42] is a specialization of Micali and Reyzin [31] and is motivated by the need to evaluate side-channel attacks on a fair basis. As will be shown in Section 5.4 of this report, this connection to the practice of side-channel attacks is required anyway, anytime a leakage bound needs to be assumed (and hence, quantified). In other words, the framework presented at Eurocrypt 2009 [42] is also useful to the work of Pietrzak [34] and more generally to any construction based on a λ-bit leakage: it can be used to estimate practical values for λ. On the negative side, the analysis in [33] considers black box security and physical security separately. It also relies on the existence of ideal ciphers.

The main advantage of [34] is to analyze the security of a PRG in a combined fashion, mixing black box and physical security issues, and in the standard model. It also introduces useful tools for the systematic investigation of physically observable devices (*e.g.* the quantification of the leakages with the HILL pseudoentropy). On the negative side, the PRG of Eurocrypt 2009 lacks a secure initialization process (as discussed in Section 6.1). It is also designed in order to face unrealistic (*i.e.* too powerful) leakages, *e.g.* the so called "future computation attacks" that we describe in Section 5.1. As a result, it exploits a (less efficient) "alternating structure" for which it is not clear if it is really required to prevent actual side-channel attacks or if it is only caused by proof technicalities. Eventually, its security proofs rely on the assumption that "only computation leaks" (or relaxed but related assumptions) which may not always be respected.

Following these observations, the goal of this report is threefold.

First, we aim to connect Micali and Reyzin's definition of leakage function to the practice of side-channel attacks. Doing so, we review the intuitions behind some important results, *e.g.* the non equivalence between the unpredictability and indistinguishability of PRGs implemented in leaking devices. Second, we aim to investigate the underlying assumptions and the concrete security provided by two PRGs, with and without alternating structure, in a systematic manner. Doing so, we provide a proof of security for an efficient construction similar to the one of Bellare and Yee [5], using the random oracle methodology. We also introduce definitions allowing us to formalize the practical security of an implementation, inspired by the q-limited adversaries used in Vaudenay's decorrelation theory [48]. Third, we analyze the initialization of a leakage resilient PRG with a public seed. Doing so, we put forward the incompatibility of a secure initialization with a fully adaptive selection of the leakage functions. We also emphasize that standard constructions of pseudorandom functions (PRF) [18] can be shown leakage resilient, in the random oracle model. We conclude this paper with a negative result. We show with a simple example that the leakage resilience of Bellare and Yee's PRG cannot be directly (meaning, without additional computational and physical hypotheses) proven in the standard model. We leave as an open problem to determine the minimum black box assumptions and restrictions of the leakage function that would be required for this purpose.

Note that this work mainly focuses on the formal techniques used to analyze two PRGs, namely [33] and [34]. Obviously, they are not the only attempts to study side-channel attacks from a more theoretical point of view. Several other references could be acknowledged, *e.g.* [8, 41], typically. However, we believe these two PRGs are emblematic of one approach for proving the security against side-channel attacks that we denote as the "global approach" in Section 4.

2 Background

2.1 Notations

We take advantage of notations from [39, 42]. In particular, let $x \overset{R}{\longleftarrow} \mathcal{X}$ be a uniformly distributed plaintext picked up from a set $\mathcal{X}$ and $k \overset{R}{\longleftarrow} \mathcal{K}$ be a uniformly distributed key picked up from a set $\mathcal{K}$. For simplicity, we take $\mathcal{X} = \mathcal{K} = \{0, 1\}^n$. Let also $\mathsf{E}_k(x)$ be the encryption of a plaintext x under a key k with a n-bit block cipher. In classical cryptanalysis, an adversary is able to query the block cipher in order to obtain pairs of the form $[x_i, \mathsf{E}_k(x_i)]$. In side-channel attacks, he is additionally provided with the output of a leakage function of which exemplary outputs are illustrated in Figure 2. Let $\mathbf{x}_q = [x_1, x_2, \ldots, x_q]$ be a vector containing a sequence of q input plaintexts to a target implementation. In our notations, the measurements resulting from the observation of the encryption of these q plaintexts are stored in a leakage vector that we denote as $\mathbf{l}_q = [l_1, l_2, \ldots, l_q]$. Each element l_i of the leakage vector is referred to as a leakage trace and is in the set of leakages $\mathcal{L}$. Typically, $\mathcal{L} = \mathbb{R}^{N_l}$, where N_l is the number of samples per trace (possibly coming from multiple channels [44]). For example, Figure 2 represents 4 leakage traces, corresponding to 4 input plaintexts encrypted under the same key. Eventually, we denote the t^{th} leakage sample of a trace as $l_i(t)$.

2.2 Definition of a leakage function

Following [31], a leakage function is an abstraction that models all the specificities of a side-channel (*e.g.* the power consumption or the EM radiation of a chip), up to the measurement setup used to monitor the physical observables.

Using the previous notations, it means that each leakage sample $l_i(t)$ in a leakage trace l_i is the output of a leakage function L_t. In our block cipher example, this leakage function takes at least a plaintext x_i and a secret key k as inputs. But in theory, the leakages take many more parameters into account. For this reason, Micali and Reyzin consider three input arguments, namely the target device's internal configuration C, a measurement parameter M and a random string R. Note that in order to be reflective of actual physical leakages, C has to contain all the configurations of the device before the computation corresponding

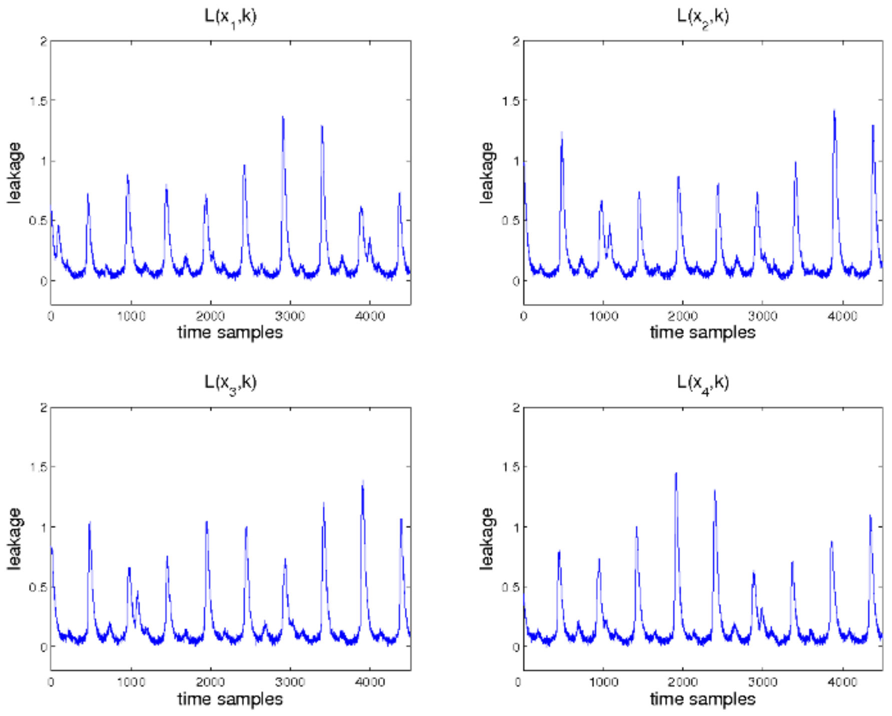


Fig. 2: Exemplary leakage traces.

to its current inputs, *i.e.* before the t^{th} sample of input x_i has been produced in our block cipher example. This incorporates the fact that the leakages can in principle be dependent on anything that has been computed prior to this time sample t, *e.g.* on the transitions between a former state and a current state in standard CMOS devices. Since Micali and Reyzin define a leakage function as a polynomial time function of its input arguments, this "history" is necessary to include forward secure primitives for which previous states are not polynomial time computable from current states. It yields:

$$l_i(t) = \mathsf{L}_t(\underbrace{(x_i, k, \ldots)}_{C}, M, R) \tag{1}$$

For convenience and in order to avoid unnecessarily heavy notations, we will omit the parameters of Equation (1) that are not directly useful for our discussions in the following of this paper (*e.g.* M, typically). We will also sometimes replace the generic state C by the parts of the state for which the leakage dependencies are exploited in a side-channel attack (*e.g.* inputs and keys).

We note that polynomial time functions actually correspond to more powerful leakages than what is usually observed in practice. As will be discussed in Section 5.1, leakage functions generally have a limited complexity of which the exact specification is an interesting direction to obtain more efficient side-channel resistant constructions. Also, a consequence of the previous generic definition is that we potentially have a different leakage function for every implementation[1]. It implies that any security analysis that one may perform in the context of side-channel attacks has to be positioned between two extremes:

1. On the one hand, we can analyze cryptographic primitives with respect to a generic adversary who is enhanced with an arbitrary leakage function. But then, generic and positive statements are hard to obtain and prove. They are also difficult to interpret for a specific device.
2. On the other hand, we can completely fix one instance of leakage function (*i.e.* perform an experimental attack against a given target device). But then, the conclusions obtained are only valid within this particular setting.

Quite naturally, an interesting approach would be to investigate the existence of intermediate contexts, *i.e.* to restrict the leakage function in such a way that conclusions can be drawn as long as the leakages fulfill a set of practically meaningful conditions. It is typically the approach followed, *e.g.* by [22, 33, 34].

Note that a concurrent solution for the analysis of physical security issues in cryptography is to rely on the existence of some minimum primitives from which it is possible to build secure devices. This corresponds, *e.g.* to the “minimal one way function” of Micali and Reyzin [31] or the “tamper proof” pieces of hardware in [17, 24]. As a matter of fact, these approaches are complementary. Namely, one focuses on the construction of physically secure objects while the other focuses on their exploitation in the construction of advanced functionalities.

3 Unpredictability *vs.* indistinguishability

Although protecting leaking devices against key recovery attacks is already difficult, ensuring security against this type of attacks is unfortunately not sufficient. One generally expects pseudorandom generators, functions or permutations to be indistinguishable from truly random. For example, we can refer to the formal definitions of security for symmetric encryption schemes in [4]. As an illustration, we use the real-or-random indistinguishability. In this setting, the adversary has access to an oracle $\mathsf{Enc}_k(\mathtt{RR}(\cdot, b))$ that takes an input message x and does the following: toss a coin b, if $b = 0$, return $\mathsf{Enc}_k(x)$, else return $\mathsf{Enc}_k(r)$, where $\mathsf{Enc}_k(x)$ is an encryption scheme (*i.e.* typically, our block cipher $\mathsf{E}_k(x)$ put into a certain mode of operation) and $r \overset{R}{\leftarrow} \mathcal{X}$ is a uniformly distributed random message.

[1] In [31, 42], an implementation is defined as the combination of a target device and a measurement setup. We use the same definition in this paper.

Definition 1. *Let* $\mathsf{Enc} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$ *be an encryption scheme and* A *be an algorithm that has access to the oracle* $\mathsf{Enc}_k(\mathtt{RR}(\cdot, b))$. *We consider:*

$$\mathbf{Succ}^{\mathrm{ror-ind}-b}_{\mathsf{Enc},\mathsf{A}} = \Pr[k \overset{R}{\longleftarrow} \mathcal{K} : \mathsf{A}(\mathsf{Enc}_k(\mathtt{RR}(\cdot, b))) = 1]$$

The ror-advantage of a chosen plaintext adversary A *against* Enc *is:*

$$\mathbf{Adv}^{\mathrm{ror-cpa}}_{\mathsf{Enc},\mathsf{A}} = \left| \mathbf{Succ}^{\mathrm{ror-ind-1}}_{\mathsf{Enc},\mathsf{A}} - \mathbf{Succ}^{\mathrm{ror-ind-0}}_{\mathsf{Enc},\mathsf{A}} \right|$$

We say that an encryption scheme Enc *is ror-indistinguishable if this ror-advantage is negligible for any polynomial time adversary.*

The central issue when trying to adapt this definition to a physically observable device is that in general, a leakage trace easily allows distinguishing real inputs/outputs from random ones. This can be intuitively understood by looking at Figure 3 where the leakage trace corresponding to an encryption $y_i = \mathsf{Enc}_k(x_i)$ is plotted. In this trace, we see that different dependencies can be observed and exploited: the beginning of the trace mainly leaks about the input x_i; the core of the trace leaks jointly about the input x_i and the secret key k; finally, the end of the trace mainly leaks about the the output y_i. In most practical side-channel attacks, the leakage is in fact sufficient to recover the secret key k, provided a sufficient amount of (different) encrypted plaintexts can be observed.

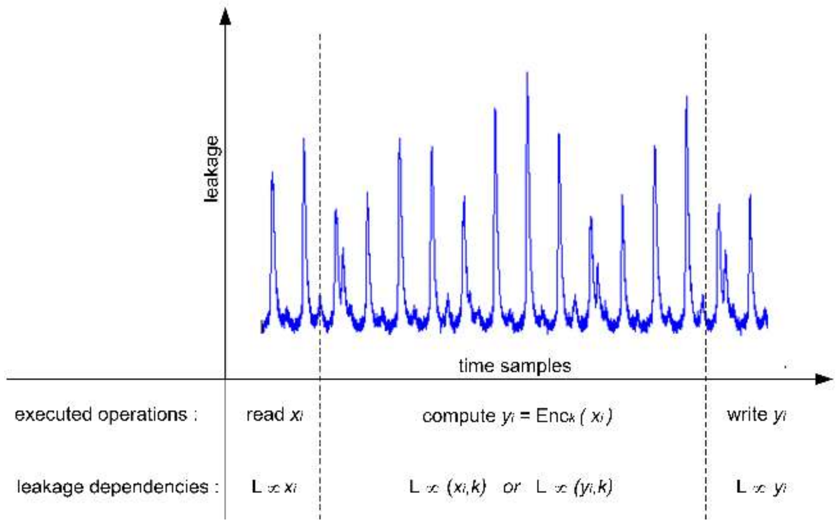


Fig. 3: Impossibility to assume indistinguishability for block ciphers.

Say now that the adversary does not have to recover the key, but to distinguish a real output $\mathsf{Enc}_k(x_i)$ from a random one $\mathsf{Enc}_k(r)$, given the input x_i and the leakage trace corresponding to the encryption of x_i. In fact, the leakage trace will easily allow doing this distinction. For example, imagine that some samples in the leakage trace depend on the Hamming weight $H_W(x_i)$, a frequently considered model in practice. With high probability, this Hamming weight will not be equal to $H_W(r)$. In other words, since we can hardly achieve implementations secure against key recovery, it is even harder to achieve implementations providing an indistinguishability notion. At least, this is definitely not something that we can take for granted. Note that for exactly the same reasons, the existence of durable functions and maximal one way functions as assumed in [31] cannot be considered as reasonable foundations for a leakage resilient cryptography.

The previous discussion suggests that "protecting" the inputs/outputs of a cryptographic algorithm against distinguishing attacks enhanced with physical leakages is a difficult task. Following this observation, a natural idea is to try analyzing simpler security notions for simpler primitives first. For example, [22, 31, 33, 34] consider PRGs. In this context, two security definitions can be considered, namely the next bit unpredictability introduced in [6] and the current output indistinguishability introduced in [49]. In a black box setting, these two notions are equivalent. But as pointed out in [31], this equivalence does not hold anymore with physically observable devices. The reason of this difference can be easily understood from the example in Figure 3. Intuitively, what essentially matters from a side-channel point of view is the word "*next*" in these definitions. That is, distinguishing current outputs from random ones is trivial if one can access the leakage corresponding to this output (as in the Hamming weight example). But predicting the next output bit may still be difficult in this case. Therefore, the following sections will consider a definition of security that corresponds to the next output indistinguishability (or equivalently to the next bit unpredictability). We denote this security notion as *physical indistinguishability.*

Definition 2. *Let $\mathsf{G}^q : \mathcal{K} \rightarrow \mathcal{K} \times \mathcal{X}^q$ be an iterative pseudorandom generator, with q iterations denoted as $[k_1, x_1] = \mathsf{G}(x_0)$, $[k_2, x_2] = \mathsf{G}(k_1)$, $\ldots$, $[k_q, x_q] = \mathsf{G}(k_{q-1})$. Let $\mathbf{l}_q = [\mathsf{L}(k_1, x_1), \mathsf{L}(k_2, x_2), \ldots, \mathsf{L}(k_q, x_q)]$ be the leakage vector corresponding to these q iterations. Let $\mathsf{P}^q = (\mathsf{G}^q, \mathsf{L})$ be the physical implementation corresponding to the combination of the pseudorandom generator G^q with the leakage function L. Let finally A be an algorithm that takes the plaintexts $\boldsymbol{x}_q = [x_1, x_2, \ldots, x_q]$ and leakages $\mathbf{l}_q$ as input and returns a bit. We consider:*

$$\mathbf{Succ}^{\mathrm{prg-ind-0}}_{\mathsf{P}^q,\mathsf{A}} = \Pr[k_0 \overset{R}{\longleftarrow} \mathcal{K}, [\mathbf{x}_{q+1}, k_{q+1}] = \mathsf{P}^{q+1}(k_0) \rightsquigarrow \mathbf{l}_{q+1} : \mathsf{A}(\mathbf{x}_{q+1}, \mathbf{l}_q) = 1)$$

$$\mathbf{Succ}^{\mathrm{prg-ind-1}}_{\mathsf{P}^q,\mathsf{A}} = \Pr[k_0 \overset{R}{\longleftarrow} \mathcal{K}, [\mathbf{x}_q, k_q] = \mathsf{P}^q(k_0) \rightsquigarrow \mathbf{l}_q; x_{q+1} \overset{R}{\longleftarrow} \mathcal{X} : \mathsf{A}(\mathbf{x}_{q+1}, \mathbf{l}_q) = 1)$$

The ind-advantage of A against P^q is defined as:

$$\mathbf{Adv}^{\mathrm{prg-ind}}_{\mathsf{P}^q,\mathsf{A}} = |\mathbf{Succ}^{\mathrm{prg-ind-0}}_{\mathsf{P}^q,\mathsf{A}} - \mathbf{Succ}^{\mathrm{prg-ind-1}}_{\mathsf{P}^q,\mathsf{A}}|$$

The implementation of a PRG is physically indistinguishable if the ind-advantage of any polynomial time adversary against this implementation is negligible.

We observe that, contrary to the definitions of Dziembowski and Pietrzak [12, 34], our definition is not adaptive in the sense that the leakage function is the predefined before the q iterations of the PRG. We believe that this definition is realistic since the information leaked is essentially a function of the targeted device rather than a choice of the adversary. Besides, letting the adversary select different leakage functions for different runs of the circuit results in an overly strong definition in many cases, as we will further discuss in Sections 5.1, 6.4.

4 Physical assumptions: local *vs.* global approach

Since the apparition of power analysis attacks in the late 1990s, various solutions have been proposed to counteract them. Among these countermeasures, a first category aims to provide design guidelines for the implementation of cryptographic primitives. That is, they study how to implement (*e.g.*) a block cipher is in such a way that it leaks as little as possible. Such local countermeasures have been intensively analyzed in the last ten years and typically include hiding [47] or masking [19]. Their limitation is that up to now, no solution allows to completely get rid of the leakages. For example, masking schemes have been shown vulnerable to higher-order attacks [32] and hiding countermeasures generally give rise to small data dependent leakages that can be exploited by an adversary. As a consequence, a complementary approach is to accept that cryptographic implementations leak a certain amount of information and try to use them in such a way that these leakages do not lead to complete security failures. In this global approach, one essentially assumes that a single iteration of the target cryptographic primitive "does not leak too much" in some sense.

In the following of this paper, we investigate this second option. It implies the need to define what is meant by "bounded leakage". For this purpose, the proofs in [34] assume a leakage of λ bits on a key K if this key is (computationally) indistinguishable from a distribution Y having an average min entropy conditioned on the leakage of $n-\lambda$ bits. This is formalized by the notion of HILL pseudoentropy. Here, we denote with $\delta^{\mathsf{D}}(K;Y)$ the advantage of a circuit D in distinguishing the random variables K, Y, *i.e.* $\delta^{\mathsf{D}}(K;Y) = |\Pr[\mathsf{D}(K)=1] - \Pr[\mathsf{D}(Y)=1]|$. We also define $\delta_s(K;Y)$ as the maximum of $\delta^{\mathsf{D}}(K;Y)$ taken over all circuits D of size s. We finally use the standard definitions:

Definition 3. *The min entropy of a random variable K is defined as:*

$$\mathrm{H}_\infty(K) = -\log \max_{k \in \mathcal{K}} \Pr[K = k],$$

and the average min entropy of K conditioned on L is defined as:

$$\mathrm{H}_\infty(K|L) = -\log\left(\mathbf{E}_{l \leftarrow L}\Big[\max_{k \in \mathcal{K}} \Pr[K = k | L = l]\Big]\right)$$

Then, we define the following computational analogues:

Definition 4. *K has HILL pseudoentropy n, denoted by $\mathrm{H}^{\mathsf{HILL}}_{\epsilon,s}(K) \geq n$, if there exist a distribution Y with min entropy $\mathrm{H}_{\infty}(Y) \geq n$ where $\delta_s(K;Y) \leq \epsilon$.*

Definition 5. *K has HILL pseudoentropy n conditioned on L, denoted by $\mathrm{H}^{\mathsf{HILL}}_{\epsilon,s}$ $(K|L) \geq n$, if there exists a collection of distributions Y_l (giving rise to a joint distribution (Y,L)), such that $\mathrm{H}_{\infty}(Y|L) \geq n$ and $\delta_s((K,L);(Y,L)) \leq \epsilon$.*

From such definitions, there are two possible research directions. First, one can investigate how to best exploit a λ-bit leakage, *e.g.* in the design of PRGs (this will be analyzed in Section 5). Second, it is also required to determine what is a reasonable value for λ, in practical settings. As already mentioned, this amount of leakage highly depends on the target device and adversarial strategy. Leakage traces can be made of gigabytes of data and the selection of a good decoding algorithm to extract λ bits of information is a quite challenging issue. As for classical cryptanalysis concerns, this is where reasonable assumptions have to be introduced in accordance with practical works in the area of side-channel attacks. We show in this section that the framework of [42] can be used for this purpose. Without entering into details that are out of the scope of this paper, the goal of this framework is to provide tools allowing one to evaluate a leaking implementation and a side-channel adversary. As summarized in Figure 4, it can be seen as an interface between practical and theoretical concerns raised by physically observable devices. When designing new attacks or local countermeasures, it allows determining their effectiveness with a combination of information theoretic and security metrics. For example, it is shown in [43] that an information theoretic metric nicely captures the impact of a local countermeasure such as masking. When building new cryptographic primitives, it allows to quantify the λ-bit leakage assumed in the proofs of constructions such as [12, 34].

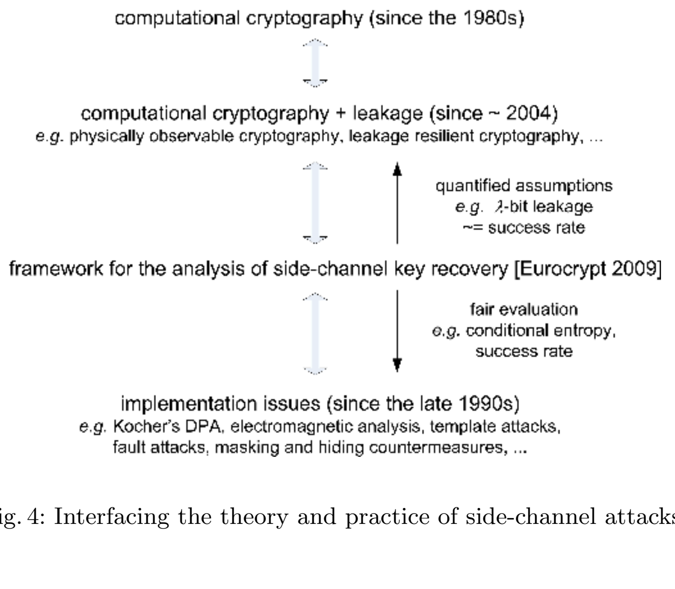

computational cryptography (since the 1980s)
computational cryptography + leakage (since ~ 2004)
e.g. physically observable cryptography, leakage resilient cryptography, ...
quantified assumptions
e.g. λ-bit leakage
~= success rate
framework for the analysis of side-channel key recovery [Eurocrypt 2009]
fair evaluation
e.g. conditional entropy,
success rate
implementation issues (since the late 1990s)
e.g. Kocher's DPA, electromagnetic analysis, template attacks,
fault attacks, masking and hiding countermeasures, ...

Fig. 4: Interfacing the theory and practice of side-channel attacks.

Literally, the min entropy $\mathrm{H}_\infty(X)$ is the negative logarithm of the probability that a variable X is determined correctly with only one guess of the form "is X equal to x?", with an optimal strategy [7]. Since an exact evaluation of the min entropy (and its computational analogues such as the HILL pseudoentropy) is generally hard to obtain, a simple assumption is to approximate it with the success rate of a side-channel key recovery attack (briefly recalled in Appendix A), as can be estimated in practical settings:

Assumption. *The HILL pseudoentropy remaining after a q-query side-channel key recovery attack has been performed against a key variable K is approximated by* $\mathrm{H}^{\mathsf{HILL}}_{\epsilon,s}(K|\boldsymbol{L}_q) \simeq -\log_2(\boldsymbol{Succ}^{sc\text{-}kr\text{-}1,K}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q))$*, where* $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ *is a "best available" adversary with time, memory and data complexity* τ*,* m *and* q*.*

Otherwise said, we have: $\lambda(\tau, m, q) \simeq n + \log_2(\mathbf{Succ}^{\text{sc-kr-1},K}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q))$. Exemplary values for λ in different contexts will be discussed in the next section. We mention that considering the success rate of specific attacks may appear as weak from a theoretical point of view. But this assumption is in fact imposed to some extent by the computational difficulty of perfectly estimating the side-channel leakage, as we now argue. That is, ideally, estimating λ would require, for each key value (that is the only unknown variable in a side-channel attack):

- to estimate the probability distribution of the leakage conditionally to this key,
- to compute the (*e.g.* average min) entropy by integrating this distribution.

For practical ciphers (*e.g.* the AES Rijndael), this would mean the estimation of 2^{128} distributions, which is unfeasible. Note that for large leakage traces (as usually observed in practice, see [44]), even the estimation of one distribution may be computationally hard (*e.g.* assuming that the leakages are normally distributed, it requires to estimate a covariance matrix of size $N_l \times N_l$). In order to avoid such limitations, actual side-channel attacks generally exploit the approximated probability distribution of a reduced set of leakage samples, and for enumerable subkeys, *i.e.* the generic template attacks detailed in [42].

As a consequence, there is a hardly avoidable gap between the λ-bit leakage assumed in the proofs and the λ-bit leakage that can be approximated in practice. Solutions to reduce this gap can also be devised in two directions. On the one hand, the approximations of λ should consider reasonable security margins, as in the practical security approach when designing block ciphers, *e.g.* in [25]. On the other hand, proofs could be based on weaker assumptions than $n - \lambda$ bits of HILL pseudoentropy. For example, one could try to exploit the unpredictability entropy defined in [36] (that is implied by HILL pseudoentropy):

Definition 6. *K has unpredictability entropy n conditioned on L, denoted by* $\mathrm{H}^{\mathsf{unp}}_s(K|L) = n$*, if for any* A *of size s it holds that* $\Pr[\mathsf{A}(L) = K] \leq 2^{-n}$*.*

Or, alternatively, one could assume that the leakage function is hard to invert (which seems a very minimal assumption), as in the work of Dodis *et al.* on cryptography with auxiliary input. We will use this assumption in Section 5.3.

Note finally that the parameters τ and m correspond to the circuit size s in our definitions of computational entropy. They allow considering the effectiveness of techniques combining side-channel leakages with classical cryptanalysis such as the collision or algebraic side-channel attacks introduced in [40] and [36, 37], *i.e.* attacks in which the time (and memory) complexity is non-negligible.

Analogy with classical cryptanalysis. Before moving to the security analysis of different PRGs, it is worth emphasizing that this situation, although not satisfying, is not very different than the one in classical (*e.g.* linear, differential) cryptanalysis. In this case, one also considers the best available attacks to evaluate the security of a cipher. Hence, local and global countermeasures are not contradictory. It is both required to know how to design implementations that do not leak to much and how to exploit such implementations in order to provide good cryptographic properties. By analogy, there is no contradiction between the wide trail strategy [9], that has been used in the design of the AES Rijndael (*i.e.* a type of local countermeasure against linear/differential attacks), and the proof that an encryption mode is secure if its underlying block cipher is indistinguishable from a pseudorandom permutation [4] (*i.e.* a more global approach). Similarly, countermeasures such as masking and hiding (or more formal solutions like [8, 16, 22]) can be used to design implementations with limited leakages that can then be used in secure PRG (or other) constructions.

5 Leakage resilient PRGs

5.1 On the difficulty of modeling a leakage function

An appealing solution to build PRGs secure against side-channel attacks is to consider forward secure primitives, *e.g.* as introduced in [5] for symmetric encryption. In this section, we analyze the "future computation attack" to illustrate the need of a new type of construction in [34]. This attack can be explained from Figure 5, in which a length doubling PRG is denoted as 2PRG and the l states corresponding to l iterations of the arbitrary length PRG are denoted as S_i.

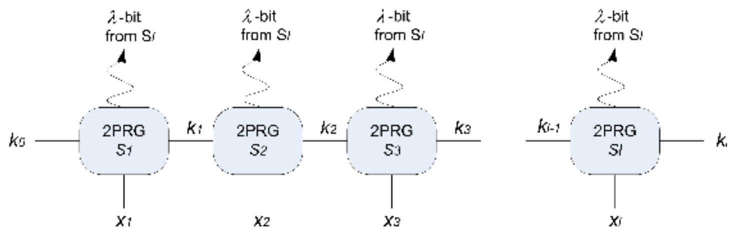


Fig. 5: The "future computation attack".

In [34], the leakage of each iteration of the 2PRG is bounded to λ bits. As discussed in the previous section, this is a reasonable abstraction. But the λ bits leaked by the computation of a state S_i can be selected adaptively by the adversary, as long as they are a polynomial time function of this state. Therefore, a straightforward attack depicted on the figure is to select λ bits of S_l during the computation of S_1, another λ bits of S_l during the computation of S_2, ... until the state S_l is completely revealed which trivially (and completely) breaks the security of the scheme. Looking at the physical reality, this attack is obviously an overestimation of what a side-channel adversary could achieve. In general, the leakage function is not selected by the adversary, at least not arbitrarily. It is rather a feature of the target device. In certain settings (*e.g.* electromagnetic leakages), one could imagine that the antenna is moved adaptively during the attack. But it at least remains that a leakage function does never leak about future computations. In reaction to this type of (unrealistic) attacks, three different positions could be adopted, that we now detail:

1. One can consider stronger assumptions for the 2PRG. For example, in the ideal cipher model, the outputs of any iteration of the PRG are uniformly random and the computation of a state S_i cannot leak about any state S_j with $j > i$. This is typically the solution considered in [33].
2. Another solution is to keep the model as it is (*i.e.* unrealistic with respect to the physics) and to build constructions that can even cope with this type of leakage. This is typically the solution considered in [34].
3. Eventually, a more elegant solution is to restrict the leakage function in a meaningful way. A natural direction with this respect would be to limit the complexity of this function (*e.g.* in terms of circuit size).

The goal of the next section is to analyze the security of different PRGs against side-channel attacks. For simplicity, we selected the forward secure PRG from [5] and the leakage resilient PRG from [34], pictured in Figure 6.

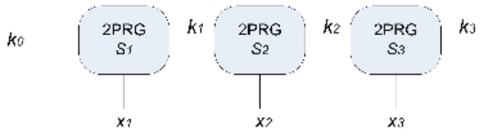
Note that the block cipher based PRG from [33] could be similarly investigated (*i.e.* it has the same properties as [5] in terms of leakage resilience). Following these papers, we aim to work out the question: "is the alternating structure of [34] a requirement for leakage resilient PRGs or is a forward secure primitive such as [5, 33] sufficient to withstand side-channel attacks?".

5.2 Theoretical security analysis and limitations

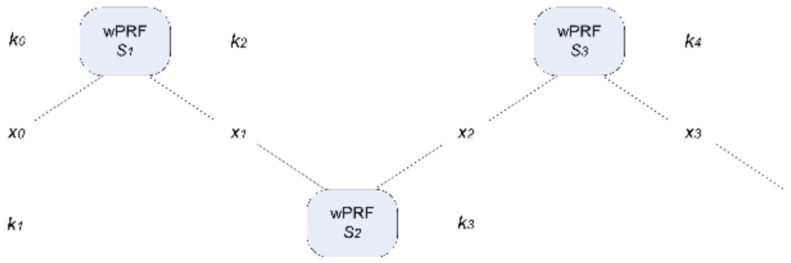
We start with an intuitive description of the two approaches.

Alternating structure. As previously mentioned, a central difficulty when modeling a leakage function is the fact that polynomial time computations from a state S_i may potentially leak information about future states S_j with $j > i$. The solution proposed in [34] may be summarized as follows:

1. Double the key size and use an "alternating structure" such as in the lower part of Figure 6 in which wPRF is a weak pseudorandom function (*i.e.* a PRF in which the inputs are not chosen by the adversary but at random).



(a) Forward secure PRG from [5].



(b) Leakage resilient PRG from [34].

Fig. 6: Two PRG constructions

2. Assume that when computing the odd states (*i.e.* $S_1, S_3, \ldots$), the even states (*i.e.* $S_2, S_4, \ldots$) are not manipulated by the device and therefore do not leak.

As a result, the sequence of keys $k_0, k_2, k_4, \ldots$ cannot be determined from the sequence $k_1, k_3, k_5, \ldots$ It implies that, *e.g.* when leaking λ bits about S_1, the states S_i for any $i > 1$ are not polynomial time computable from S_1 because k_1 is still safe. This prevents the future computation attack. The main limitation of this proposal is that it is not clear if the alternating structure is only motivated by the proof technique (*i.e.* the willing to be in the standard model) or if there is also a physical concern that makes it necessary. Its main advantage is the security analysis combining physical and classical security notions (whereas, *e.g.* [33] was dealing with both issues separately, in a more specific scenario).

Forward secure PRGs. Alternatively, one can prevent side-channel attacks with a forward secure primitive without alternating structure. But if no additional restrictions are considered for the leakage function, proving the security against the future computation attack then requires the assumption that the 2PRGs in the construction of Figure 6a behave as random oracles that the leakage function cannot query. Interestingly, this solution also allows to get rid of certain physical assumptions, as will be detailed in the next section.

The main security features of these two approaches are listed in Table 1. In summary, we have an efficient construction that requires strong black box assumptions on the one hand. And on the other hand, we have a less efficient construction proven in the standard model. It is worth to emphasize that the random oracle is only used to prove the leakage resilience of [5]. Yet, in the black box setting, this construction remains secure in the standard model. In other words, the random oracle essentially prevents the “future computation attack”, which seems a reasonable abstraction, while keeping the proofs simpler.

	Forward secure PRG [5]	Alternating structure [34]
Black box assumptions	random oracles	standard model
Leakage assumptions	non-invertibility [10] + non-adaptivity	HILL pseudoentropy + "independent leakages"
Key size	n	$2n$
Output bits per round	n	n
Exhaustive key search	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Construction	2PRG $\rightsquigarrow$ PRG	wPRF $\rightsquigarrow$ PRG
Tolerated leakage	$\lambda = \alpha n$ (with $\alpha \in [0,1]$)	$\lambda \simeq \log_2(n)$ (if wPRF secure against poly. size adversaries) $\lambda = \alpha n$ (if wPRF secure against exp. size adversaries)

Table 1: Summary of the security features for two leakage resilient PRGs.

5.3 Proving leakage resilience with random oracles

In this section, we provide a proof of security for the PRG depicted in Figure 6a. For this purpose, we first detail the three assumptions that we require for our proof to hold and discuss their pros and cons compared to the approach in [34].

a. Random oracles. In order to ensure the locality of the leakages, we model $\mathsf{2PRG} : k_i \rightarrow (k_{i+1}, x_{i+1})$ as a random oracle mapping values from $\{0,1\}^n$ to values in $\{0,1\}^{2n}$. Then, during the i-th invocation of $\mathsf{2PRG}$, the attacker receives two leakages associated to this evaluation, $\mathsf{L}_i^i(k_{i-1})$ and $\mathsf{L}_i^o(k_i, x_i)$, together with the output x_i of the execution of $\mathsf{2PRG}$. Most importantly, we assume that the leakage functions cannot query $\mathsf{2PRG}$, which makes it impossible to use them to obtain information about previous or future invocations of $\mathsf{2PRG}$.

The fact that we model $\mathsf{2PRG}$ as a random oracle that the leakage function cannot query corresponds to the experimental observation that any useful information obtained through the side-channels of a circuit implementation is related to simple functions of the inputs and outputs of that circuit, but will not provide any sophisticated function that is not present in the circuit state, *e.g.* a future output of the PRG. This, in particular, appears to be realistic for circuits implementing block ciphers where the measured information can be interpreted as a simple function of the block cipher input and key during the first few rounds of the block cipher computation, and/or as a simple function of the block cipher output and key during the last few rounds of the block cipher computation, but where any useful function of the block cipher input and output remains elusive (or would be the sign of a completely broken implementation, *e.g.* as in [36, 37]).

b. Bounded leakage per iteration. Formally, we require that the leakages given to the adversary preserve the secrecy of the PRG seed in the following sense: the probability that an adversary recovers the seed used as input or provided as output during two iterations of the PRG construction should be small. Considering two iterations is minimal here since half of the output of an iteration is the input of the next iteration, and there are therefore two leakages taken on each secret variable. This is formalized through the following definition.

Definition 7. *Let* $(\mathsf{L}^o, \mathsf{L}^i)$ *be a pair of functions,* $\mathsf{A}^{\mathsf{2PRG}}$ *an algorithm representing the side-channel adversary with oracle access to* $\mathsf{2PRG}$*,* n *a fixed integer, and* $\mathsf{Pr}_{\mathsf{Guess}}(n)$ *the following probability:* $\Pr[\mathsf{A}^{\mathsf{2PRG}}(\mathsf{L}^o(k_1, x_1), x_1, \mathsf{L}^i(k_1)) = k_1 : k_0 \leftarrow \{0,1\}^n; (k_1, x_1) := \mathsf{2PRG}(k_0)]$*. The pair* $(\mathsf{L}^o, \mathsf{L}^i)$ *is said to be* ϵ-seed-preserving *for security parameter* n *and* $\mathsf{A}^{\mathsf{2PRG}}$ *if* $\mathsf{Pr}_{\mathsf{Guess}}(n) \leq \epsilon$*. A pair of functions* $(\mathsf{L}^o, \mathsf{L}^i)$ *is said to be* seed-preserving *if, for every PPT* $\mathsf{A}^{\mathsf{2PRG}}$*, there is a negligible function* ϵ *such that* $(\mathsf{L}^o, \mathsf{L}^i)$ *is* $\epsilon(n)$*-seed-preserving for every security parameter* n *and* $\mathsf{A}^{\mathsf{2PRG}}$ *running on input* 1^n*. A sequence of pairs of functions* $(\mathsf{L}^o_1, \mathsf{L}^i_1), \ldots, (\mathsf{L}^o_l, \mathsf{L}^i_l)$ *is said to be* uniformly seed-preserving *if, for every PPT* $\mathsf{A}^{\mathsf{2PRG}}$*, there is a negligible function* ϵ *such that each pair of this sequence is* $\epsilon(n)$*-seed-preserving for every security parameter* n *and* $\mathsf{A}^{\mathsf{2PRG}}$ *running on input* 1^n*.*

Assuming that adversaries only receive outputs of seed-preserving functions is in fact similar to the assumptions made in the cryptography with auxiliary input setting [10]. We believe it captures physical leakages particularly well in the sense that we do not put any constraint on the form of the leakage: it can be a simple computation time, a huge sequence of power consumption measurements, a map of electromagnetic radiations, or anything else. Moreover, it might very well be the case that the full information about the $\mathsf{2PRG}$ seeds is included in these leakages. We only require that the leakage functions cannot be inverted efficiently. Overall, this is a weaker assumption than requiring a high HILL pseudoentropy as in [12, 34]. But when to be quantified in practice, it also suffers from the gap described in Section 4. That is, we can hardly evaluate the performance of every possible adversary and need to rely on specific attacks.

Stronger versions of these notions of seed preservation would provide the extra-ability to the adversary to check whether a condidate key k_1 is the correct one. This can happen in different contexts in practice: it might be the case that half of $\mathsf{2PRG}(k_1)$ is available as public output of a next round, enabling the adversary to perform some comparisons; it might also be the case that the adversary is able to re-initialize the circuit to a value of his choice, and to compare the leakages observed in the targeted execution to the leakage occurring in an execution triggered after re-initialization. The security of the PRG and PRF construction, as claimed next in Theorems 1 and 2, could be rephrased in terms of this strengthened notion of seed preservation. The proofs would remain the same, except that the reductions would become tighter by a factor corresponding to the number of random oracle queries made by the adversary.

c. Only the current iteration leaks. We assume that the leakage of each state S_i only depends on its current inputs/outputs and is independent of the previous states. This is a reasonable restriction in regard to most experimental side-channel attacks. But in fact, even if history effects were observed in the leakages (*e.g.* if the leakage in state S_i was dependent on k_{i-2}), it would be possible to relax this assumption by simply generalizing the seed-preserving property to more than two PRG iterations. We then would require, for instance, that a triple of leakage functions $(\mathsf{L}_1, \mathsf{L}_2, \mathsf{L}_3)$ with appropriate inputs does not give any PPT adversary a non negligible advantage in guessing any of the involved secrets.

Finally, it is worth emphasizing that we do not need the "only computation leaks" assumption that was first introduced in [31]. This is interesting since in advanced CMOS technologies (65 nanometer or smaller), the power consumption is not dominated by its dynamic part anymore and the so-called "static leakages" start to play a significant role. This means that leakages happen independently of the fact that computations are performed. In our setting, these non computational leakages can simply be given to the adversary in Definition 7, as long as they provide a negligible advantage in guessing the seeds.

Similarly, we do not need the condition of independent leakages as in [34]. In this respect, and contrary to what is claimed in [35], this independence requirement is not a completely mild assumption and it may be contradicted by coupling effects in microelectronic devices. As an illustration, [2] suggests different leakage models that can be observed, among which linear and quadratic ones. If we denote by $S(i)$ the i-th bit of a state S, it yields:

$$\mathsf{L}_{quad}(S) = \underbrace{\sum_i \alpha_i \times S(i)}_{\mathsf{L}_{lin}(S)} + \sum_{i,j} \beta_{i,j} \times S(i) \times S(j) + \cdots$$

Clearly, if the quadratic term (that typically captures coupling effects) is non negligible and S contains two consecutive states of the alternating structure, their leakage are not independent (*i.e.* depend on the combination of both parts). Note that even if partially unverified, none of these assumption has the potential to break the constructions in practice. But quantifying them empirically and reflecting non-computational or dependent leakages in the proofs would increase their relevance. Summarizing, the combination of an alternating structure and the assumption of independent leakages can be seen as the counterparts that one has to pay in order to get rid of the random oracles in the proofs of [34].

d. Security of the forward secure PRG of Figure 6a. We define the security for the PRG construction of Figure 6a through the experiment $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$ during which the adversary $\mathsf{A}^{\mathsf{2PRG}}$ tries to predict something about the next round's output of PRG given the output and leakages from the past rounds, where the leakages are taken from a family of leakage functions $\overline{\mathsf{L}} := \langle \mathsf{L}_1^i, \mathsf{L}_1^o, \mathsf{L}_2^i, \mathsf{L}_2^o, \ldots \rangle$.

Experiment $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$:

1. During initialization, a key k_0 is selected uniformly at random in the set $\{0,1\}^n$, and a counter i is set to 0.
2. On input 1^n, adversary $\mathsf{A}^{\mathsf{2PRG}}$ starts sending $\mathsf{request}$ queries. On each $\mathsf{request}$ query, the counter i is incremented, the pair (k_i, x_i) is computed as $\mathsf{2PRG}(k_{i-1})$, and the leakages $\mathsf{L}_i^i(k_{i-1})$ and $\mathsf{L}_i^o(k_i, x_i)$ are returned to $\mathsf{A}^{\mathsf{2PRG}}$, together with x_i.
3. When $\mathsf{A}^{\mathsf{2PRG}}$ outputs a test query, a bit $b \in \{0,1\}$ and a value $r \in \{0,1\}^n$ are chosen uniformly at random, and r is given to $\mathsf{A}^{\mathsf{2PRG}}$ if $b = 0$ or x_{i+1} is given otherwise, computed as $(k_{i+1}, x_{i+1}) := \mathsf{2PRG}(k_i)$.
4. $\mathsf{A}^{\mathsf{2PRG}}$ outputs a bit b', and $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$ is set to 1 iff $b = b'$.

We show that, as long as the pairs of leakage functions that are evaluated on the same keys are uniformly seed-preserving and can be evaluated efficiently, no efficient adversary can efficiently predict the output of the next round of the PRG from the outputs and leakages of the past rounds. That is:

Theorem 1. *Let $\mathsf{A}^{\mathsf{2PRG}}$ be a PPT adversary playing the $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$ experiment. Then, we have $\Pr[\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n) = 1] = \frac{1}{2} + \mathsf{negl}(n)$, provided that the family of pairs of leakage functions $(\perp, \mathsf{L}_1^i)$, $(\mathsf{L}_1^o, \mathsf{L}_2^i)$, ... is uniformly seed-preserving and that all leakage functions can be evaluated in probabilistic polynomial time.*

In other words, the implementation of the PRG in Figure 6a is physically indistinguishable (as in Definition 2). The proof is given in Appendix B.

5.4 Practical security analysis

The previous section provided an overview of the theoretical security analysis for two PRGs. Given a "small enough" leakage, these two constructions are expected to be secure against side-channel attacks. In order to observe these results for real devices, it then remains to evaluate exactly how much information is leaked in practice. For this purpose, a first step is to instantiate the 2PRGs and wPRFs that are necessary to implement the leakage resilient PRGs. For illustration and because they are usual targets of practical works in the area of side-channel attacks, we used the block cipher based constructions in Figure 7, following the suggestion of Pietrzak [34] for his wPRF implementation.

As detailed in Section 4, a reasonable estimation of the leakage in one iteration of a PRG is given by the success rate of the "best available adversary" for which the most important parameter is the data complexity q. A consequence is that the practical security of a construction is mainly determined by the number of different traces that can be exploited to identify each secret key. In other words, the practical security of a construction can be related to the notion of q-limited adversary that has been formalized by Vaudenay in his decorrelation theory [48]. For example, in the context of block ciphers, it yields:

Definition 8. *An adversary against a block cipher $\mathsf{E}_k(x)$ is q-limited for a key k if he can only observe the encryption of q different plaintexts under this key.*

Following this definition, it is clear that the 2PRG and wPRF of Figure 7 are similar in terms of practical security. For both constructions, one iteration limits the side-channel adversaries to two queries. We can then refine the definition:

Definition 9. *A block cipher based construction is q-limiting for a side-channel adversary A if this construction limits the number of different encryptions performed under a single key that A can observe (i.e. the data complexity) to q.*

Of course, having a q-limiting construction is not a sufficient condition to be secure against side-channel attacks. Overall, we need a combination of theoretical security (*i.e.* the physical indistinguishability discussed in Sections 3, 5.2, 5.3)

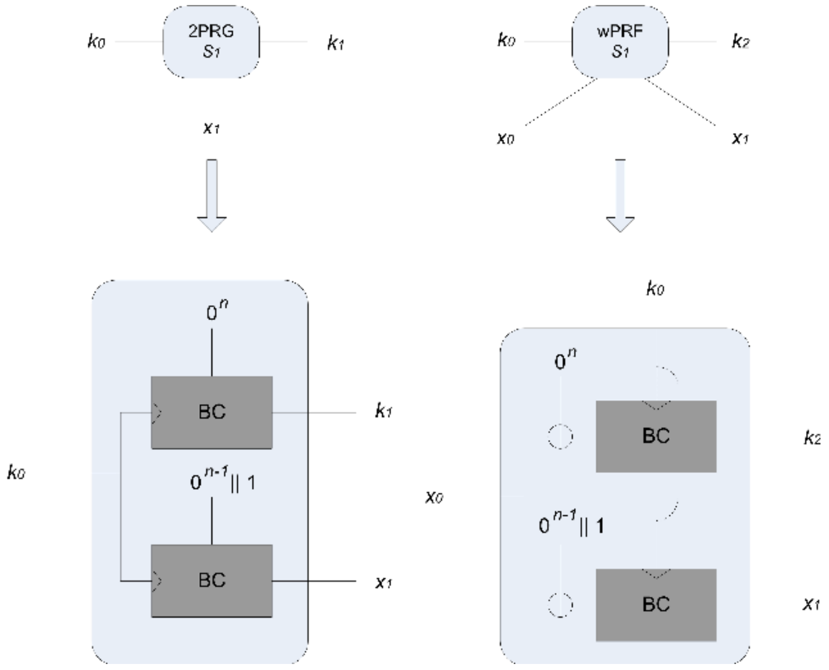


Fig. 7: Instantiation of a 2PRG and a wPRF with the AES Rijndael.

and practical security. But the q-limit allows hardware designers to know how much leakage they need to face. If a construction is 2-limiting, their goal will be to limit the success rate of the best available adversary for $q = 2$ queries. As an illustration, Table 2 provides the estimated success rates for different attacks against implementations of the DES and AES Rijndael.

Algorithm	Device	Attack	Ref.	$q = 2$	$q = 10$	$q = 100$
AES	PIC 8-bit controller	algebraic	[37]	≈ 1	≈ 1	≈ 1
AES	Atmel 8-bit controller	hexavariate templates	[45]	$\approx 2^{-16}$	≈ 1	≈ 1
AES	Atmel 8-bit controller	correlation	[45]	$\approx 2^{-128}$	$\approx 2^{-37}$	≈ 1
DES	64-bit ASIC	DoM test	[46]	$\approx 2^{-56}$	$\approx 2^{-56}$	$\approx 2^{-12}$

Table 2: Approximated success rates for different attacks.

These results illustrate that small differences in data complexity can lead to largely different success rates. Recall that side-channel attacks can be performed in various contexts (*e.g.* allowing the profiling of a device or not) which explains this large variance. Hence, the table makes a case for evaluating the metrics of [42] in various experimental settings, *e.g.* for protected circuits. Initiatives such as the DPA Contest [11] could typically be used for this purpose.

Note that the table already suggests that, for small devices (like 8-bit controllers), a reasonable λ-bit leakage can hardly be achieved if no additional (local) countermeasures are considered. For example and when applicable, the 1-limited algebraic side-channel attacks in [37] leak the complete AES key.

We also mention that it is important to distinguish the data complexity of a side-channel attack from the number of measurements performed by the adversary. Indeed, for a given data complexity q, the adversary may wish to repeat the measurement of his q different traces several (say n_r) times. For example, the left part of Figure 8 illustrates the impact of measuring the same leakage trace l_1 three different times (giving rise to measurements l_1, l_1' and l_1''). Intuitively, the combination of these measurements can be used to remove the noise from the traces, *e.g.* in order to obtain a more precise information about an intermediate value Y in the target cryptographic computation. This information can then be translated into information about a subkey S. Here the data complexity is $q = 1$ and we have $n_r = 3$. By contrast, standard DPA attacks generally try to combine the leakage corresponding to different plaintexts. As illustrated in the right part of Figure 8, each trace then brings a different information about the target intermediate value Y. And by translating these leakages into subkey information, one can have the intersection between the set of possible subkeys arbitrary small. Here, the data complexity is $q = 2$ and we have $n_r = 1$.

Following this observation, the expectation when using q-limiting constructions is that even if a polynomial time adversary can remove a significant part of the noise from his side-channel traces (*e.g.* the ones in Figure 2), there remains some uncertainty on the key because of the bounded data complexity. At least, one can hope that it is easier for hardware designers to guarantee this condition than to bound the success rate for a non limiting construction.

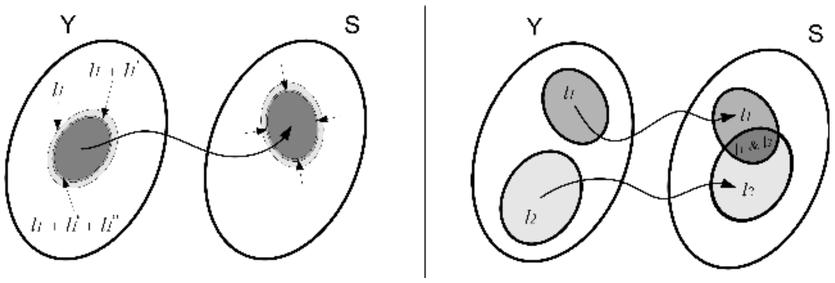


Fig. 8: Repeating a measurement *vs.* measuring a new trace.

We conclude this section with two remarks.

Remark 1. It is interesting to mention that, depending on the block cipher used in the 2PRG construction of Figure 7, such an instantiation may introduce a gap with the assumptions of Section 5.3. Just observe that we consider the leakage on the output of a 2PRG $\mathsf{L}^o(k_i, x_i)$ and the one on its input $\mathsf{L}^i(k_{i-1})$ as independent. But if a block cipher with an invertible key scheduling algorithm (*e.g.* the AES Rijndael) is used in Figure 7, the output leakage may potentially leak on the key that was used to produce this output. This observation is not expected to modify the practical security of the resulting PRG (a similar leakage actually occurs for the wPRF construction too), but suggests that carefully instantiating block cipher based constructions may be important to prevent side-channel attacks. It also recalls that non-invertible key scheduling algorithms (*e.g.* as in the FOX block cipher [23]) are desirable in the context of leaking devices. Alternatively, strictly respecting the assumption of Section 5.3 may require some performance overheads, *e.g.* by replacing $\mathsf{2PRG}(x) := \big(\mathsf{BC}_x(0^n), \mathsf{BC}_x(1||0^{n-1})\big)$ by a slightly more expensive one, *e.g.* $\mathsf{2PRG}(x) := \big(\mathsf{BC}_{\mathsf{BC}_x(0)}(0^n), \mathsf{BC}_{\mathsf{BC}_x(0)}(1||0^{n-1})\big)$.

Remark 2. It is also worth noticing that by adding more block ciphers to the 2PRG example of Figure 7, one can easily extend it towards a 3PRG, 4PRG, ... This leads to a simple tradeoff between security and performance. Namely, for a given q-limit, one can use a qPRG in Figure 6a and consequently generate $\frac{n\cdot(q-1)}{q}$ pseudorandom bits per block cipher execution.

6 Initialization issues

6.1 Breaking [34] with a standard DPA

The claim in [34] is that the alternating structure in Figure 6b can be used as a leakage resilient stream cipher. This implies the need of an initialization process with a public initialization vector. For this purpose, the authors suggest to pick up the keys k_0 and k_1 as well as the public x_0 at random. But this is in fact in contradiction with the practical requirements of a stream cipher. It is generally expected that one can re-initialize or re-synchronize a stream cipher without picking up new keys at random (*e.g.* see [15]). This is important, *e.g.* if the stream cipher has to be used in a challenge response protocol. Unfortunately, using x_0 as an initialization vector without changing the keys k_0 and k_1 in the same time leads to straightforward DPA attacks that break the stream cipher of [34] in practice. Just observe that, because of the AES based construction in Figure 7, the adversary targeting the alternating structure for a given x_0 is 2-limited. Say now the adversary re-initializes the PRG with multiple random x_0 values, *e.g.* say he does it t times with the same key (k_0, k_1). Then, the first iteration of the PRG is not 2-limited anymore but $2 \cdot t$-limited, where t can be arbitrarily large. As a consequence, the construction is no more leakage resilient. A standard DPA attack such as in the right part of Figure 8 can be applied.

Summarizing, either the alternating structure from [34] is limited to a fixed x_0 and its proof holds - but then, the resulting stream cipher cannot be efficiently re-initialized, re-synchronized, used in a challenge response protocol, ... Or it allows the adversary to observe multiple x_0 values - but then it is insecure. In other words, there is a component missing in this construction.

6.2 Secure initialization process

In order to avoid the previous issue, [33] proposed a secure initialization process of which different variants can be designed. Essentially, the idea is to bound the increase of the q-limit during initialization at the cost of some performance overheads. This can be simply achieved by adding multiplexors in the implementation and using the selection bit(s) to insert the random IV progressively in the PRG iterations. In its original form, pictured in Figure 9, this process was made of two phases (we refer to the original publication for the details):

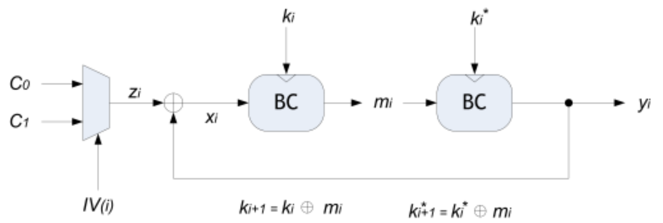


Fig. 9: Secure initialization process from ASIACCS 2008.

1. In a first (initialization) phase, n iterations of the PRG are executed with inputs z_i's that are obtained as follows: $z_i = C_0$ if $IV(i) = 0$, $z_i = C_1$ if $IV(i) = 1$, where $IV(i)$ is the i^{th} bit of the initialization vector and C_0, C_1 are two constants. The n first y_i values are not transmitted as output.
2. In a second (generation) phase, the PRG goes on iterating with a fixed input C_0 and now transmits the pseudorandom blocks y_i as outputs.

In this basic proposal and if a n-bit IV is considered, the secure initialization requires n iterations of the PRG before pseudorandom blocks can be produced. The advantage of this constructions is that it is still 2-limiting. Trading performances for security is again possible by adapting the q-limit and using larger multiplexors (*i.e.* incorporating more IV bits in each iteration).

Applying this idea to the PRGs of [5] and [34] can then be done in different ways. The simplest one is to use the IV bits to select which of the 2PRG and wPRF outputs is used as a key in the next round (*i.e.* if $IV(i) = 0$, use the lower outputs in Figure 7, if $IV(i) = 1$, use the upper one) - again without generating any pseudorandom block during initialization. Adding a multiplexor as in Figure 9 and XORing its output with the plaintexts (or keys) in the PRG constructions of Figure 6 is another solution that also allows various tradeoffs (but may requires a more careful analysis). The next section details yet another simple solution that we only apply to the forward secure PRG of [5] for convenience.

6.3 A more elegant (and standard) construction

In fact, the initialization of [33] can be directly related to the construction for pseudorandom functions introduced by Goldreich, Goldwasser and Micali in 1986 [18]. This leads to the simple process represented in Figure 10. In its basic form (in the left part of the figure), this construction can be viewed as a binary tree of depth n. The value of the root equals k_0. Then, depending on the IV bits, the left or right outputs of the 2PRG are selected. After n iterations of the 2PRG, we obtain a key k_0' that has been initialized with a public IV and can be used as a seed in PRG of Figure 6a. Again, it is possible to reduce the depth of the tree (*i.e.* to increase performances) by increasing the q-limit (*e.g.* the right part of Figure 10 uses 4PRGs with $q = 4$ and $n/2$ iterations), leading to the tradeoff:

- Number of block cipher executions for initialization: n/β,
- q-limit: 2^β, where β is an integer $\in [1, n]$.

The security of such an initialization process will be discussed in Section 7.

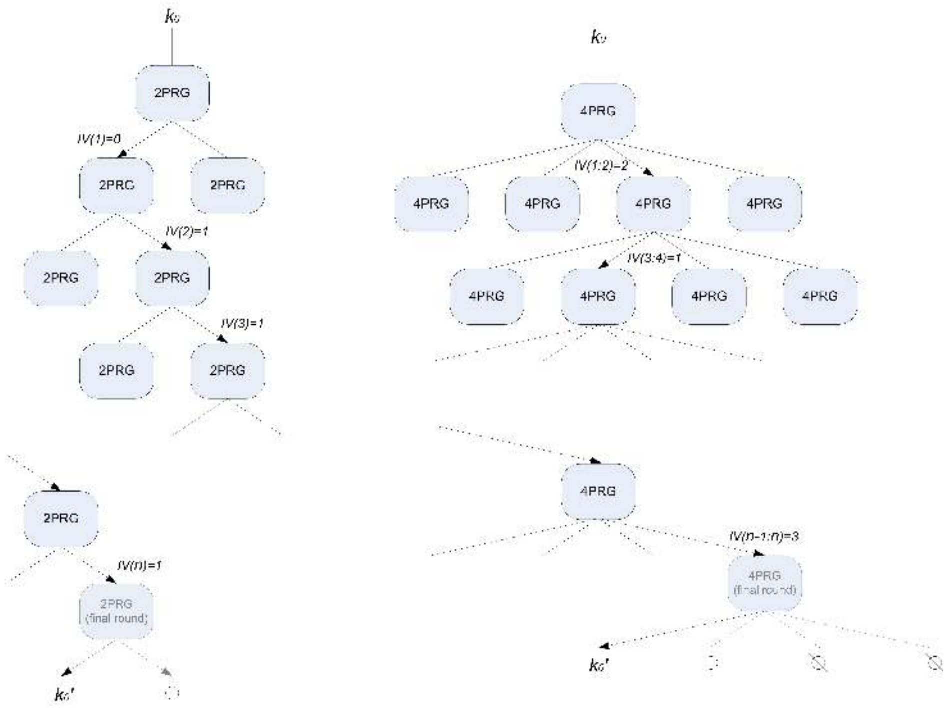


Fig. 10: Leakage resilient PRF. Left: n-iteration, Right: $\frac{n}{2}$-iteration.

Remark 3. Combining the PRG in the upper part of Figure 6 with the initialization process of Figure 10 leads to an overall tradeoff between security and performances. As an illustration, two extreme solutions are given in Appendix, Figure 11. In the upper part, a completely trusted block cipher is exploited. Hence, a 2^n-limited construction is tolerated, with an initialization process in one block cipher execution and producing n pseudorandom bits per block cipher execution. In the lower part, the block cipher has significant leakages. Hence, a 2-limited construction is used, with an initialization process in n block cipher executions and producing $n/2$ pseudorandom bits per block cipher execution.

6.4 Remark on the impossibility of a secure initialization process with an adaptive selection of the leakages

It interesting to highlight that a secure and efficient initialization process (*i.e.* where re-synchronization does not imply a new key exchange) is incompatible with a fully adaptive selection of the leakage function for stateless devices (*i.e.* devices that do not save any part of their state between two reinitializations). Even the previous proposals can be broken in this context. Indeed, if a new leakage function can be chosen anytime the PRG is initialized with the same input, then a "future computation attack" exploiting only the first iteration of the initialization process can be mounted. This can be intuitively related to Figure 8. In fact, the goal of a secure initialization process is to keep the q-limit as small as possible. Say for illustration that we have a 1-limiting process. Then, the expectation is that repeating the measurement of the same trace l_1 can only be used to remove noise, as in the left part of the figure. Good implementations should be such that this is not sufficient to leak the whole subkey S. But say now the adversary can adaptively choose his leakage function anytime he measures the trace corresponding to the same input plaintext x_1. Then, DPA attacks can again be mounted as in the right part of the figure. Quite problematically, such an attack would imply that any leaking implementation can be broken in linear number of measurements, making it impossible to build a secure device.

One solution to prevent the previous issue could be to limit the adaptivity of the leakage function to different inputs. That is, one could require that the same inputs to the target device should always give rise to the same leakage function. But from an operational point of view, the adaptivity of the leakage function relates to the possibility to change the measurement conditions during a side-channel attack (*e.g.* the antenna's position in an electromagnetic analysis). Hence, whatever modification of the setup that an adversary can do when changing the inputs can also be done when these inputs are kept constant. Therefore, if we assume that the leakage function cannot be chosen adaptively by the adversary when inputs are kept constant (which is mandatory in order to avoid trivial attacks as just described), there is no reason to allow it for variable inputs. We conclude that the leakage function should not be considered as adaptively selected by the adversary. In fact, a better solution to reflect the possible adaptivity of the measurement conditions is to include this adaptivity in the adversary's abilities and to quantify it directly in the λ-bit leakage bound.

7 Generalization to PRFs

In addition to its interesting features for initializing a PRG, the construction of Figure 10 can also be used as a PRF, if the IV is replaced by an input x. Proving the security of these constructions can be done using essentially the same technique as in Section 5.3 and Appendix B. Namely, each IV determines one trail through the tree of Figure 10. And each of these trails is similar to one iteration of a PRG. As in Section 3, we need to extend the black-box security definition of a PRF (against adaptive queries) from [18] to physical indistinguishability in presence of all leakages except for the challenge (here meaning that the output to be distinguished from random does not come with a leakage). That is:

Definition 10. *Let* $\mathsf{F} : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{X}$ *be a pseudorandom function and* $\mathsf{P} = (\mathsf{F}, \mathsf{L})$ *be the physical implementations corresponding to the combination of* F *with a leakage function* L. *Let finally* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *be a pair of PPT algorithms where* A_1 *takes a set of plaintexts* $\mathcal{S}$ *and some information* $\mathcal{I}$ *as input and outputs a plaintext, while* A_2 *takes a plaintext, a ciphertext and some information* $\mathcal{I}$ *as input and outputs a bit. We consider the following experiments:*

```
$\mathbf{Exp}^{\mathrm{prf-ind-0}}_{\mathsf{P},\mathsf{A}}$
$\mathcal{S} = \varnothing$;
$\mathcal{I} = \varnothing$;
$k \xleftarrow{R} \{0,1\}^n$;
for $i = 1 : q$
    $x_i \leftarrow \mathsf{A}_1(\mathcal{S},\mathcal{I})$;
    $\mathcal{S} = \mathcal{S} \cup \{x_i\}$;
    $\mathcal{I} = \mathcal{I} \cup \{x_i, \mathsf{F}_k(x_i) \rightsquigarrow l_i\}$;
end;
$x_{q+1} \notin \mathcal{S} \leftarrow \mathsf{A}_1(\mathcal{S},\mathcal{I})$;
-
$b \leftarrow \mathsf{A}_2(x_{q+1}, \mathsf{F}_k(x_{q+1}), \mathcal{I})$;
```

```
$\mathbf{Exp}^{\mathrm{prf-ind-1}}_{\mathsf{P},\mathsf{A}}$
$\mathcal{S} = \varnothing$;
$\mathcal{I} = \varnothing$;
$k \xleftarrow{R} \{0,1\}^n$;
for $i = 1 : q$
    $x_i \leftarrow \mathsf{A}_1(\mathcal{S},\mathcal{I})$;
    $\mathcal{S} = \mathcal{S} \cup \{x_i\}$;
    $\mathcal{I} = \mathcal{I} \cup \{x_i, \mathsf{F}_k(x_i) \rightsquigarrow l_i\}$;
end;
$x_{q+1} \notin \mathcal{S} \leftarrow \mathsf{A}_1(\mathcal{S},\mathcal{I})$;
$y \xleftarrow{R} \{0,1\}^n$;
$b \leftarrow \mathsf{A}_2(x_{q+1}, y, \mathcal{I})$;
```

The ind-advantage of A *against* P *is defined as:*

$$\mathbf{Adv}^{\mathrm{prf-ind}}_{\mathsf{P},\mathsf{A}} = |\Pr[\mathbf{Exp}^{\mathrm{prf-ind-0}}_{\mathsf{P},\mathsf{A}} = 1] - \Pr[\mathbf{Exp}^{\mathrm{prf-ind-1}}_{\mathsf{P},\mathsf{A}} = 1]|$$

The implementation of a PRF is physically indistinguishable if the ind-advantage of any polynomial time adversary against this implementation is negligible.

In other words, the leakage is provided for the q first queries but not for the $q+$ 1'th one for which the indistinguishability game is played[2]. Also, the plaintexts can be selected adaptively by the adversary, but not the leakage function.

[2] In fact, for the PRF construction of Figure 10, we can prove a slightly stronger result. Namely, we only need that the leakage of the last PRF round (*i.e.* the last 2PRG invocation) of the last query x_{q+1} is not provided to the adversary.

We show in this section that if the leakage functions corresponding to the different rounds of the PRF in Figure 10 are seed-preserving, then the implementation of the PRF is physically indistinguishable (satisfying Definition 10). But we need a slight modification of the seed-preserving notion for this purpose. Just observe that Definition 7 has no guarantee for preserving the right-hand output of $\mathsf{2PRG}$. For example, a leakage function such that $\mathsf{L}^o(k_1, x_1) = x_1$ satisfies this definition but trivially leaks all PRF outputs rooted from x_1 to adversary $\mathsf{A}^{\mathsf{2PRG}}$. Hence, we introduce symmetric seed-preserving leakage functions in Definition 11, and then state the main result for our PRF construction in Theorem 2.

Definition 11. *Let $(\mathsf{L}^o, \mathsf{L}^i)$ be a pair of functions, $\mathsf{A}^{\mathsf{2PRG}}$ an algorithms representing the side-channel adversary with oracle access to 2PRG, n a fixed integer and $\mathsf{Pr}_{\mathsf{Guess}}(n, b)$ the following probability:* $\Pr[\mathsf{A}^{\mathsf{2PRG}}(\mathsf{L}^o(k_0, k_1), \mathsf{L}^i(k_b), k_{\bar{b}}) = k_b : k \leftarrow \{0,1\}^n; (k_0, k_1) := \mathsf{2PRG}(k)]$*. The pair $(\mathsf{L}^o, \mathsf{L}^i)$ is said to be ϵ-symmetric seed-preserving for parameter n and $\mathsf{A}^{\mathsf{2PRG}}$ if $\mathsf{Pr}_{\mathsf{Guess}}(n, b) \leq \epsilon$ for any $b \in \{0,1\}$. A pair of functions $(\mathsf{L}^o, \mathsf{L}^i)$ is said to be* symmetric seed-preserving *if, for every PPT $\mathsf{A}^{\mathsf{2PRG}}$, there is a negligible function $\epsilon(n)$ such that $(\mathsf{L}^o, \mathsf{L}^i)$ is $\epsilon(n)$-symmetric seed-preserving for every security parameter n and $\mathsf{A}^{\mathsf{2PRG}}$ running on input 1^n. A sequence of pairs of functions $(\mathsf{L}_1^o, \mathsf{L}_1^i), \ldots, (\mathsf{L}_l^o, \mathsf{L}_l^i)$ is said to be* uniformly symmetric seed-preserving *if, for every PPT $\mathsf{A}^{\mathsf{2PRG}}$, there is a negligible function $\epsilon(n)$ such that each pair of this sequence is $\epsilon(n)$-symmetric seed-preserving for every security parameter n and $\mathsf{A}^{\mathsf{2PRG}}$ running on input 1^n.*

In order to state our theorem, let us assume $\mathcal{X} = \mathcal{K} = \{0,1\}^n$, and denote the input and key of the PRF in Figure 10 as $x = b_1 \cdots b_n \in \{0,1\}^n$ and $k \in \{0,1\}^n$, respectively. Let us also define the following notations:

$$\begin{aligned}
G_0(k) \| G_1(k) &\overset{def}{=} \mathsf{2PRG}(k), \text{ with } |G_0(k)| = |G_1(k)|, \\
k_{b_1 \cdots b_i} &\overset{def}{=} G_{b_i}(\cdots G_{b_2}(G_{b_1}(k)) \cdots) \quad \text{for } 1 \leq i \leq n-1, \\
k_{b_1 \cdots b_n} &\overset{def}{=} G_0(G_{b_n}(k_{b_1 \cdots b_{n-1}})),
\end{aligned}$$

where subscripts $b_1 \cdots b_i$ identify the invocation path from root node k with one more invocation of G_0 appended to the last level, and thus the PRF output $\mathsf{F}_k(x)$ is simply given by k_x. We observe that this construction is essentially the GGM one [18], except that an extra round has been added in the end. This extra round has been introduced in order to make sure that the leakage occurring during the evaluation of F_k on one input does not trivially provide information on the output of F_k evaluated on an input that would only differ from the first one by the last bit. Note that, if $\mathsf{2PRG}$ is implemented using two AES, as depicted in Figure 7, then it is possible to evaluate only one of the two output halves, and this extra round is not needed anymore. As previously, we model the leakage function L with $n+1$ pairs of functions, $\overline{\mathsf{L}} := \langle(\mathsf{L}_1^i, \mathsf{L}_1^o), \ldots, (\mathsf{L}_{n+1}^i, \mathsf{L}_{n+1}^o)\rangle$, such that for any jth level PRG invocation $(k_{b_1 \cdots b_{j-1} 0}, k_{b_1 \cdots b_{j-1} 1}) := \mathsf{2PRG}(k_{b_1 \cdots b_{j-1}})$, the leakage is given by $(\mathsf{L}_j^i(k_{b_1 \cdots b_j}), \mathsf{L}_j^o(k_{b_1 \cdots b_j 0}, k_{b_1 \cdots b_j 1}))$. It directly yields:

Theorem 2. *The PRF construction given by* $\mathsf{F}_k(x) = G_0(G_{b_n}(\cdots G_{b_1}(k)\cdots))$ *is physically indistinguishable provided that the family of pairs of leakage functions* $\{(\bot, \mathsf{L}_1^i),\ (\mathsf{L}_1^o, \mathsf{L}_2^i),\ \ldots,\ (\mathsf{L}_n^o, \mathsf{L}_{n+1}^i)(\mathsf{L}_{n+1}^o, \bot)\}_{n\in\mathbb{N}}$*, is uniformly symmetric seed-preserving and can be evaluated in probabilistic polynomial time.*

The proof of Theorem 2 is given in Appendix C.

8 Remark on the impossibility of proving the leakage resilience for the forward secure PRG of Figure 6a in the standard model

Before to conclude this paper, we finally note that restricting the leakage function only will not be sufficient to prove the leakage resilience for the forward secure PRG of Figure 6a in the standard model. For example, say the 2PRG used in this construction can be written as:

$$\mathsf{G}(x_1, x_2, \ldots, x_n) = (x_1, \mathsf{G}^*(x_2, x_3, \ldots, x_n)) = (y_1, y_2, \ldots, y_{2n}),$$

with $\mathsf{G}^* : \{0,1\}^{n-1} \to \{0,1\}^{2n-1}$ a secure PRG. Using this 2PRG in the upper scheme of Figure 6 gives rise to a secure PRG in the black box setting. But if the first n bits of $\mathsf{G}(x_1, x_2, \ldots, x_n)$ are used as intermediate key and the second n bits as output, then a simple leakage function that only leaks x_1 will straightforwardly allow breaking the scheme. Importantly, this attack assumes a non adaptive leakage function of which the computational complexity is very low (it just leaks one bit per iteration). This counterexample shows that proving the leakage resilience of a PRG such as [5] in the standard model also requires stronger black box assumptions than traditionally considered for 2PRGs.

9 Open problems

This report implies two important scopes for further research. A first one is to better investigate side-channel resilient constructions of PRGs and PRFs (and their possible extension to PRPs). This requires to work on the minimum black box and physical assumptions that can be used to analyze the security of cryptographic devices in a formal setting. For example, important assumptions include:

1. The adaptivity of the leakage function. As discussed in Section 6.4, it is reasonable to remove this ability from the adversary's power and to reflect the possible adaptivity of the measurement setup in the λ-bit leakage bound.
2. "Only computations leak". As discussed in Section 5.3, this (and related) assumptions should ideally be avoided or reflected in the proofs quantitatively, in order to capture the behavior of advanced circuit technologies where static leakages and coupling effects are not negligible. Or, alternatively, one could investigate the exact type of dependencies that should be avoided (and translate them as requirement for hardware designers) to keep sound proofs.

3. The computational limits of the leakage function. In [31, 34], the leakage function is assumed to be a polynomial time computable function of its inputs. As discussed in Section 5.1, this incorporates attacks that exceed the power of actual adversaries. Hence, an interesting direction would be to further restrict this function. Promising candidates can be found in the early literature on side-channel attacks such as [2]. For example, considering linear functions of a device's internal configuration (or quadratic functions in order to capture possible coupling effects in the circuits) appears as a good starting point. At least, those type of functions should be captured by theoretical analysis, in view of their close connection to practice.

These physical assumptions then have to be combined with minimum black box requirements. We note that considering the black box security and the physical security with different assumptions may be an interesting alternative to demonstrate the leakage resilience while keeping proofs simple. For example, one can show the security of a construction in the standard model without leakage and then use a random oracle (that cannot be queried by the leakage function) to prove side-channel resilience, as in this report. Since the random oracle is then mainly used to capture the idea that "side-channel leakages do not leak about the future", it seems a reasonable abstraction in this context. Overall, this discussion shows that there are interesting tradeoffs to consider between the black box and physical requirements that one imposes to algorithms and implementations.

Second, we need to move from the abstract description of leakage resilient primitives towards their concrete implementations, in order to quantify the actual information leakages that they provide, in function of the adversary's abilities (*i.e.* data, time and memory complexity). It implies extending Table 2 in this paper and evaluating the security of more algorithms and devices against various attacks. This situation is in fact analogical to the classical cryptanalysis of block ciphers, where it is essential to determine the best known attacks against various ciphers. In a physical setting, the same question arises for every combination of algorithm and device - with the additional difficulty of exactly specifying the adversary's power, *e.g.* in terms of profiling and a priori knowledge of the underlying hardware. Concentrating the community's experimental efforts on a few implementations (*e.g.* by standardizing measurement boards [38]) would be very useful in this respect. Initiatives such as [11] could also be adapted for this purpose. We note that the large variability of the success rates in Table 2 suggests that keeping λ as small as possible in practical devices is certainly as challenging as properly exploiting small λ's in secure PRG (or other) constructions.

Eventually, physical security is a young topic and its proper formalization is still a scope for further research. Hence, it is important to question the validity of the models used to analyze the security of leaking devices first. Because of the physical origin of, *e.g.* side-channel attacks, it implies the need of experimental evaluations. And the implementation cost also has to be part of this analysis since overall (and in particular for small embedded devices), what matters to hardware designers is to obtain the best security at the lowest cost.

References

1. A. Akavia, S. Goldwasser, V. Vaikuntanathan, *Simultaneous Hardcore Bits and Cryptography against Memory Attacks*, in the proceedings of TCC 2009, LNCS, vol 5444, pp 474-495, San Francisco, CA, USA, March 2009.
2. M.L. Akkar, R. Bévan, P. Dischamp, D. Moyart, *Power Analysis, What is Now Possible...*, in the proceedings of ASIACRYPT 2001, LNCS, vol 1976, pp 489-502, Kyoto, Japan, December 2001.
3. R. Anderson, M. Kuhn, *Tamper Resistance - a Cautionary Note*, USENIX Workshop on Electronic Commerce, pp 1-11, Oakland, CA, USA, November 1996.
4. M. Bellare, A. Desai, E. Jokipii, P. Rogaway, *A Concrete Security Treatment of Symmetric Encryption*, in the proceedings of FOCS 1997, pp 394-403, Miami Beach, Florida, USA, October 1997.
5. M. Bellare, B. Yee, *Forward-Security in Private-Key Cryptography*, in the proceedings of CT-RSA 03, LNCS, vol 2612, pp 1-18, San Francisco, CA, USA, April 2003.
6. M. Blum, S. Micali, *How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits*, SIAM Journal of Computing, vol 13, num 4, pp 850-863, 1984.
7. C. Cachin, *Entropy Measures and Unconditional Security in Cryptography*, PhD Thesis, ETH Dissertation, num 12187, 1997.
8. S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power-Analysis Attacks*, in the proceedings of CRYPTO 1999, LNCS vol 1666, pp 398-412, Santa Barbara, California, USA, August 1999.
9. J. Daemen, V. Rijmen, *The Wide Trail Design Strategy*, in the proceedings of Cryptography and Coding, 8th IMA International Conference, LNCS, vol 2260, pp 222-238, Cirencester, UK, December 2001.
10. Y. Dodis, Y. Tauman Kalai, S. Lovett, *On Cryptography with Auxiliary Input*, in the proceedings of STOC 2009, pp 621-630, Bethesda, Maryland, USA, June 2009.
11. Télécom ParisTech, *The DPA Contest*, http://www.dpacontest.org/
12. S. Dziembowski, K. Pietrzak, *Leakage-Resilient Cryptography*, in the proceedings of FOCS 2008, pp 293-302, Washington, DC, USA, October 2008.
13. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, M.T. Manzuri Shalmani, *On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme*, in the proceedings of CRYPTO 2008, LNCS, vol 5157, pp 203-220, Santa Barbara, CA, USA, August 2008.
14. ECRYPT Network of Excellence in Cryptology, *The Side-Channel Cryptanalysis Lounge* , http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html.
15. ECRYPT Network of Excellence in Cryptology, *The eSTREAM Project*, http://www.ecrypt.eu.org/stream/, http://www.ecrypt.eu.org/stream/call/
16. S. Faust, L. Reyzin, E. Tromer, *Protecting Circuits from Computationally-Bounded Leakage*, Cryptology ePrint Archive, Report 2009/379.
17. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, *Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering*, in the proceedings of TCC 2004, LNCS, vol 2951, pp 258-277, Cambridge, Massachussets, USA, February 2004.
18. O. Goldreich, S. Goldwasser, S. Micali, *How to Construct Random Functions*, Journal of the ACM, vol 33, num 4, pp 792-807; 1986.
19. L. Goubin, J. Patarin, *DES and Differential Power Analysis*, in the proceedings of CHES 1999, LNCS, vol 1717, pp 158-172, Worcester, MA, USA, August 1999.
20. J.A Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, E.W. Felten, *Lest We Remember: Cold Boot Attacks*

on Encryption Keys, in the proceedings of the USENIX Security Symposium 2008, pp 45-60, San Jose, CA, USA, August 2008.

21. C. Hsiao, C. Lu, L. Reyzin, *Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility*, in the proceedings of EUROCRYPT 2007, LNCS, vol 4515, pp 169-186, Barcelona, Spain, May 2007.
22. Y. Ishai, A. Sahai, D. Wagner, *Private Circuits: Securing Hardware against Probing Attacks*, in the proceedings of Crypto 2003, LNCS, vol 2729, pp 463-481, Santa Barbara, California, USA, August 2003.
23. P. Junod, S. Vaudenay, *FOX : A New Family of Block Ciphers*, in the proceedings of SAC 2004, LNCS, vol 3357, pp 114-129, Waterloo, Canada, August 2004.
24. J. Katz, *Universally Composable Multi-party Computation Using Tamper-Proof Hardware*, in the proceedings of EUROCRYPT 2007, LNCS, vol 4515, pp 115-128, Barcelona, Spain, May 2007.
25. L.R. Knudsen, *Practically Secure Feistel Ciphers*, in the proceedings FSE 1993, LNCS, vol 809, pp 211-221, Cambridge, UK, December 1993.
26. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, in the proceedings of Crypto 1999, LNCS, vol 1666, pp 398-412, Santa-Barbara, CA, USA, August 1999.
27. P. Kocher, *Leak Resistant Cryptographic Indexed Key Update*, US Patent 6539092.
28. P. Kocher, *Design and Validation Strategies for Obtaining Assurance in Countermeasures to Power Analysis and Related Attacks*, in the proceedings of the NIST Physical Security Workshop, Honolulu, Hawai, September 2005.
29. B. Köpf, D. Basin, *An Information Theoretic Model for Adaptive Side-Channel Attacks*, in the proceedings of the ACM Conference on Computer and Communications Security 2007, pp 286-296, Alexandria, Virginia, USA, October 2007.
30. M. Luby, C. Rackoff, *How to Construct Pseudorandom Permutations from Pseudorandom Functions*, SIAM Journal of Computing, vol 17, num 2, pp 373-386, 1988.
31. S. Micali, L. Reyzin, *Physically Observable Cryptography*, in the proceedings of TCC 2004, LNCS, vol 2951, pp 278-296, Cambridge, MA, USA, February 2004.
32. T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant Software.*, in the proceedings of CHES 2000, LNCS, vol 2523, pp 238-251, Worcester, Massachussets, USA, August 2000.
33. C. Petit, F.-X. Standaert, O. Pereira, T.G. Malkin, M. Yung, *A Block Cipher based PRNG Secure Against Side-Channel Key Recovery*, in the proceedings of ASIACCS 2008, pp 56-65, Tokyo, Japan, March 2008.
34. K. Pietrzak, *A Leakage-Resilient Mode of Operation*, in the proceedings of Eurocrypt 2009, LNCS, vol 5479, pp 462-482, Cologne, Germany, April 2009.
35. K. Pietrzak, *Provable Security for Physical Cryptography*, invited talk, in the proceedings of WEWORC 2009, Graz, Austria, July 2009.
36. M. Renauld, F.-X. Standaert, *Algebraic Side-Channel Attacks*, Cryptology ePrint Archive, Report 2009/279, http://eprint.iacr.org/2009/279.
37. M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, *Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA*, in the proceedings of CHES 2009, LNCS, vol 5746, pp 97-111, Lausanne, Switzerland, September 2009.
38. RCIS (Research Center for Information Security), *SASEBO (Side-Channel Attack Standard Evaluation Boards)*, http://www.rcis.aist.go.jp/special/SASEBO/
39. W. Schindler, K. Lemke, C. Paar, *A Stochastic Model for Differential Side-Channel Cryptanalysis*, in the proceedings of CHES 2005, LNCS, vol 3659, pp 30-46, Edinburgh, Scotland, September 2005.
40. K. Schramm, T.J. Wollinger, C. Paar, *A New Class of Collision Attacks and Its Application to DES*, in the proceedings of FSE 2003, LNCS, vol 2887, pp 206-222, Lund, Sweden, February 2003.

41. N. Smart, D. Page, E. Oswald, *Randomised Representations*, in IET Information Security, vol 2, num 2, pp 19-27, June 2008.
42. F.-X. Standaert, T.G. Malkin, M. Yung, *A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks*, in the proceedings of Eurocrypt 2009, LNCS, vol 5479, pp 443-461, Cologne, Germany, April 2009, extended version available on the Cryptology ePrint Archive, Report 2006/139, http://eprint.iacr.org/2006/139.
43. F.-X. Standaert, E. Peeters, C. Archambeau, J.-J. Quisquater, *Towards Security Limits in Side-Channel Attacks*, in the proceedings of CHES 2006, LNCS, vol 4249, pp 30-45, Yokohama, Japan, October 2006, latest version available on the Cryptology ePrint Archive, Report 2007/222, http://eprint.iacr.org/2007/222.
44. F.-X. Standaert, C. Archambeau, *Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages*, in the proceedings of CHES 2008, LNCS, vol 5154, Washington DC, USA, August 2008.
45. F.-X. Standaert, B. Gierlichs, I. Verbauwhede, *Partition vs. Comparison Side-Channel Distingsuishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks*, in the proceedings of ICISC 2008, LNCS, vol 5461, pp 253-267, Seoul, Korea, December 2008.
46. F.-X. Standaert, P. Bulens, G. de Meulenaer, N. Veyrat-Charvillon, *Improving the Rules of the DPA Contest*, Cryptology ePrint Archive, Report 2006/139, http://eprint.iacr.org/2006/139.
47. K. Tiri, M. Akmal, I. Verbauwhede, *A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*, ESSCIRC 2003, Estoril, Portugal, September 2003.
48. S. Vaudenay, *Decorrelation: a Theory for Block Cipher Security*, in Journal of Cryptology, vol 16, num 4, pp 249-286, 2003.
49. A.C. Yao, *Theory and Applications of Trapdoor Functions (Extended Abstract)*, in the proceedings of FOCS 1982, pp 80-91, Chicago, Illinois, November 1982.

A Security metric

We consider a side-channel key recovery adversary $\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}$ with time complexity τ, memory complexity m and q queries to the target physical computer. His goal aim is to guess a key class s with non negligible probability. For this purpose and for each candidate s^*, he compares the actual observation of a leaking device $\mathbf{l}_q$ with some key dependent model for these leakages $\mathsf{M}(s^*,.)$. Let $\mathsf{T}(\mathbf{l}_q, \mathsf{M}(s^*,.))$ be the statistical test used in the comparison. We assume that the highest value of the statistic corresponds to the most likely key candidate. For each observation $\mathbf{l}_q$, we store the result of the statistical test T in a vector $\mathbf{g}_q = \mathsf{T}(\mathbf{l}_q, \mathsf{M}(s^*,.))$ containing the key candidates sorted according to their likelihood: $\mathbf{g}_q := [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ (*e.g.* $|\mathcal{S}|$=256 for key bytes). Then, for any side-channel attack exploiting a leakage vector $\mathbf{l}_q$ and giving rise to a result $\mathbf{g}_q$, we define the success function of order o against a key byte s as: $\mathsf{S}_k^o(\mathbf{g}_q)$=1 if $k \in [g_1, \ldots, g_o]$, else $\mathsf{S}_k^o(\mathbf{g}_q)$=0. It leads to the o^{th}-order success rate:

$$\mathbf{Succ}^{\text{sc-kr-}o,S}_{\mathsf{A}_{\mathsf{E}_K,\mathsf{L}}}(\tau, m, q) = \mathop{\mathbf{E}}_{k} \mathop{\mathbf{E}}_{\mathbf{l}_q} \mathsf{S}_k^o(\mathbf{g}_q) \tag{2}$$

Intuitively, a success rate of order 1 (*resp.* 2) relates to the probability that the correct key byte is sorted first (*resp.* among the two first ones) by the adversary.

B Proof of Theorem 1

Proof (Theorem 1). Let $\mathsf{A}^{\mathsf{2PRG}}(1^n)$ be an adversary who wins the $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\underline{\mathsf{L}}}(n)$ game with probability $\frac{1}{2} + \eta(n)$, and let p be a polynomial such that $p(n)$ is an upper bound on the number of $\mathsf{request}$ queries made by $\mathsf{A}^{\mathsf{2PRG}}(1^n)$. Let Query_l (resp. Query_a) be the event that $\mathsf{A}^{\mathsf{2PRG}}(1^n)$ makes a query to $\mathsf{2PRG}$ on the last key k_i (resp. any key) computed by the challenger before the test query is made.

We distinguish between the cases where the Query_l event happens or not: $\Pr[\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\underline{\mathsf{L}}}(n) = 1] \leq \Pr[\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\underline{\mathsf{L}}}(n) = 1 \wedge \neg\mathsf{Query}_l] + \Pr[\mathsf{Query}_l]$.

The probability $\Pr[\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\underline{\mathsf{L}}}(n) = 1 \wedge \neg\mathsf{Query}_l]$ is bounded by $\frac{1}{2} + \frac{p(n)^2}{2^n}$, which is the sum of the probability of a pure guess and an upper bound on the probability that a collision happens between PRG's last output and an output of a previous round.

We now show that $\Pr[\mathsf{Query}_l]$ is negligible. To this purpose we build an adversary $\mathsf{A}'^{\mathsf{2PRG}}$ as follows.:

Adversary $\mathsf{A}'^{\mathsf{2PRG}}$:

1. On input 1^n, start an instance of $\mathsf{A}^{\mathsf{2PRG}}$ with input 1^n, and record all interactions between $\mathsf{A}^{\mathsf{2PRG}}$ and the $\mathsf{2PRG}$ oracle.
2. Pick $j \leftarrow [0, p(n)]$ and $r_0 \leftarrow \{0,1\}^n$ uniformly at random, and set a counter i to 0.
3. Ask a challenger to pick $k_0 \in \{0,1\}^n$ uniformly at random, to compute $(k_1, x_1) := \mathsf{2PRG}(k_0)$ and to provide $(\mathsf{L}_j^o(k_1, x_1)$, x_1, $\mathsf{L}_{j+1}^i(x_1))$.

4. On each $\mathsf{request}$ query from $\mathsf{A}^{\mathsf{2PRG}}$, proceed as follows: increment the counter i, select $(r_i, x_i) \leftarrow (\{0,1\}^n)^2$ uniformly at random, and submit $\mathsf{L}_i^i(r_{i-1})$, y_i and $\mathsf{L}_i^o(r_i, y_i)$ to $\mathsf{A}^{\mathsf{2PRG}}$, unless $i = j$ in which case $\mathsf{L}_j^o(k_1, x_1)$, x_1 and $\mathsf{L}_{j+1}^i(x_1)$ are submitted instead.
5. On the test query from $\mathsf{A}^{\mathsf{2PRG}}$, pick $y_{i+1} \leftarrow \{0,1\}^n$ uniformly at random and submit that value to $\mathsf{A}^{\mathsf{2PRG}}$.
6. Let $\{z_1, \ldots, z_q\}$ be the set of requests made by $\mathsf{A}^{\mathsf{2PRG}}$ to $\mathsf{2PRG}$ until it halts. Output an element z selected uniformly at random into that set.

The strategy of adversary $\mathsf{A}'^{\mathsf{2PRG}}$ is based on the assumption that, in a normal run of the $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$ experiment, $\mathsf{A}^{\mathsf{2PRG}}$ would make a query on (at least) one of the keys involved in the experiment. So, $\mathsf{A}'^{\mathsf{2PRG}}$ makes a uniform guess on the index of the first key on which such a query is made; guessing the first queried key ensuring that that key will only be correlated to one thing: the corresponding leakages (and not any previous call on $\mathsf{2PRG}$). This guess will be correct with probability $\frac{1}{p(n)+1}$. Then, $\mathsf{A}'^{\mathsf{2PRG}}$ provides leakages to $\mathsf{A}^{\mathsf{2PRG}}$ computed from random values of its own choice, except for the j index, for which the leakages and PRG output are replaced by those obtained from a challenger for the seed-preserving property. $\mathsf{A}'^{\mathsf{2PRG}}$ also provides a random value y_{l+1} as final input to $\mathsf{A}^{\mathsf{2PRG}}$. If the guess on the index j is correct, all the inputs sent to $\mathsf{A}^{\mathsf{2PRG}}$ are distributed exactly as in the $\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n)$ experimen, as long as $\mathsf{A}^{\mathsf{2PRG}}$ does not make a query on the value k_1 computed by the challenger. Therefore, when $\mathsf{A}^{\mathsf{2PRG}}$ halts, $\mathsf{A}'^{\mathsf{2PRG}}$ can select one of the inputs of the q queries made by $\mathsf{A}^{\mathsf{2PRG}}$ and, if $\mathsf{A}^{\mathsf{2PRG}}$ made a query on k_1, that guess will be correct with probability $\frac{1}{q}$. So, eventually, we have that $\Pr[z = k_1 | \mathsf{Query}_a] = \frac{1}{q(p(n)+1)}$.

Now, we observe that $\Pr[z = k_1 | \mathsf{Query}_a] \leq \frac{\Pr[z=k_1]}{\Pr[\mathsf{Query}_a]}$, and that $\Pr[\mathsf{Query}_l] \leq \Pr[\mathsf{Query}_a]$, which implies that $\Pr[\mathsf{Query}_l] \leq q(p(n)+1)\Pr[z = k_1]$.

Eventually, we observe that $\mathsf{A}'^{\mathsf{2PRG}}$ runs in PPT: $\mathsf{A}^{\mathsf{2PRG}}$ runs in PPT, and the leakage functions can be evaluated in PPT too. Therefore, since the leakage function family $\overline{\mathsf{L}}$ is uniformly seed-preserving, there is a negligible function ϵ such that $\Pr[x = k_1] \leq \epsilon(n)$. As a result, $\Pr[\mathsf{Query}_l] \leq q(p(n)+1)\epsilon(n)$, which is negligible.

So, we have that $\Pr[\mathsf{Pred}_{\mathsf{A}^{\mathsf{2PRG}},\overline{\mathsf{L}}}(n) = 1] \leq \frac{1}{2} + \frac{p(n)^2}{2^n} + q(p(n)+1)\epsilon(n)$, as desired.

C Proof of Theorem 2

Proof (Theorem 2). In the context of this construction, $\mathbf{Exp}_{\mathsf{P},\mathsf{A}}^{\mathrm{prf-ind-0}}$ (as in Definition 10) is the output of $\mathsf{A}^{\mathsf{2PRG}}$ with the first q adaptive queries to $(\mathsf{F}, \overline{\mathsf{L}})$ and the $(q+1)^{th}$ query to F alone, which we visually write as $\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{F},\overline{\mathsf{L}})_{[1,q]},(\mathsf{F},\perp)_{[q+1]}\rangle}$ and analogously denote $\mathbf{Exp}_{\mathsf{P},\mathsf{A}}^{\mathrm{prf-ind-1}}$ as $\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{F},\overline{\mathsf{L}})_{[1,q]},(\mathsf{R},\perp)_{[q+1]}\rangle}$, where random function R is constructed using the same tree structure except that all tree nodes are fresh randomness (rather than being generating by invoking $\mathsf{2PRG}$ on their parent node as in the F), the output of R on input x is the leaf node reached by taking path x from the root. Then, by triangle inequality we have

$$\begin{aligned}
&|\Pr[\mathbf{Exp}^{\mathrm{prf-ind-0}}_{\mathsf{P,A}} = 1] - \Pr[\mathbf{Exp}^{\mathrm{prf-ind-1}}_{\mathsf{P,A}} = 1]| \\
&\leq |\Pr[\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{F},\overline{\mathsf{L}})_{[1,q]},(\mathsf{F},\perp)_{[q+1]}\rangle} = 1] - \Pr[\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{R},\overline{\mathsf{L}})_{[1,q]},(\mathsf{R},\perp)_{[q+1]}\rangle} = 1]| \\
&+ \ |\Pr[\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{R},\overline{\mathsf{L}})_{[1,q]},(\mathsf{R},\perp)_{[q+1]}\rangle} = 1] - \Pr[\mathsf{A}^{\mathsf{2PRG},\langle(\mathsf{F},\overline{\mathsf{L}})_{[1,q]},(\mathsf{R},\perp)_{[q+1]}\rangle} = 1]| \\
&\leq |\Pr[\mathsf{A}^{\mathsf{2PRG},(\mathsf{F},\overline{\mathsf{L}})_{[1,q+1]}} = 1] - \Pr[\mathsf{A}^{\mathsf{2PRG},(\mathsf{R},\overline{\mathsf{L}})_{[1,q+1]}} = 1]| \\
&\ \ +|\Pr[\mathsf{A}^{\mathsf{2PRG},(\mathsf{R},\overline{\mathsf{L}})_{[1,q]}} = 1] - \Pr[\mathsf{A}^{\mathsf{2PRG},(\mathsf{F},\overline{\mathsf{L}})_{[1,q]}} = 1]|
\end{aligned}$$

Therefore, we reduce the problem to showing the oracle indistinguishability between $(\mathsf{F},\overline{\mathsf{L}})$ and $(\mathsf{R},\overline{\mathsf{L}})$.

The main idea of the rest proof is that the ability to distinguish $(\mathsf{F},\overline{\mathsf{L}})$ and $(\mathsf{R},\overline{\mathsf{L}})$ using p queries to $\mathsf{2PRG}$ and q queries to the above oracle pair (by a hybrid argument and efficient simulation of the GGM tree [18]) implies an efficient algorithm (with only slightly more complexity than the distinguisher) to invert one of $2q$ independent instances of symmetric seed-preserving functions with probability of $\epsilon(n)/(2pn)$, or equivalently, solve one such instance with probability at least $\epsilon(n)/(4pqn)$, and thus a contradiction to the symmetric seed-preserving property in Definition 11.

Following Section 7, we use $k_{b_1\cdots b_j}$ to denote the string on the j^{th} level tree node by taking invocation path $b_1\cdots b_j$ from the 0^{th} level (root) node k. We consider hybrids $(\mathsf{H_0},\mathsf{L_0})$, $\cdots$ $(\mathsf{H_{n+1}},\mathsf{L_{n+1}})$, where each $(\mathsf{H_j},\mathsf{L_j})$ is constructed using the tree structure with all nodes of up to level j are randomly chosen, and the rest nodes (of higher levels) obtained by invoking $\mathsf{2PRG}$ on their parent node, and let $\mathsf{H_j}(x)$ be the leaf node reached by taking path x from the root, and define $\mathsf{L_j}$ by invoking $\overline{\mathsf{L}}$ on the corresponding nodes. It is not hard to see that $(\mathsf{F},\mathsf{L_F})$ is identical to $(\mathsf{H_0},\mathsf{L_0})$, and that $(\mathsf{R},\mathsf{L_R})$ is identical to $(\mathsf{H_{n+1}},\mathsf{L_{n+1}})$.

Suppose to the contrary that there exists $\mathsf{A}^{\mathsf{2PRG}}$ that distinguishes $(\mathsf{F},\mathsf{L_F})$ and $(\mathsf{R},\mathsf{L_R})$ with advantage $\epsilon(n)$ by making p queries to $\mathsf{2PRG}$ and q queries to $(\mathsf{F},\mathsf{L_F})/(\mathsf{R},\mathsf{L_R})$, then $\mathsf{A}^{\mathsf{2PRG}}$ can distinguish at least one neighboring hybrids $(\mathsf{H_j},\mathsf{L_j})$ and $(\mathsf{H_{j+1}},\mathsf{L_{j+1}})$ with advantage at least $\epsilon(n)/(n+1)$ by the same number of queries. We stress that $(\mathsf{H_j},\mathsf{L_j})$ and $(\mathsf{H_{j+1}},\mathsf{L_{j+1}})$ are identically distributed to anyone without access to $\mathsf{2PRG}$, or otherwise said, $\mathsf{A}^{\mathsf{2PRG}}$ must obtain the $\epsilon(n)/(n+1)$ advantage by recovering of one of the j^{th} level on-path (with respect to the q queries) nodes of $(\mathsf{H_j},\mathsf{L_j})$ and then run $\mathsf{2PRG}$ on it iteratively to verify whether the final output corresponds to the output of $\mathsf{H_j}$. Note that the ability to recover nodes on any other level does not gives $\mathsf{A}^{\mathsf{2PRG}}$ any advantage in distinguishing $(\mathsf{H_j},\mathsf{L_j})$ and $(\mathsf{H_{j+1}},\mathsf{L_{j+1}})$, which can be seen by a comparison between the two. We show in the following an inverting algorithm that uses the above distinguisher to solve one of $2q$ independent instances of pair of functions $(\mathsf{L}^o_{j-1},\mathsf{L}^i_j)$ with probability at least $\epsilon(n)/(2p(n+1))$.

The inverting algorithm $\mathsf{Inv}^{\mathsf{2PRG}}$ that runs $\mathsf{A}^{\mathsf{2PRG}}$ as subroutine, simulates $(\mathsf{H_j},\mathsf{L_j})$, and inverts $2q$ independent instances of $(\mathsf{L}^o_{j-1},\mathsf{L}^i_j)$ works as follows.

We recall (see Definition 11) that each problem instance is in the format of $(\mathsf{L}^o_{j-1}(k_0,k_1),\mathsf{L}^i_j(k_b),k_{\bar{b}})$, and the challenge is to recover k_b.

Adversary $\mathsf{Inv}^{\mathsf{2PRG}}$:

1. $\mathsf{Inv}^{\mathsf{2PRG}}$ takes as input 1^n, q independent instances of $(\mathsf{L}^o_{j-1},\mathsf{L}^i_j)$ with $b=0$, and another q independent instances with $b=1$. On startup, $\mathsf{Inv}^{\mathsf{2PRG}}$ starts an instance of $\mathsf{A}^{\mathsf{2PRG}}$ with input 1^n.
2. For each query $b_1\cdots b_n$ from $\mathsf{A}^{\mathsf{2PRG}}$ to $(\mathsf{H_j},\mathsf{L_j})$, $\mathsf{Inv}^{\mathsf{2PRG}}$ simulates $(\mathsf{H_j},\mathsf{L_j})$ as follows: for the first query (otherwise skip the common prefix that matches any of the previous ones), sample and record random strings k', $(k'_{b_1},k'_{\overline{b_1}})$, $(k'_{b_1b_2},k'_{b_1\overline{b_2}})$, $\ldots$, $(k'_{b_1\cdots b_{j-2}b_{j-1}},k'_{b_1\cdots b_{j-2}\overline{b_{j-1}}})$ of level up to $j-1$, and compute corresponding leakages using the leakage functions from $\overline{\mathsf{L}}$. At the j^{th} level, if $b_1\cdots b_{j-1}$ does not match the prefix of previous queries, toss a random bit $a_t \xleftarrow{R} \{0,1\}$ $(1\le t\le q)$ and install a new instance of $(\mathsf{L}^o_{j-1},\mathsf{L}^i_j)$ with $b=b_j\oplus a_t$ on it, *i.e.*, $(\mathsf{L}^o_{j-1}(k_0,k_1),\mathsf{L}^i_j(k_b),k_{\bar{b}})$ as the output-layer leakage of level j, input-layer leakage on input $k'_{b_1\cdots b_{j-1}b}$, and $k'_{b_1\cdots b_{j-1}\bar{b}}$ respectively. In case that $a_t=1$ (*i.e.* $b_j=\bar{b}$), the strings and leakages on level j and above can be simulated by invoking $\mathsf{2PRG}$ (on $k_{\bar{b}}$ and onwards) and computing the leakage functions, and in the other case that $a_t=0$, the simulation is identical except that $k'_{b_1\cdots b_j0}$ and $k'_{b_1\cdots b_j1}$ are sampled randomly instead of being obtained by invoking $\mathsf{2PRG}$ on $k_{\bar{b}}$.
3. $\mathsf{Inv}^{\mathsf{2PRG}}$ records all $\mathsf{A}^{\mathsf{2PRG}}$'s queries to $\mathsf{2PRG}$. When $\mathsf{A}^{\mathsf{2PRG}}$ halts, $\mathsf{Inv}^{\mathsf{2PRG}}$ randomly selects one of the recorded p queries and produces it as output, and then halts.

As discussed, $\mathsf{A}^{\mathsf{2PRG}}$ can invert one (say the t^{th}) of the $2q$ instances with chance at least $\epsilon(n)/(n+1)$ on condition that it is selected from the p candidates (with probability $1/p$) and that $a_t=0$ (with probability $1/2$), and thus the overall probability is $\epsilon(n)/(2p(n+1))$, as desired.

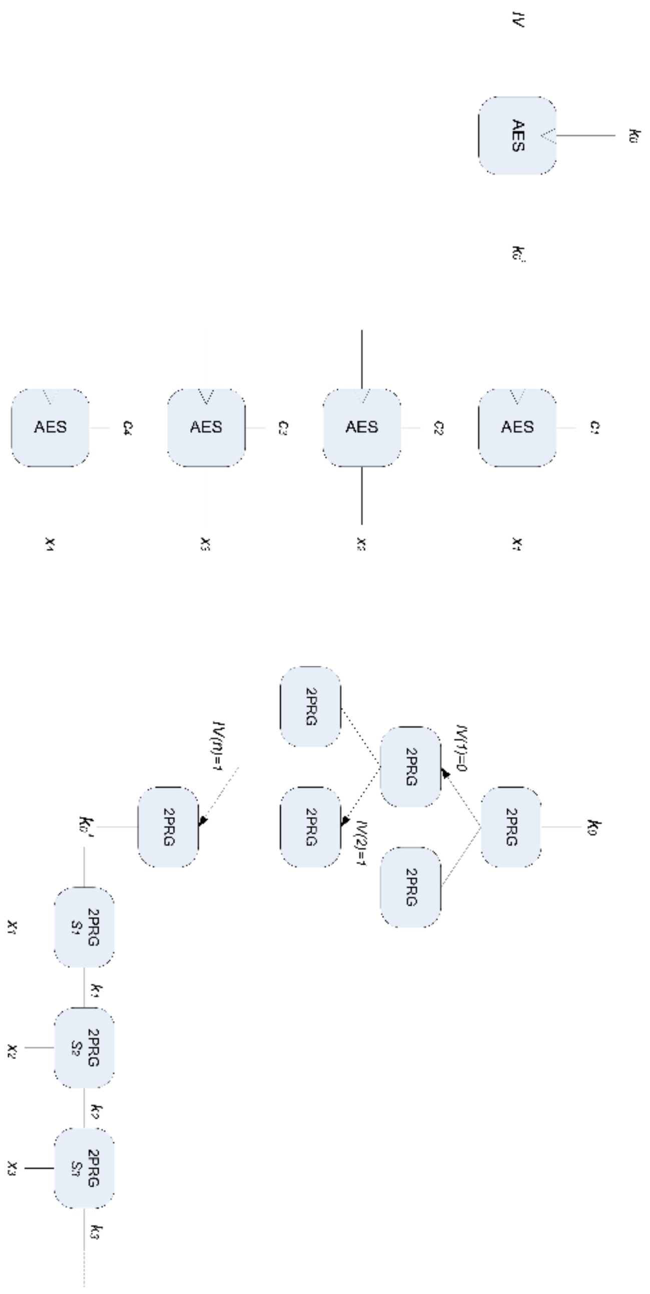


Fig. 11: Securely re-initialized PRGs. Up: 2^n-limited construction (*i.e.* best efficiency, worst security); Down: 2-limited contstruction (best security, worst efficiency).