Masking and Leakage-Resilient Primitives: One, the Other(s) or Both?

Sonial Belaïd[1,2], Vincent Grosso[3], François Xavier-Standaert[3]

[1] École Normale Supérieure, 45 rue dUlm, 75005 Paris.
[2] Thales Communications & Security, 4 Avenue des Louvresses, 92230 Gennevilliers.
[3] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.

Abstract. Securing cryptographic implementations against side-channel attacks is one of the most important challenges in modern cryptography. Many countermeasures have been introduced for this purpose, and analyzed in specialized security models. Formal solutions have also been proposed to extend the guarantees of provable security to physically observable devices. Masking and leakage-resilient cryptography are probably the most investigated and best understood representatives of these two approaches. Unfortunately, claims whether one, the other or their combination provides better security at lower cost remained vague so far. In this paper, we provide the first comprehensive treatment of this important problem. For this purpose, we analyze whether cryptographic implementations can be security-bounded, in the sense that the time complexity of the best side-channel attack is lower-bounded, independent of the number of measurements performed. Doing so, we first put forward a significant difference between stateful primitives such as leakage-resilient PRGs (that easily ensure bounded security), and stateless ones such as leakage-resilient PRFs (that hardly do). We then show that in practice, leakage-resilience alone provides the best security vs. performance tradeoff when bounded security is achievable, while masking alone is the solution of choice otherwise. That is, we highlight that one (x)or the other approach should be privileged, which contradicts the usual intuition that physical security is best obtained by combining countermeasures.

Besides, our experimental results underline that despite defined in exactly the same way, the bounded leakage requirement in leakage-resilient PRGs and PRFs imply significantly different challenges for hardware designers. Namely, such a bounded leakage is much harder to guarantee for stateless primitives (like PRFs) than for statefull ones (like PRGs). As a result, constructions of leakage-resilient PRGs and PRFs proven under the same bounded leakage assumption, and instantiated with the same AES implementation, may lead to different practical security levels.

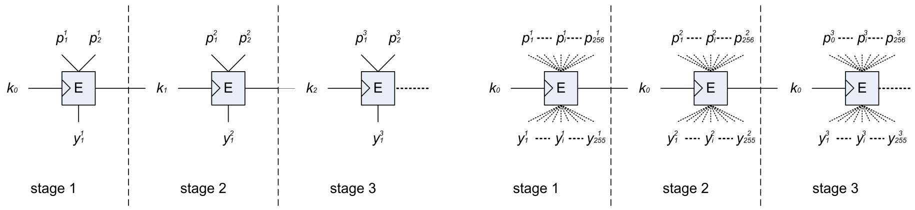
1 Introduction

Masking is a frequently considered solution to improve security against side-channel attacks [5, 19]. A large number of papers investigated its application to smart card implementations of the AES (e.g. [17, 37, 53, 55]). It essentially randomizes all the sensitive variables in a cryptographic device, by splitting them into d shares, and performs all computations on these shares afterwards. The resulting process is expected to improve physical security since if the masking scheme is carefully implemented (i.e. if the leakages of all the shares are independent), higher-order moments of the leakage distribution have to be estimated to reveal key-dependent information. It has been shown that the number of measurements needed to perform a successful DPA (Differential Power Analysis) increases exponentially with the number of shares (see, e.g. [44, 59]).

One limitation of masking is that (as most countermeasures against side-channel attacks [30]) it "only" reduces the amount of information leakage, at the cost of sometimes strong performance overheads [20]. Another line of work, next denoted as leakage-resilient cryptography, followed a complementary approach and tried to make the exploitation of this information more difficult (e.g. computationally). For this purpose, the main assumption is that the information leakage per iteration is limited in some sense. When applied in the context of symmetric cryptography, most instances of leakage-resilient constructions rely on re-keying strategies for this purpose, as first suggested by Kocher [27]. Examples of primitives include Pseudo-Random Generators (PRGs) [12, 15, 41, 57, 58, 64, 65] and Pseudo-Random Functions (PRFs) [1, 10, 15, 34, 58, 64].

The topic of leakage resilience has given rise to quite animated debates in the cryptographic community. Several assumptions have been proposed, and the quest for models that adequately capture physical reality is still ongoing (see [57] for a recent discussion). Yet, and independent of the relevance of the proofs obtained within these models, a more pragmatic problem is to find out the security levels of leakage-resilient constructions in front of standard side-channel adversaries (i.e. the same as the ones considered in security evaluations for masking). That is, are these primitives useful to help cryptographic designers to pass current certification procedures (e.g. EMVco [14] or Common Criteria [7])?

Unfortunately, claims in one or the other direction remained vague so far. The main reason is that, as hinted by Bernstein in a CHES 2012 rump session talk, substantiated answers require to consider both security and performances [3], i.e. two qualities that are generally hard to quantify. In this paper, we aim to contribute to this issue and provide tools allowing to determine the best way to reach a given security level in different (software and hardware) scenarios, within the limits of what empirical evaluations can provide. For this purpose, we will consider the AES-based PRG and PRF illustrated in Figures 1 and 2, respectively. For every key k_i, the PRG produces a key k_{i+1} and N-1 strings $y_1^i, y_2^i, \ldots, y_{N-1}^i$, both obtained by encrypting N public plaintexts p_j^i with k_i.



p_1^1 p_2^1
k_0
E
y_1^1
stage 1
p_1^2 p_2^2
k_1
E
y_1^2
stage 2
p_1^3 p_2^3
k_2
E
y_1^3
stage 3
p_1^1 ···· p_i^1 ···· p_{256}^1
k_0
E
y_1^1 ···· y_i^1 ···· y_{255}^1
stage 1
p_1^2 ···· p_i^2 ···· p_{256}^2
k_0
E
y_1^2 ···· y_i^2 ···· y_{255}^2
stage 2
p_0^3 ···· p_i^3 ···· p_{256}^3
k_0
E
y_1^3 ···· y_i^3 ···· y_{255}^3
stage 3

Fig. 1. Stateful leakage-resilient PRG with $N = 2$ (left) and $N = 256$ (right).

As for the PRF, we use the tree-based construction from Goldreich, Goldwasser and Micali [18], where each step incorporates $\log_2[N]$ input bits and generates $k_{i+1} = \mathsf{AES}_{k_i}(p_j^i)$. Following [34], the last stage is optionally completed by a whitening step, in order to limit the data complexity of attacks targeting the PRF output to one (e.g. when using large N values, typically).

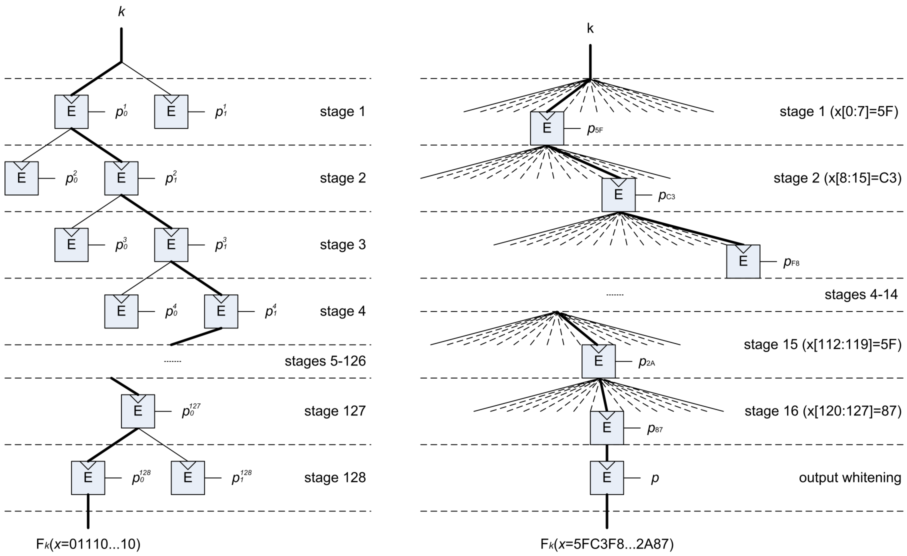


Fig. 2. Stateful leakage-resilient PRF with $N = 2$ (left) and $N = 256$ (right).

Quite naturally, there is a simple security versus efficiency tradeoff for both types of constructions. In the first (PRG) case, we produce a 128-bit output stream every $\frac{N}{N-1}$ AES encryptions. In the second (PRF) case, we produce a 128-bit output every $\frac{128}{\log(N)}$ AES encryptions (+1 if output whitening is used). The details of these primitives are not necessary for the understanding of this work. The only important feature in our discussions is that the PRG construction is stateful while the PRF one is stateless. As a result, the PRG limits the *number* *of measurements* that a side-channel adversary can perform with the same key, while the PRF limits his *data complexity* (i.e. the number of plaintexts that can be observed). In practice, it means that in this latter case, the same measurement can be repeated multiple times, e.g. in order to get rid of the physical noise through averaging. As already discussed by Medwed et al. in [34], Section 3, this may lead to significant difference in terms of security against DPA.

In order to compare and combine these two primitives with masking, we investigate whether they can lead to *security-bounded implementations*, i.e. implementations such that the time complexity of the best side-channel attack remains bounded independent of the number of measurements performed by the adversary. Doing so, we first show that the stateful leakage-resilient PRG in Figure 1 naturally leads to such implementations. By contrast, this guarantee is harder to reach with (stateless) leakage-resilient PRFs such as in Figure 2. The tweaked construction proposed in [34] (that takes advantage of hardware parallelism) is in fact the only security-bounded PRF we found in our experiments. Next, we put forward that better security at lower cost is obtained by using the leakage-resilient PRG alone (i.e. without masking), while masking alone is the most efficient solution for improving the security of stateless primitives whenever the implementations cannot be security-bounded. Therefore, our results underline that both masking and leakage-resilient primitives can be useful ingredients in the design of physically secure designs. But they also lead to the counterintuitive observation that sometimes (in fact, frequently), these solutions are better used separately, hence contradicting the usual intuition that security against side-channel attacks is best obtained via a combination of countermeasures.

Admittedly, these results are only obtained for a set of side-channel attacks that are representative of the state-of-the-art. Hence, positive observations such as made for the tweaked construction in [34] are not proven: they only indicate that the cryptanalysis of such schemes may be hard with current knowledge. In the same lines, the differences between leakage-resilient PRGs and PRFs do not contradict their proofs: they only indicate that the (crucial) assumption of bounded leakage can imply different challenges for hardware designers. Hence, instantiating these primitives with the same AES implementation can lead to different security levels (even if the same N value is used in both cases).

2 Methodology & limitations

The main goal of this paper is to provide sound techniques to evaluate how leakage-resilient PRGs/PRFs and masking combine. In this section, we provide a brief description of the methodology we will use for this purpose, and underline its limitations. The two main components, namely performance and security evaluations, are detailed in Sections 3 and 4, and then combined in Section 5. Our proposal essentially holds in five steps that we detail below.

1. *Fix the target security level.* In the following, we will take the AES Rijndael with 128-bit key as case study. Since a small security degradation due to

side-channel attacks is unavoidable, we will consider 120-bit, 100-bit and 80-bit target security levels for illustration. We do not go below 80-bit keys since it typically corresponds to current short-term security levels [9].

2. *Choose an implementation.* Given a cryptographic algorithm, this essentially corresponds to the selection of a technology and possibly a set of countermeasures to incorporate in the designs to evaluate. In the following, we will consider both software and hardware implementations for illustration, since they lead to significantly different performance and security levels. As for countermeasures, different types of masking schemes will be considered.
3. *Evaluate performances / extract a cost function.* Given an implementation, different metrics can be selected for this purpose (such as code size, RAM, or cycle count in software and area, frequency, throughput or power consumption in hardware). Both for software and hardware implementations, we will use combined functions, namely the "code size × cycle count" product and the "area / throughput" ratio. While our methodology would be perfectly applicable to other choices of metrics, we believe they are an interesting starting point to capture the efficiency of our different implementations. In particular for the hardware cases, such metrics are less dependent on the serial vs. parallel nature of the target architectures (see [26], Section 2).
4. *Evaluate security / extract the maximum number of measurements.* This central part of our analysis first requires to select the attacks from which we will evaluate security. In the following, we will consider the "standard DPA attacks" described in [31] for this purpose. Furthermore, we will investigate them in the profiled setting of template attacks (i.e. assuming that the adversary can build a precise model for the leakage function) [6]. This choice is motivated by the goal of approaching worst-case evaluations [56]. Based on these attacks, we will estimate the security graphs introduced in [61], i.e. compute the adversaries' success rates in function of their time complexity and number of measurements. From a given security level (e.g. 120-bit time complexity), we will finally extract the maximum number of measurements per key tolerated, as can be bounded by the PRG construction[1].
5. *Compute a global cost metric (possibly with an application constraint).* In case of security-bounded implementations, the previous security evaluation can be used to estimate how frequently one has to "re-key" within a leakage-resilient construction. From this estimate, we derive the average number of AES encryptions to execute per 128-bit output. By multiplying this number with the cost function of our performance evaluations, we obtain a global metric for the implementation of an AES-based design ensuring a given security level. In case of security-unbounded implementations, re-keying is not sufficient to maintain the target security level independent of the number of measurements performed by the adversary. So the cost functions have to be combined with an application constraint, stating the maximum number of measurements that can be tolerated to maintain this security level.

[1] Not the PRF which, as previously mentioned, can only bound the data complexity.

Quite naturally, such a methodology is limited in the same way as any performance and security evaluation. From the performance point-of-view, our investigations only apply to a representative subset of the (large) set of AES designs published in the literature. Because of place constraints, we first paid attention to state-of-the-art implementations and countermeasures, but applying our methodology to more examples is naturally feasible (and desirable). A very similar statement holds for security evaluations. Namely, we considered standard DPA attacks as a starting point, and because they typically correspond to the state-of-the-art in research and evaluation laboratories. Yet, cryptanalytic progresses can always appear[2]. Besides, countermeasures such as masking may rely on physical assumptions that are difficult to compare rigorously (since highly technology-dependent), as will be detailed next with the case of "glitches".

Note that these limitations are to a large extent inherent to the problem we tackle, and our results also correspond to the best we can hope in this respect. Hence, more than the practical conclusions that we draw in the following sections (that are of course important for current engineers willing to implement physically secure designs), it is the fact that we are able to compare the performance vs. security tradeoffs corresponding to the combination of leakage-resilient constructions with masking that is the most important contribution of this work. Indeed, these comparisons are dependent on the state-of-the-art implementations and attacks that are considered to be relevant for the selected algorithm.

3 Performance evaluations

In this section, we provide our performance evaluations for unprotected and masked AES designs. As previously mentioned, we will consider both software and hardware examples for this purpose. In this context, the main challenge is to find implementations that are (reasonably) comparable. This turned out to be relatively easy in the software case, for which we selected a couple of implementations in 8-bit microcontrollers, i.e. typical targets for side-channel analysis. By contrast, finding implementations in the same technology turns out to be more challenging in hardware: transistor sizes have evolved from (more than) 130μm to (less than) 65ηm over the last 15 years (i.e. the period over which most countermeasures against side-channel attacks have been proposed). Hence, published performance evaluations for side-channel protected designs are rarely comparable. Yet, we could find several designs in a recent FPGA technology, namely the Xilinx Virtex-5 devices (that are based on a 65ηm process).

The performances of the implementations we will analyze are summarized in Table 1. As previously mentioned, our software cost function is the frequently considered "code size × cycle count" metric, while we use the "area / throughput" ratio in the hardware (FPGA) case. As for the countermeasures evaluated, we first focused on the higher-order masking scheme proposed by Rivain and

[2] For example, the algebraic side-channel attacks introduced in [49, 50], while somewhat unrealistic for now, would certainly lead to different security levels.

Table 1. Performance of some illustrative AES implementations.

Software (8-bit) Implementations	*code size (bytes)*	*cycle count*	*cost function*	*physical assumptions*
Unprotected [13]	1659	4557	7.560	-
1-mask Boolean [53]	3153	$129 \cdot 10^3$	406.7	glitch-sensitive
1-mask polynomial [20, 45]	20 682	$1064 \cdot 10^3$	22 000	glitch-resistant
2-mask Boolean [53]	3845	$271 \cdot 10^3$	1042	glitch-sensitive
FPGA (Virtex-5) Implementations	*area (slices)*	*throughput (enc/sec)*	*cost function*	*physical assumptions*
Unprotected (128-bit) [48]	478	$\frac{245 \cdot 10^6}{11}$	21.46	-
1-mask Boolean (128-bit) [48]	1462	$\frac{100 \cdot 10^6}{11}$	160.8	glitch-sensitive
Threshold (8-bit) [36]	958	$\frac{170 \cdot 10^6}{266}$	1499	glitch-resistant

Prouff at CHES 2010, which can be considered as the state-of-the-art in software [53]. We then added the CHES 2011 polynomial masking scheme of Prouff and Roche [45] (and its implementation in [20]), as a typical example of "glitch-resistant" solution relying on secret sharing and multiparty computation (see the discussion in the next paragraph). A similar variety of countermeasures is proposed in hardware, where we also consider an efficient but glitch-sensitive implementation proposed in [48], and a threshold AES implementation that is one of the most promising solutions to deal with glitches in this case [36]. Note that this latter implementation is based on an 8-bit architecture (rather than a 128-bit one for the others). So although our cost function is aimed at making comparisons between different architectures more reflective of the algorithms' and countermeasures' performances, more serial implementations as this one generally pay a small overhead due to their more complex control logic.

Physical assumptions and glitches. As explicit in Table 1, countermeasures against side-channel attacks always rely on a number of physical assumptions. In the case of masking, a central one is that the leakage of the shares manipulated by the target implementation should be independent of each other [22]. Glitches, that are transient signals appearing during the computations in certain (e.g. CMOS) implementations, are a typical physical default that can cause this assumption to fail, as first put forward by Mangard et al. in [32]. There are two possible solutions to deal with such physical defaults: either by making explicit to cryptographic engineers that they have to prevent glitches at the physical level, or by designing countermeasures that can cope with glitches.

Interestingly, the first solution is one aspect where hardware and software implementations significantly differ. Namely, while it is usually possible to ensure independent leakages in masked software, by ensuring a sufficient time separation between the manipulation of the shares, it is extremely difficult to avoid glitches in hardware [33]. Yet, even in hardware it is generally expected that the "glitch signal" will be more difficult to exploit by adversaries, especially if designers pay attention to this issue [35]. In this context, the main question is to determine the amplitude of this signal: if sufficiently reduced in front of the measurement noise, it may turn out that a glitch-sensitive masked implementation leads to improved security levels (compared to an unprotected one). Since this amplitude is highly technology-dependent, we will use it as a parameter to analyze the security of our hardware implementations in the next sections. Yet, we recall that it is a safe practice to focus on glitch-resistant implementations when it comes to hardware. Besides, we note that glitches are not the only physical default that may cause the independent leakage assumption to be contradicted in practice [42, 51].

4 Security evaluations

We now move to the core of our analysis, namely the security evaluation of different implementations. For this purpose, we first need to discuss the type of security evaluation we will conduct, which can be viewed as a tradeoff between generality and informativeness. That is, one ideally wants to reach general conclusions in the sense that they are independent of the underlying device technology. A typical solution for this purpose is to evaluate the “security order” of a countermeasure, as defined by Coron et al. [8]. Informally, the security order corresponds to the largest moment in the leakage probability distributions that is key-independent (hence from which no information can be extracted). For example, an unprotected implementation can be attacked by computing mean values (i.e. first-order moments) [28]. By contrast, the hope of masking is to ensure that adversaries will have to estimate higher-order moments, which is expected to increase the data complexity required to extract information, as first shown by Chari et al. [5]. Evaluating the order is interesting because under the independent leakage assumption mentioned in the last section, it can be done based on the mathematical description of a countermeasure only. Of course, the informativeness of such an abstract evaluation is limited since (1) it indeed does not allow testing whether the independent leakage assumption is fulfilled, and (2) even if this assumption is fulfilled, there is no strict correspondance between the security order and the security level of an implementation (e.g. measured with a probability of success corresponding to some bounded complexities). This is because already for masking (i.e. the countermeasure that aims at increasing the security order), and even if independent leakages are observed in practice, the actual complexity of a side-channel attack highly depends on the amount of noise in the measurements. And of course, there are also countermeasures that simply do not aim at increasing the security order, e.g. shuffling [21].

One appealing way to mitigate the second issue is to perform so-called “simulated attacks”. This essentially requires to model the leakage corresponding to different sensitive operations in an idealized implementation. For example, a usual approximation is to consider that all the intermediate values during a cryptographic computation (such as the S-boxes inputs and outputs for a block cipher) leak the sum of their Hamming weight and a Gaussian distributed noise [30]. It is then possible to accurately estimate the evaluation metrics proposed in [56] (i.e. mutual information, success rate, guessing entropy) from these mathematically generated leakages. Furthermore, one can use the noise variance as a security parameter and analyze its impact on the time and data complexity of successful attacks. Quite naturally, such an alternative still does not solve the first issue (i.e. the independent leakage assumption), for which the only possibility is to evaluate the real measurements of an actual implementation, in a given technology. This latter solution is admittedly the most informative, but also the least general, and is quite intensive for comparison purposes (since it requires to have access to source codes, target devices and measurement setups for all the designs to evaluate). Interestingly, it has been shown that simulated attacks can be quite close to real ones in the context of standard DPA and masking [59]. So since our goal is to show that there exist realistic scenarios where leakage-resilient PRGs/PRFs and masking are useful ingredients to reach a given security level at the lowest cost, we will use this type of evaluations in the following.

Note finally that performing simulated attacks could not be replaced by computing explicit formulae for the success rate such as, e.g. [16, 52]. Indeed, these formulae only predict subkey (typically key bytes) recoveries while we consider security graphs for full 128-bit master keys. Beside, they are only applicable to unprotected devices so far, and hardly capture masked implementations and the effect of key-dependent algorithmic noise as we will consider next.

4.1 Evaluation setups

We will consider two types of setups in our evaluations: one for software, one for hardware. As illustrated in Figure 3 in the case of a Boolean-masked S-box implementation with two shares, the main difference is that the software performs all the operations sequentially, while the hardware performs them in parallel. We

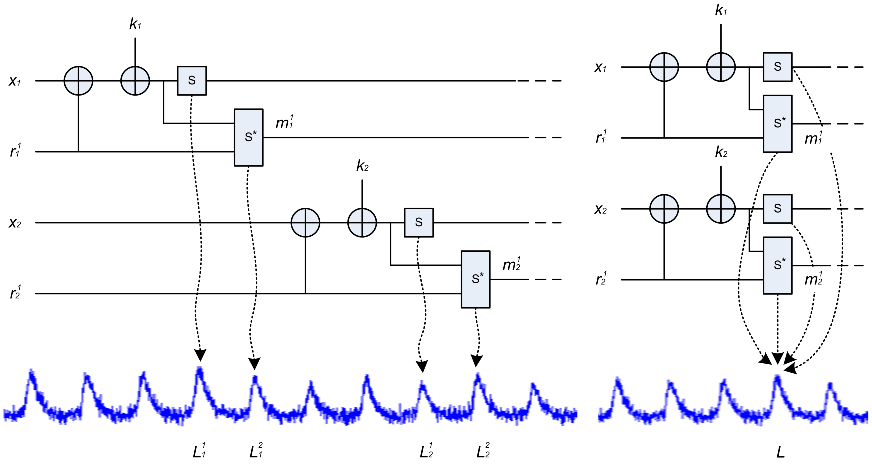


Fig. 3. Simulated leaking implementations. Left: software, right: hardware.

will further assume that the leakage of parallel operations is summed [40]. As previously mentioned, we will illustrate our analyses with a Hamming weight leakage function. Additionally, we will consider a noise variance of 10, corresponding to a Signal-to-Noise Ratio of 0.2 (as defined in [29])[3]. This is a typical value, both for software implementations [11] and FPGA measurement boards [25].

Let us denote the AES S-box as S, a byte of plaintext and key as x_i and k_i (respectively), the random shares used in masking as r_i^j (before the S-box) and m_i^j (after the S-box), the Hamming weight function as HW, the bitwise XOR as $\oplus$, the field multiplication used in polynomial masking as $\otimes$, and Gaussian-distributed noise random variables N_i^j. From these notations, we can specify the list of all our target implementations as summarized in Table 2.

A couple of observations are worth being underlined as we now discuss.

First, and as already mentioned, the main difference between software and hardware implementations is the number of exploitable leakage samples: there is a single such sample per plaintext in hardware while there are $16 \times (N_m + 1)$ ones in software (with N_m the number of masks). Next, we only considered glitches in hardware (since it is generally possible to ensure independent leakage in software, by ensuring a sufficient time separation between the manipulation of the shares). We assumed that "first-order glitches" can appear in our Boolean-masked FPGA implementation, and modeled the impact of the mask as an additive binomial noise in this case. We further assumed that the amplitude of this first-order signal was reduced according to a factor f. This factor corresponds to the parameter used to quantify the amplitude of the glitches mentioned in the previous section. Note that this modeling is sound because the complexity of a first-order DPA only depends on the value of its SNR (which is equivalent to correlation and information theoretic metrics in this case, as proven in [31]). So even leakage functions deviating from the Hamming weight abstraction would lead to similar trends. Since the threshold implementation in [36] guarantees the absence of first-order glitches, we only analyzed the possibility of second-order glitches for this one, and modeled them in the same way as just described (i.e. by considering the second mask M_i^2 as an additive binomial noise, and reducing the amplitude of the second-order signal by a factor f). Third, the chosen-plaintext construction of [34] is only applicable in hardware. Furthermore, we only evaluated its impact for the unprotected implementation, and the 1-mask Boolean one with glitches. As will become clear in the next section, this is because the data complexity bound to 256 (that is the maximum tolerated by design in this case) is only relevant when successful side-channel attacks occur for such small complexities (which was only observed for implementations with first-order signal).

For convenience, we denoted each implementation in our experiments with three letters. The first one corresponds to the type of scenario considered, i.e. with Known (K) or carefully Chosen (C) plaintexts. The second one indicates

[3] The SNR corresponds to ratio between the signal variance (that equals 2 for the Hamming weights of uniformly distributed 8-bit values) and the noise variance.

Table 2. List of our target implementations.

Ref.	**8-bit software**	*leakage function* ($\forall\ i \in [1;16]$)	*glitches*	*construction*
KSU	Unprotected [13]	$L_i = \mathsf{HW}(\mathsf{S}(x_i \oplus k_i)) + N_i$	no	KP
KSB_1	1-mask Boolean [53]	$L_i^1 = \mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i) + N_i^1,\ L_i^2 = \mathsf{HW}(M_i) + N_i^2$	no	KP
KSP_1	1-mask polynomial [20, 45]	$L_i^1 = \mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i \otimes P_0) + N_i^1,$ $L_i^2 = \mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i \otimes P_1) + N_i^2$	no	KP
KSB_2	2-mask Boolean [53]	$L_i^1 = \mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i^1 \oplus M_i^2) + N_i^1,$ $L_i^2 = \mathsf{HW}(M_i^1) + N_i^2,\ L_i^3 = \mathsf{HW}(M_i^2) + N_i^3$	no	KP
Ref.	**Virtex-5 FPGA**	*leakage function* (sum over $1 \leq i \leq 16$)	*glitches*	*construction*
KHU	Unprotected (128-bit) [48]	$L = \sum[\mathsf{HW}(\mathsf{S}(x_i \oplus k_i))] + N$	no	KP
CHU	Unprotected (128-bit) [48]	$L = \sum[\mathsf{HW}(\mathsf{S}(x \oplus k_i))] + N$	no	CP
KHB_1	1-mask Boolean (128-bit) [48]	$L = \sum[\mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i) + \mathsf{HW}(M_i)] + N$	no	KP
KHB_1^*	1-mask Boolean (128-bit) [48]	$L = \sum[\frac{\mathsf{HW}(\mathsf{S}(x_i \oplus k_i))}{f} + \mathsf{HW}(M_i)] + N$	1st-order	KP
CHB_1^*	1-mask Boolean (128-bit) [48]	$L = \sum[\frac{\mathsf{HW}(\mathsf{S}(x \oplus k_i))}{f} + \mathsf{HW}(M_i)] + N$	1st-order	CP
KHT_2	Threshold (8-bit) [36]	$L = \sum[\mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i^1 \oplus M_i^2) + \mathsf{HW}(M_i^1) + \mathsf{HW}(M_i^2)] + N$	no	KP
KHT_2^*	Threshold (8-bit) [36]	$L = \sum[\frac{\mathsf{HW}(\mathsf{S}(x_i \oplus k_i) \oplus M_i^1) + \mathsf{HW}(M_i^1)}{f} + \mathsf{HW}(M_i^2)] + N$	2nd-order	KP

whether we are in a Software (S) or Hardware (H) case study. The third one corresponds to the type of countermeasure selected, i.e. Unprotected (U), 1- or 2-mask Boolean (B_1, B_2), 1-mask Polynomial (P_1) and 2-mask threshold (T_2). The additional star signals finally reflect the presence of (first-order or second-order) glitches. For example, KHB_1^* is an AES design protected with a 1-mask Boolean scheme, implemented in an imperfect hardware leading to first-order glitches, and analyzed in the context of known (uniform) plaintexts.

4.2 Template attacks and security graphs

Given the leakage functions defined in Table 2, a template attack first requires to build a leakage model. In the following, and for each byte of the AES master key, we will consider Gaussian templates for unprotected implementations, and Gaussian mixtures for masked implementations. Let us denote the probability density function of a Gaussian distribution taken on input z, with mean μ (resp. mean vector $\boldsymbol{\mu}$) and variance σ^2 (resp. covariance matrix Σ) as $\mathcal{N}(z|\mu, \sigma^2)$ (resp. $\mathcal{N}(z|\boldsymbol{\mu}, \Sigma)$). This notation directly leads to models of the form:

$$\Pr_{\text{model}}[k_i|l_i, x_i] = \frac{\mathcal{N}(l_i|\mu_{k_i,x_i}, \sigma_{k_i,x_i})}{\sum_{k_i^*\in\mathcal{K}} \mathcal{N}(l_i|\mu_{k_i^*,x_i}, \sigma_{k_i^*,x_i}))}, \tag{1}$$

$$\Pr_{\text{model}}[k_i|l, x_i] = \frac{\mathcal{N}(l|\mu_{k_i,x_i}, \sigma_{k_i,x_i})}{\sum_{k_i^*\in\mathcal{K}} \mathcal{N}(l|\mu_{k_i^*,x_i}, \sigma_{k_i^*,x_i}))}, \tag{2}$$

for (software and hardware) unprotected implementations and:

$$\Pr_{\text{model}}[k_i|l_i^1, l_i^2, x_i] = \frac{\sum_{m_i^*\in M} \mathcal{N}(l_i^1, l_i^2|\boldsymbol{\mu}_{k_i,x_i,m_i^*}, \Sigma_{k_i,x_i,m_i^*})}{\sum_{k_i^*\in\mathcal{K}} \sum_{m_i^*\in M} \mathcal{N}(l_i^1, l_i^2|\boldsymbol{\mu}_{k_i^*,x_i,m_i^*}, \Sigma_{k_i^*,x_i,m_i^*})}, \tag{3}$$

$$\Pr_{\text{model}}[k_i|l, x_i] = \frac{\sum_{m_i^*\in M} \mathcal{N}(l|\mu_{k_i,x_i,m_i^*}, \sigma_{k_i,x_i,m_i^*})}{\sum_{k_i^*\in\mathcal{K}} \sum_{m_i^*\in M} \mathcal{N}(l|\mu_{k_i^*,x_i,m_i^*}, \sigma_{k_i^*,x_i,m_i^*})}, \tag{4}$$

for (software and hardware) masked implementations with two shares. The formula naturally extends to more shares, by just adding more sums over the masks. Note that in these models, all the noise (including the algorithmic one in hardware implementations) is captured by the Gaussian distribution[4]. Given these models, the template adversary will accumulate information on the key bytes k_i, by computing products of probabilities corresponding to multiple plaintexts. Doing so and for each key byte, he will produce lists of 256 probabilities corresponding each possible candidate $\tilde{k_i}$, defined as follows:

$$p_{\tilde{k_i}} = \prod_{j=1}^{q} \Pr_{\text{model}}[\tilde{k_i}|\mathbf{L}^{(j)}, x_i^{(j)}], \tag{5}$$

[4] While algorithmic noise is generated with a binomial distribution in our experiments (as mentioned in the previous subsections), it is closely approximated by a normal one, since combined with enough (simulated) physical noise that is Gaussian.

with the leakage vector $\mathbf{L}^{(j)}$ respectively corresponding to $l_i^{(j)}$ (resp. $l^{(j)}$) in the context of Equ. 1 (resp. Equ. 2) and $l_i^{1,(j)}, l_i^{2,(j)}$ (resp. $l^{(j)}$) in the context of Equ. 3 (resp. Equ. 4) The number of measurements is given by q in Equ. 5. Next and for each target implementation, we will repeat 100 experiments. And for each value of q in these experiments, use a rank estimation algorithm to evaluate the time complexity needed to recover the full AES master key [61]. Eventually, we will build "security graphs" where the attack probability of success is provided in function of a time complexity and a number of measurements.

Iterative DPA against constructions with carefully chosen plaintexts. Note that while standard DPA attacks are adequate to analyze the security of unprotected and masked implementations in a known-plaintext scenario, their divide-and-conquer strategy hardly applies to the PRF in [34], with carefully-chosen plaintexts leading to key-dependent algorithmic noise. This is because the (maximum 256) constants c_j used in this proposal are such that all 16 bytes are always identical. Hence, a standard DPA will provide a single list of probabilities, containing information about the 16 AES key bytes at once. In this case, we additionally considered the iterative DPA described in this previous reference, which essentially works by successively removing the algorithmic noise generated by the best-rated key bytes. While such an attack can only work under the assumption that the adversary has an very precise leakage model in hand, we use it as a representative of worst-case attack against such a construction.

4.3 Experimental results

For illustration, the security graph of the AES implementation KHB_1 is given in Figure 4, where we additionally provide the maximum number of measurements tolerated to maintain security levels corresponding to 2^{120}, 2^{100} and 2^{80} time complexity. All the implementations in Table 2 have been similarly evaluated and the result of these experiments are in Appendix A, Figures 8 to 13. Note that in the aforementioned case of iterative DPA (Appendix A, Figure 14), the adversary recovers the AES key bytes but still has to find their position within the AES state, which (roughly) corresponds to $16! \approx 2^{44}$ possibilities [2].

5 Security vs. performance tradeoffs

We now combine the results in the previous sections to answer our main question. Namely, what is the best way to exploit masking and/or leakage-resilient primitives to resist standard DPA in hardware and software implementations?

5.1 Leakage-resilient PRGs

Let M be the maximum number of measurements tolerated to maintain a given security level for one of the implementations in section 4. The re-keying in leakage-resilient PRGs is such that it is exactly this number M that is limited by design (i.e. the value N in Figure 1 bounds M for the adversary), hence

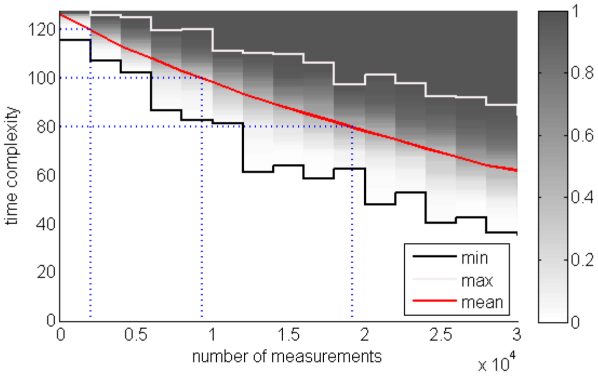


Fig. 4. Security graph for the Boolean-masked hardware implementation KHB_1.

directly leading to security-bounded implementations. The global cost metric we use in this case can be written as $\frac{M}{M-1} \times$ *cost function*, where the first factor corresponds to the average number of AES encryptions that are used to produce each 128-bit output string, and the second one is the cost function of Table 1.

A comparison of different leakage-resilient PRG implementations in software (i.e. based on different unprotected and protected AES implementations) is given in Figure 5 for 80-bit and 120-bit security levels (the results for 100-bit security are in Appendix A, Figure 15, left). The main observation in this context is that the straightforward implementation of the PRG with an unprotected AES design is the most efficient solution. This is mainly because moving from the smallest M value (i.e. $M = 2$, as imposed by the 120-bit security level in the unprotected case - see Figure 8-left) to large ones (e.g. $M > 1000$ for masked implementations) can only lead to a gain factor of 2 for the global cost metric, which is not justified in view of the performance overheads due to the masking. For a similar reason (i.e. the limited interest of increasing M), the global cost metric is essentially independent of the target security level in the figure. In other words, there is little interest in decreasing this security level since it leads to poor performance improvements. The hardware implementations in Appendix A, Figures 15-right and 16 lead to essentially similar intuitions, as also witnessed by the limited impact of decreasing the amplitude of the glitch signal with the f factor (see the KHB_1^* and KHT_2^* implementations for which $f = 10$ in the latter figures).

5.2 Leakage-resilient PRFs

Security-unbounded implementations. Let us now consider (stateless) leakage-resilient PRFs. As already mentioned, those constructions only bound the adversary's data complexity. The main observation in this case is that if random plaintexts are considered, such implementations can only be security-unbounded (with the slight cautionary note that we give below). This fact can be easily explained when the PRF is instantiated with an unprotected software implementation of the AES. What happens then is that the adversary can repeat

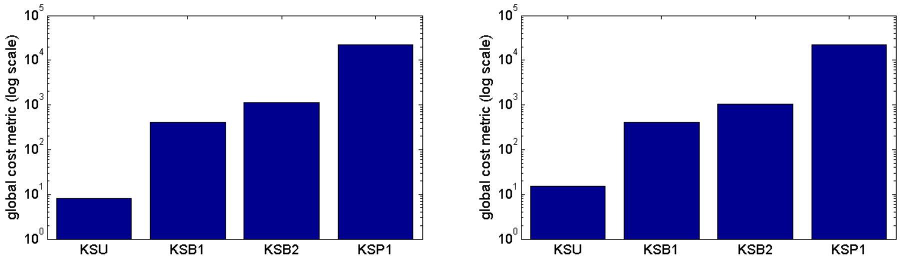


Fig. 5. LR-PRGs in software. 80-bit (left) and 120-bit (right) security.

his measurements to get rid of the physical noise, and consequently move from the security graph of Appendix A, Figure 8-left to the one of Appendix A, Figure 13-right. Such a "repeating" attack is exactly the one already mentioned in [34] to argue that bounded data complexity is not enough to bound (computational) security. In fact, it similarly applies to masked implementations. The only difference is that the adversary will not average his measurements, but rather combine them as in Equation 5. This is because given a leakage function, e.g. the Hamming weight one that leads to 9 distinguishable events, the distribution of the measurements in a masked implementation will lead to the same number of distinguishable events: the only difference is that more sampling will be necessary to distinguish them (see the appendices in [60] for a plot of these distributions). So if the number of measurements is not bounded, attacks with low time complexities as in Appendix A, Figure 13 right will always exist.

One important consequence is that using the PRF construction in this context is essentially useless for all the AES implementations we consider in this paper. The only way to maintain a target security level for such stateless primitives is to limit the number of measurements by putting a constraint on the lifetime of the system. And this lifetime will be selected according to the maximum number of measurements tolerated that can be extracted from our security graphs, which now highly depends on the countermeasure selected. In other words, we can only evaluate the cost function and the security level attained independently in this case, as illustrated in Figure 6 for our software instances (the 100-bit security level is again given in Appendix A, Figure 17-left). Here, we naturally come back to the standard result that Boolean (resp. polynomial) masking increases security at the cost of performance overheads that are roughly quadratic (resp. cubic) in the number of shares. Note that the security level of the 1-mask polynomial scheme is higher than the 2-mask Boolean one for the noise variance we consider, which is consistent with the previous work of Roche and Prouff [54]. Similar conclusions are obtained with hardware implementations (Appendix A, Figure 17-right and Appendix A, Figure 18), for which the impact of glitches is now clearly visible. For example, a factor $f = 10$ essentially multiplies the number of measurements by f for the Boolean masking with first-order glitches, and f^2 for the threshold implementation with second-order glitches.

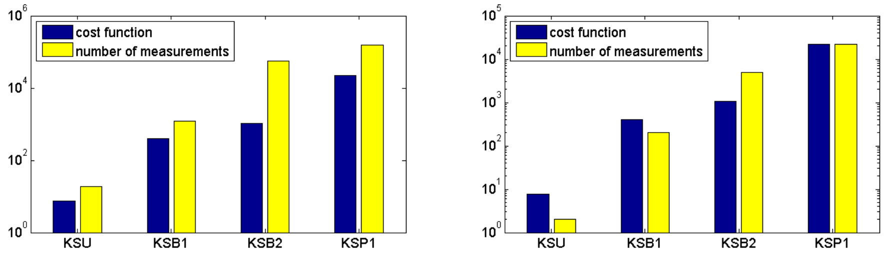


Fig. 6. LR-PRFs in software with KP. 80-bit (left) and 120-bit (right) security.

Cautionary note. The statement that stateless leakage-resilient PRFs can only be security unbounded if known plaintexts are considered essentially relates to the fact that repeated measurements allow removing the effect of the noise and the masks in a leaking implementation. Yet, this claim should be slightly mitigated in the case of algorithmic noise in hardware implementations. Indeed, this part of the noise can only be averaged up to the data complexity bound that is imposed by the PRF design. Taking the example of our hardware implementations where all 16 S-boxes are manipulated in parallel, the SNR corresponding to algorithmic noise can be computed as the ratio between the variance of a uniformly distributed 8-bit values's Hamming weight (i.e. 2) and the variance of 15 such values (i.e. 30). Averaging this noise over M plaintexts will lead to SNRs of $\frac{1}{15/M}$, which is already larger than 17 if $M = 256$ (i.e. a noise level for which the security graph will be extremely close to the worst case one of Appendix A, Figure 13-right). So although there is a "gray area" where a leakage-resilient PRF implemented in hardware can be (weakly) security-bounded, these contexts are of quite limited interest because the will imply bounds on the data complexity that are below 256, i.e. they anyway lead to less efficient solutions than the tweaked construction that we investigate in the next subsection.

Security-bounded implementations. As just discussed, stateless primitives hardly lead to security bounded implementations if physical and algorithmic noise can be averaged - which is straightforwardly feasible in a known plaintext scenario. The tweaked construction in [34] aims at avoiding such a weakness by preventing the averaging of the algorithmic noise, thanks to the combined effect of hardware parallelism and carefully chosen plaintexts leading to key-dependencies in this noise. Since only the physical noise can be averaged in this case, the bounded data complexity that the leakage-resilient PRF guarantees consequently leads to security-bounded implementations again. This is illustrated both by the standard DPAs (such as in Appendix A, Figures 10-right and 12-left) and the iterative attacks (such as in Appendix A, Figure 13) that can be performed against this PRF[5]. As in Section 5.1, we extracted the maximum

[5] As previously mentioned, there is an additional $16! \approx 2^{44}$ time complexity implied in the iterative DPA attacks, corresponding to the enumeration of a permutation over the 16 AES key bytes that is necessary to test each key candidate.

data complexity D from these graphs, and produced as global cost metric:

$$\left\lceil \frac{128}{\lfloor \log_2(D) \rfloor} \right\rceil \times \mathit{cost\ function},$$

where the first factor corresponds to the (rounded) average number of AES encryptions needed to produce a 128-bit output, and the second one is the cost function of Table 1. A comparison of our different leakage-resilient PRFs instantiated with a hardware implementation of the AES and chosen plaintexts is given in Figure 7. Here again, we observe that the most efficient solution is to consider an unprotected design. Interestingly, we also observe that for the unprotected AES, the iterative attack is the worst case for the 80-bit security level (where it forces the re-keying after 97 plaintexts vs. 256 for the standard DPA), while the standard DPA is the worst-case for the 120-bit security level (where it forces the re-keying after 10 plaintexts vs. 37 for the iterative attack). This nicely fits the intuition that iterative attacks become more powerful as the data complexity increases, i.e. when the additional time complexity corresponding to the enumeration of a permutation over 16 bytes becomes small compared to the time complexity required to recover the 16 AES key bytes (unordered).

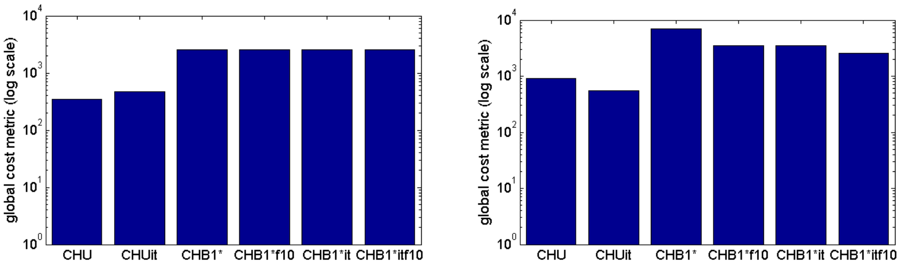


Fig. 7. LR-PRFs in hardware with CP. 80-bit (left) and 120-bit (right) security.

6 Conclusion

The results in this work essentially show that masking and leakage-resilient constructions hardly combine constructively. For (stateful) PRGs, our experiments indicate that both for software and hardware implementations, a leakage-resilient design instantiated with an unprotected AES is the most efficient solution to reach any given security level. For stateless PRFs, they rather show that a bounded data complexity guarantee is (mostly) ineffective in bounding the (computational) complexity of the best attacks. So implementing masking and limiting the lifetime of the cryptographic implementation is the best solution in this case. Nevertheless, the chosen-plaintext tweak proposed in [34] is an interesting exception to this conclusion, as it leads to security-bounded hardware implementations for stateless primitives that are particularly interesting from an application point-of-view, e.g. for re-synchronization, challenge-response protocols, … Beyond the further analysis of such constructions, their extension to software implementations is an interesting scope for further research. In this respect, the combination of a chosen-plaintext leakage-resilient PRF with the shuffling countermeasure in [62] seems promising, as it could "emulate" the key-dependent algorithmic noise ensuring security bounds in hardware.

Acknowledgements. F.-X. Standaert is an associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Work funded in parts by the European Commission through the ERC project 280141 (CRASH) and the European ISEC action grant HOME/2010/ISEC/AG/INT-011 B-CCENTRE project.

References

1. Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Bertoni and Coron [4], pages 471–488.
2. Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jorn-Marc Schmidt, Francois-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient PRFs: Cipher design principles and analysis. Cryptology ePrint Archive, Report 2013/305, 2013. `http://eprint.iacr.org/`.
3. Daniel J. Bernstein. Implementing "practical leakage-resilient cryptography". CHES 2012 Rump Session Talk, Leuven, Belgium, September 2012.
4. Guido Bertoni and Jean-Sébastien Coron, editors. *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*. Springer, 2013.
5. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [63], pages 398–412.
6. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
7. Common Criteria Portal. http://www.commoncriteriaportal.org/.
8. Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In Paillier and Verbauwhede [38], pages 28–44.
9. Cryptographic Key Length Recommendation. http://www.keylength.com/.
10. Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
11. François Durvaux, Mathieu Renauld, François-Xavier Standaert, Loïc van Oldeneel tot Oldenzeel, and Nicolas Veyrat-Charvillon. Efficient removal of random delays from embedded software implementations using hidden markov models. In Stefan Mangard, editor, *CARDIS*, volume 7771 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2012.
12. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

13. Thomas Eisenbarth, Zheng Gong, Tim Güneysu, Stefan Heyse, Sebastiaan Indesteege, Stéphanie Kerckhof, François Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, François-Xavier Standaert, and Loïc van Oldeneel tot Oldenzeel. Compact implementation and performance evaluation of block ciphers in ATtiny devices. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT*, volume 7374 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2012.
14. Europay Mastercard Visa. http://www.emvco.com/.
15. Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Prouff and Schaumont [46], pages 213–232.
16. Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for dpa with novel algorithmic confusion analysis. In Prouff and Schaumont [46], pages 233–250.
17. Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In Preneel and Takagi [43], pages 240–255.
18. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *FOCS*, pages 464–479. IEEE Computer Society, 1984.
19. Louis Goubin and Jacques Patarin. Des and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
20. Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: How large is the gap for the AES? In Bertoni and Coron [4], pages 400–416.
21. Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.
22. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
23. Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013.
24. Antoine Joux, editor. *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*. Springer, 2009.
25. Toshihiro Katashita, Akashi Satoh, Katsuya Kikuchi, Hiroshi Nakagawa, and Masahiro Aoyagi. Evaluation of DPA characteristics of sasebo for board level simulation. In Sorin Huss and Werner Schindler, editors, proceedings of *COSADE* 2010, 4 pages, Darmstadt, Germany, February 2011.
26. Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In Prouff and Schaumont [46], pages 390–407.
27. Paul C. Kocher. Leak resistant cryptographic indexed key update. US Patent 6539092.
28. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [63], pages 388–397.

29. Stefan Mangard. Hardware countermeasures against DPA ? a statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
30. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
31. Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.
32. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked cmos gates. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.
33. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Rao and Sunar [47], pages 157–171.
34. Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient PRFs. In Prouff and Schaumont [46], pages 193–212.
35. Amir Moradi and Oliver Mischke. Glitch-free implementation of masking in modern FPGAs. In *HOST*, pages 89–95. IEEE, 2012.
36. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In Paterson [39], pages 69–88.
37. Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the AES S-Box. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
38. Pascal Paillier and Ingrid Verbauwhede, editors. *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*. Springer, 2007.
39. Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.
40. Eric Peeters, François-Xavier Standaert, Nicolas Donckers, and Jean-Jacques Quisquater. Improved higher-order side-channel attacks with FPGA experiments. In Rao and Sunar [47], pages 309–323.
41. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Joux [24], pages 462–482.
42. Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. Evaluation of the masked logic style mdpl on a prototype chip. In Paillier and Verbauwhede [38], pages 81–94.
43. Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.
44. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Johansson and Nguyen [23], pages 142–159.
45. Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Preneel and Takagi [43], pages 63–78.

46. Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012.
47. Josyula R. Rao and Berk Sunar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
48. Francesco Regazzoni, Wang Yi, and François-Xavier Standaert. FPGA implementations of the AES masked against power analysis attacks. In Sorin Huss and Werner Schindler, editors, proceedings of *COSADE* 2011, pp 56-66, Darmstadt, Germany, February 2011.
49. Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.
50. Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic side-channel attacks on the AES: Why time also matters in DPA. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.
51. Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Paterson [39], pages 109–128.
52. Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 165–183. Springer, 2008.
53. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
54. Thomas Roche and Emmanuel Prouff. Higher-order glitches free implementation of the AES using secure multi-party computation protocols extended version . Cryptology ePrint Archive, Report 2011/413, 2011. `http://eprint.iacr.org/`.
55. Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
56. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Joux [24], pages 443–461.
57. François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.
58. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer, 2010.
59. François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

60. Francois-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. Cryptology ePrint Archive, Report 2010/180, 2010. `http://eprint.iacr.org/`.
61. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Johansson and Nguyen [23], pages 126–141.
62. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
63. Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.
64. Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.
65. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 141–151. ACM, 2010.

A Additional figures

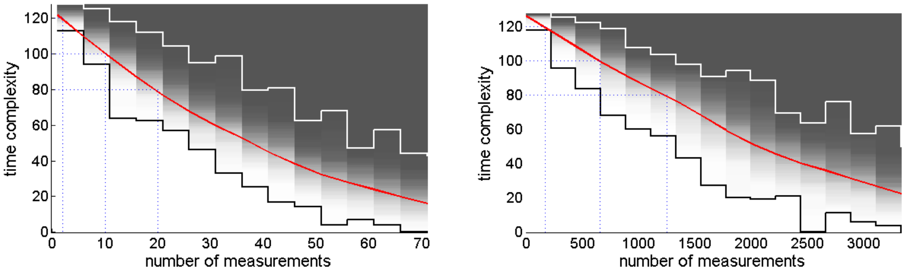


Fig. 8. DPA-based security graphs for KSU (left) and KSB$_1$ (right).

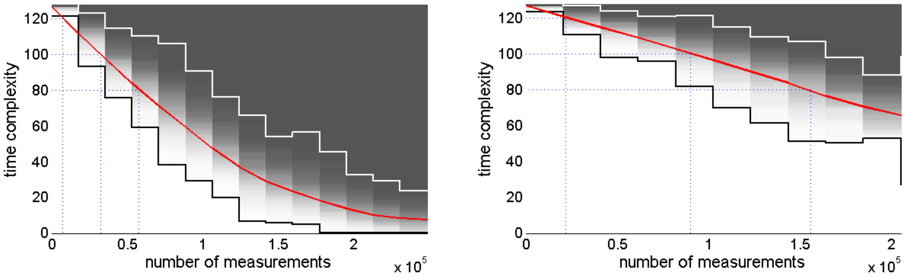


Fig. 9. DPA-based security graphs for KSB$_2$ (left) and KSP$_1$ (right).

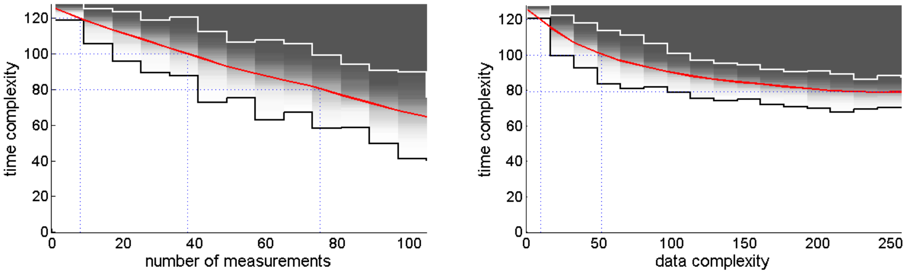


Fig. 10. DPA-based security graphs for KHU (left) and CHU (right).

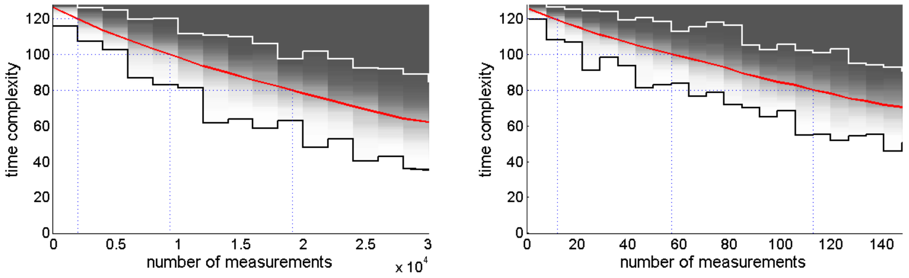


Fig. 11. DPA-based security graphs for KHB_1 (left) and $\mathrm{KHB}_1^*/f = 1$ (right).

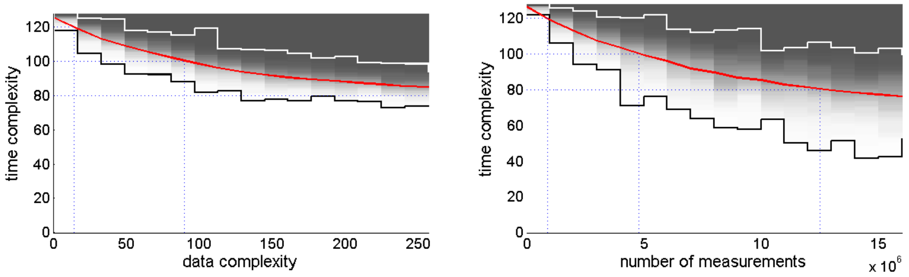


Fig. 12. DPA-based security graphs for $\mathrm{CHB}_1^*/f{=}1$ (left) and KHT_2 (right).

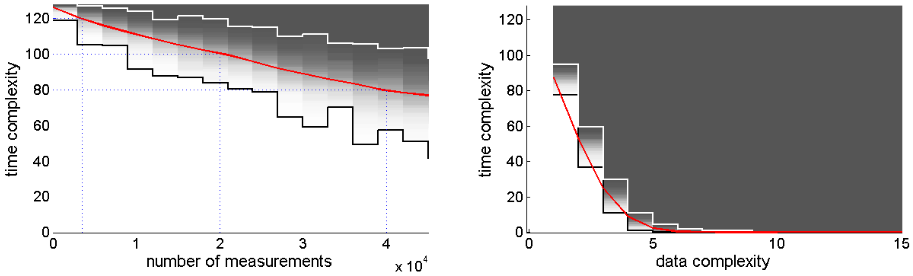


Fig. 13. DPA-based security graphs for $\mathrm{KHT}_2^*/f{=}1$ (left) and repeating attacks (right).

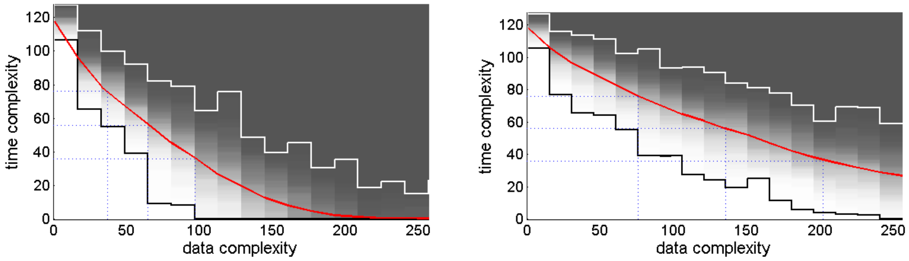


Fig. 14. Iterative DPA-based security graphs for CHU (left) and $\mathrm{CHB}_1^*/f = 1$ (right).

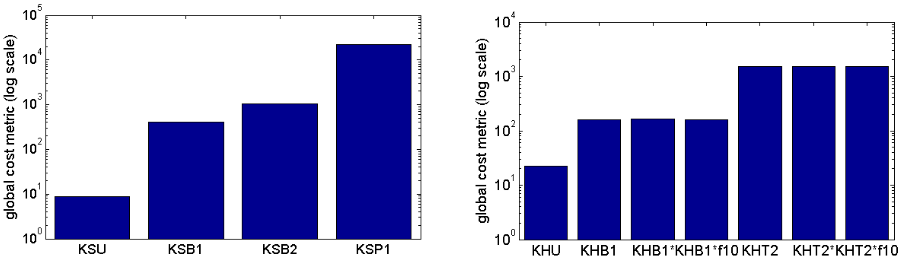


Fig. 15. LR-PRGs in software (left) and hardware (right). 100-bit security.

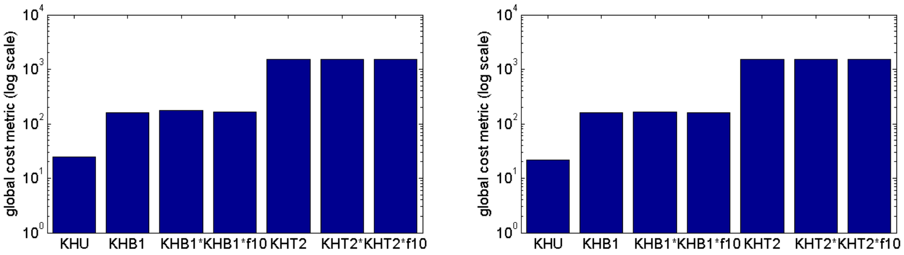


Fig. 16. LR-PRGs in hardware. 80-bit (left) and 120-bit (right) security.

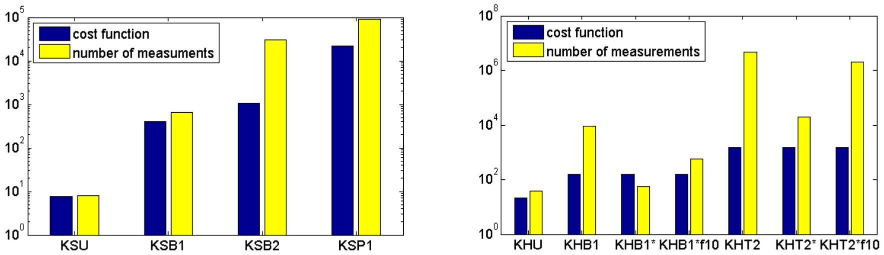


Fig. 17. LR-PRFs in software (left) and hardware (right) with KP. 100-bit security.

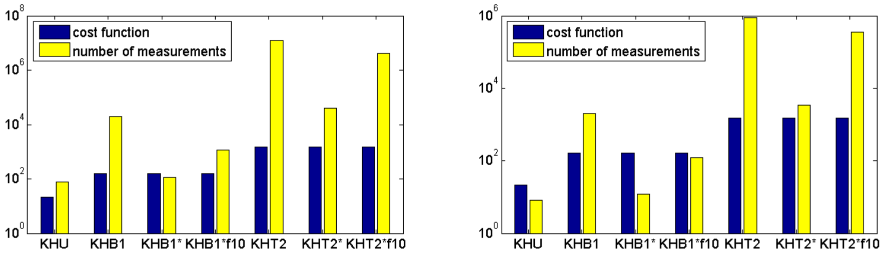


Fig. 18. LR-PRFs in hardware with KP. 80-bit (left) and 120-bit (right) security.