Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs

Marcel Medwed[1⋆], François-Xavier Standaert[1⋆⋆], Antoine Joux[2].

[1] UCL Crypto Group, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.
[2] DGA and Université de Versailles Saint-Quentin-en-Yvelines, Laboratoire PRISM.
45 avenue des États-Unis, F-78035 Versailles Cedex, France.

Abstract. Leakage-resilient constructions have attracted significant attention over the last couple of years. In practice, pseudorandom functions are among the most important such primitives, because they are stateless and do not require a secure initialization as, e.g. stream ciphers. However, their deployment in actual applications is still limited by security and efficiency concerns. This paper contributes to solve these issues in two directions. On the one hand, we highlight that the condition of bounded data complexity, that is guaranteed by previous leakage-resilient constructions, may not be enough to obtain practical security. We show experimentally that, if implemented in an 8-bit microcontroller, such constructions can actually be broken. On the other hand, we present tweaks for tree-based leakage-resilient PRFs that improve their efficiency and their security, by taking advantage of parallel implementations. Our security analyses are based on worst-case attacks in a noise-free setting and suggest that under reasonable assumptions, the side-channel resistance of our construction grows super-exponentially with a security parameter that corresponds to the degree of parallelism of the implementation. In addition, it exhibits that standard DPA attacks are not the most relevant tool for evaluating such leakage-resilient constructions and may lead to overestimated security. As a consequence, we investigate more sophisticated tools based on lattice reduction, which turn out to be powerful in the physical cryptanalysis of these primitives. Eventually, we put forward that the AES is not perfectly suited for integration in a leakage-resilient design. This observation raises interesting challenges for developing block ciphers with better properties regarding leakage-resilience.

1 Introduction

Physical attacks, in which adversaries take advantage of the peculiarities of the devices on which cryptographic operations are running, are an important concern for modern security applications. They typically include side-channel attacks (where the adversary monitors the leakage due to the cryptographic computations [26]), fault attacks (where the adversary tries to force a device to

⋆ Postdoctoral researcher funded by the 7th framework European project TAMPRES.
⋆⋆ Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC project 280141 (acronym CRASH).

perform erroneous computations [6]), tampering attacks (where the adversary probes a few wires of the implementation [4]) and memory attacks (where the adversary directly monitors parts of the memory [18]). As a result, a variety of hardware-level countermeasures have been proposed, in order to reduce the amount of information an implementation may provide to adversaries. Over the last few years, such hardware-level countermeasures have been complemented by significant efforts to extend the formal guarantees of provable security from cryptographic algorithms towards cryptographic implementations. For this purpose, various models have been introduced, trying to capture physical reality in abstract terms, with the goal of allowing meaningful reasoning about physical security. Examples of models to formalize side-channel attacks include Micali and Reyzin's physically observable cryptography [29] and Dziembowski and Pietrzak's leakage-resilient cryptography [14]. Examples of models to formalize fault attacks, tampering attacks and memory attacks can also be found, e.g. in [3, 16, 21, 22]. Eventually, more general abstractions, such as the auxiliary input model [11], or the bounded retrieval model [10, 13], have been introduced for similar purposes. Quite naturally, these different models raise numerous questions about their relevance to practice, e.g. regarding the correspondence (or lack thereof) between the basic assumptions used in proofs and what can actually be guaranteed by hardware designers. In general, there remain many open problems to answer in order to specify a fully satisfying model (see, e.g. [38, 39]).

Nevertheless, and somewhat independent of the practical relevance of the models used to formalize physical security issues, it may very well be that (small variations of) ideas proposed in these theoretical works actually provide significantly enhanced security against large categories of "practical" side-channel attacks (such as surveyed in [28]), when analyzed in more restricted frameworks such as [41]. In this paper, we follow this direction and focus on the security of symmetric cryptographic primitives such as block and stream ciphers. This focus is naturally motivated by the fact that such low-cost algorithms are among the most frequently considered targets, e.g. for power and EM analysis.

Related work. The very idea to prevent side-channel attacks at the protocol level relying on key updates refers to Kocher [25]. Following, Pseudo-Random Number Generators (PRNGs) allowing security against side-channel attacks have first been proposed and analyzed in a specialized setting [33] (with noisy Hamming weight or identity leakage functions). Leakage-resilient stream ciphers based on an "alternating structure", with a proof of security in the standard model have then been described in [14, 34]. Similar constructions without alternating structure, but with a proof in a model relying on a random-oracle assumption can be found in [42, 43]. Finally, an attempt to prove the security of a PRG without alternating structure and in the standard model, assuming non-adaptive leakage functions, was also suggested in [43], and was later shown to require large amounts of public randomness for the proof to hold in [15]. It remains an open question to determine if the exact construction proposed in [43], using only two alternating public values, can be proven secure or attacked in a practical setting.

While theoretically interesting (and efficiently implemented), these stream ciphers and PRNGs all suffer from the limitation that they require a secure initialization mechanism. As already clear in [33], this problem of initialization is an important issue for deploying leakage-resilient constructions in real-world devices, as it typically implies much larger performance overheads. Roughly speaking, and taking AES-based designs as an example, all previous leakage-resilient PRNGs can output one 128-bit block every two AES executions. By contrast, the initialization mechanism proposed at ASIACCS 2008 [33] requires 128 AES executions per block (if the best security against side-channel attacks is privileged). Following solutions did not allow any improvement in this respect.

Interestingly, this requirement of a secure initialization process for PRNGs can actually be translated into the need of a leakage-resilient PseudoRandom Function (PRF). As a consequence, it was first observed in [42] that a tree-based construction such as the one of Goldreich, Goldwasser and Micali (GGM) [17] inherently brings improved resistance against side-channel attacks. Again taking an AES-based example, these PRFs allow ensuring that every intermediate key in the tree is only used twice. The construction in [42] was proven secure against side-channel attacks under a random-oracle assumption, together with the observation that leakage-resilience for stateless PRFs requires to limit the leakage function to be non-adaptive. Next, Dodis and Pietrzak constructed a leakage-resilient PRF and proved its security in the standard model (with non-adaptive leakages as well) [12], by applying a GGM-like construction to the stream cipher with alternating structure from [34]. How to replace the alternating structure by alternating public randomness is additionally discussed in [15].

Finally, the construction of PseudoRandom Permutations (PRPs) was first discussed at CRYPTO 2010 [12]. In this paper, Dodis and Pietrzak describe efficient attacks (with non-adaptive leakage functions) against Feistel ciphers constructed from leakage-resilient PRFs. It is shown in [15] that more positive results can be obtained in a known-plaintext (rather than chosen-plaintext) adversarial scenario. However, as for the PRF constructions listed above, the practicality of these PRP constructions is strongly limited by performance overheads.

In this work, we tackle two important questions related to leakage-resilient PRFs.

First, we study their implementation in leaking devices and the security level that they provide against standard side-channel attacks. Doing so, we put forward the difference between a side-channel attack with bounded data complexity and a side-channel attack with bounded number of measurements. As previous constructions in [12, 15, 42] all guarantee a bounded data complexity, but do not prevent unbounded number of measurements for the PRF executed on the same inputs, we exhibit attacks in low-cost (8-bit) microcontrollers taking advantage of these capabilities. We use these experiments to argue about the need of parallelism for leakage-resilient PRF implementations in general.

Second, we study how to exploit parallelism in the implementations of PRFs, in order to significantly improve their efficiency. For this purpose, we take advantage of a careful selection of the public values used in an AES-based construction. By enforcing that these public values are such that all the bytes corresponding

to the first-round S-boxes are identical, we succeed in significantly reducing the success rate of a Differential Power Analysis (DPA) against our implementations. Doing so, we also observe that DPA is not the most suitable tool for cryptanalyzing such leakage-resilient designs, and describe advanced attacks exploiting lattice reduction, that allow us to better evaluate worst-case security levels. This analysis puts forward the need to enumerate a permutation of the AES bytes, which offers an interesting security parameter (as the number of such permutations grows as a factorial function). Furthermore, this trick allows us to reduce the 128 AES iterations per block, required in the execution of previous PRFs constructions, down to 17 (and even less if the PRF is used for encryption in counter mode). This results in an overhead factor that is much more in line with the ones of other countermeasures against side-channel attacks. While our proposal goes against the requirement of independent public randomness in [15], it is backed up by a practical security analysis, which again raises the question whether this requirement is motivated by the physics or by proof artifacts. Besides, our proposal could be integrated into the PRF of CRYPTO 2010, in which the alternative structure removes the need of public randomness in the proofs.

We finally remark that the techniques analyzed in this paper raise interesting challenges for the design of new block ciphers allowing efficient implementations when inserted in leakage-resilient PRFs, or for the direct design of ad hoc constructions. As suggested by the AES-128 instance that we study in Section 6, the Rijndael algorithm may not be the best suited block cipher for this purpose, and we suggest a few directions that could lead to improved solutions.

2 The leakage resilient PRF constructions

In the first sections of this paper, we will base our discussions on the GGM construction depicted in the left part of Figure 1. Let $\mathsf{F}_k(x)$ denote the PRF indexed by k and evaluated on x. Further, let the building blocks $\mathsf{E}_{k^i}(p_j^i)$ denote the application of a block cipher E to a plaintext p_j^i under a key k^i (the figure takes the example of E = AES-128 with $1 \leq i \leq 128$ and $0 \leq j \leq 1$). Let also $x(i)$ denote the ith bit of x. The PRF first initializes $k^0 = k$ and then iterates as follows: $k^{i+1} = \mathsf{E}_{k^i}(p_0^i)$ if $x(i) = 0$ and $k^{i+1} = \mathsf{E}_{k^i}(p_1^i)$ if $x(i) = 1$. Eventually, the $(n+1)^{th}$ intermediate key k^{128} is the PRF output as $\mathsf{F}_k(x)$.

In this basic version, the execution of the PRF guarantees that any side-channel adversary will at most observe the leakage corresponding to two plaintexts p_0^i and p_1^i per intermediate key. This implies 128 executions of the AES-128 to produce a single 128-bit output. A straightforward solution to trade improved performances for additional leakage is to increase the number of observable plaintexts per intermediate key. If one has N_p such plaintexts per stage, the number of AES-128 executions to produce a 128-bit output is divided by $\log_2(N_p)$. However, as will be discussed in Section 3, such a tradeoff scales badly and very rapidly decreases the side-channel security of an implementation. The more efficient alternative that we propose in this paper is based on a slight variation of this idea,

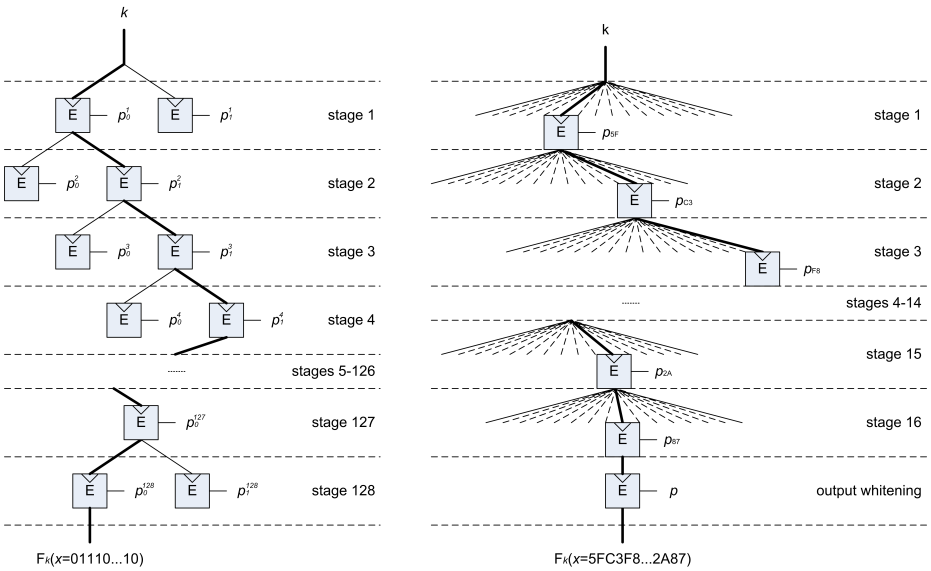

Fig. 1. Leakage-resilient PRFs: straight GGM (left) and efficient alternative (right).

illustrated in the right part of Figure 1. It can be viewed as a GGM construction with $N_p = 256$, but where the same set of 256 carefully chosen plaintexts is re-used in each PRF stage, excepted for the last stage where $N_p = 1$. Note that this is in contrast with the proof of leakage-resilience in [15], that requires all the p_j^i's to be public random values that are independently picked prior to encryption. In terms of efficiency, this proposal reduces the number of stages of a PRF based on the AES-128 to 17 (i.e. 16 plus one final whitening). As will be seen in Section 5, it also leads to interesting practical security guarantees.

3 Bounded data complexity may not be enough

Let us consider the PRF construction in the left part of Figure 1 with the AES-128 as block cipher. As previously mentioned, for each intermediate key k^i, this construction prevents adversaries to mount side-channel attacks with data complexity larger than 2. By contrast, nothing prevents the repetition of large number of measurements for the same input p_j^i. In this section, we investigate whether this condition of bounded data complexity is sufficient to guarantee practical security against side-channel attacks. For this purpose, we set up experimental attacks against an implementation of the AES in an 8-bit microcontroller, with limited data complexity (i.e. small N_p values). For illustration, we used the measurement setup and statistical tools previously described in [40]. More precisely, we considered template attacks in principal subspaces, using only power consumption measurements, and Principal Component Analysis (PCA) as dimensionality reduction technique. We exploited templates for two target operations, namely the AddRoundKey and SubBytes in the first AES round.

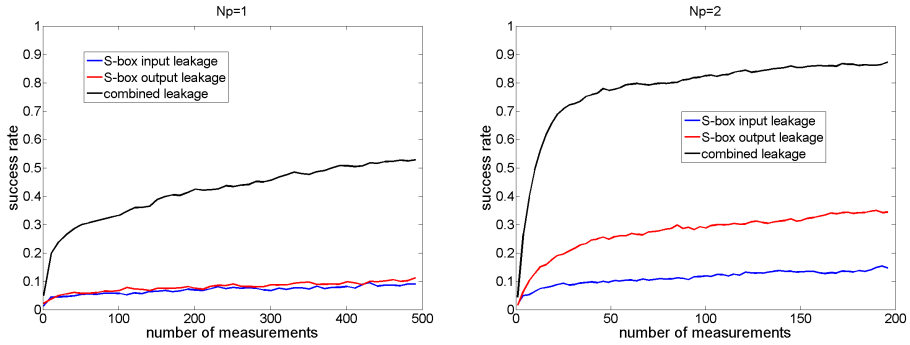

Fig. 2. Experimental attacks against the AES with bounded data complexity.

The results of attacks targeting the first AES master key byte are given in Figure 2 for $N_p = 1, 2$. The number of measurements used in the attacks is given on the x-axis, and their first-order success rate (following the definition in [41]) is given on the y-axis. The right part of the figure corresponds to the result for $N_p = 2$, i.e. the exact data complexity tolerated by the PRF construction. It can be observed that high success rates can already be obtained with our simple attack setting. In fact, due to the 8-bit bus of our microcontroller, even attacks with data complexity $N_p = 1$ allow reaching non-negligible success rates. As shown in Appendix C, Figure 8, this success rate dramatically increases with N_p, clearly suggesting that enhancing the PRF efficiency in this direction is not acceptable for security reasons. Admittedly, this simple scenario may not be reflective of better protected or larger, parallel devices. But it at least suggests that the security assumptions in all previous works on leakage-resilient PRFs overlook the important difference between data complexity and number of measurements.

As a result, two natural directions can be envisioned. On the one hand, one could design new (stateful) PRFs ensuring a bounded number of measurements. This would essentially correspond to the storage of all the intermediate nodes that have been computed in previous invocations of the tree-based PRF in Figure 1. Although different security vs. memory tradeoffs could be considered, this solution is hardly realistic from an implementation cost point of view. On the other hand, one could investigate the impact of large (parallel) implementations, where the bounded data complexity would be better reflected in the attacks' success rates. The following section investigates this second option.

4 Efficiently exploiting parallelism

In this section, we study how parallelism improves the security against DPA attacks and the efficiency of a tree-based PRF. For this purpose, we will mainly focus on one step of the constructions in Figure 1, and take the example of $k^0 = k$. In this context, there are three main parameters to consider when evaluating the side-channel security of the PRF, next denoted as N_p, N_s and σ_n^2. First, the

adversary is allowed to encrypt N_p (for now, random) plaintexts p_j $(1 \geq j \geq N_p)$ under the key k. Second, we target an AES-like block cipher where N_s S-boxes are executed in parallel. Finally, the leakage measurements are affected by a noise with variance σ_n^2. Let us denote the bytes of the plaintexts as $p_j[i]$ and the bytes of the key as $k[i]$. We will consider leakages of the form:

$$l_j = \sum_{i=1}^{N_s} \mathsf{L}(\mathsf{S}(p_j[i] \oplus k[i])) + n, \tag{1}$$

with S the AES S-box, L a leakage function and n a Gaussian-distributed noise with variance σ_n^2. In such a setting, parallelism essentially depends on the number of S-boxes N_s. Increasing this parameter typically allows increasing the amount of "algorithmic noise" in the attacks, as we now detail. For illustration, we considered a Hamming weight leakage function $\mathsf{L} = \mathrm{W_H}(\cdot)$ with $\sigma_n^2 = 0$, and a DPA adversary using Bayesian templates [9]. The left part of Figure 3 summarizes the joint effect of N_s and N_p in this random plaintext scenario, where the guessing entropy (in $\log_2$ scale) of the first master-key byte is used as evaluation metric [41]. It indicates the average position of the correct key byte in the scored list provided by the DPA attack, and thus reflects the key-search complexity of an adversary who is given such a list. One can clearly see the strong impact of increasing N_p, as the key search complexity decreases almost exponentially with it. In addition, the random plaintext scenario allows directly recovering information on all key bytes, by applying a straightforward divide-and-conquer strategy.

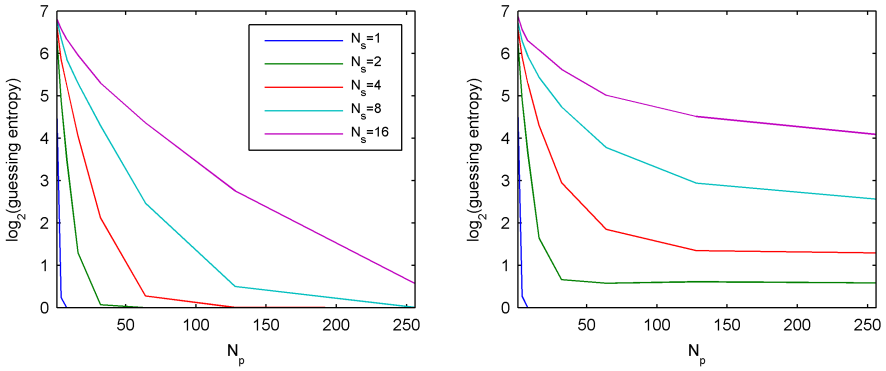

Fig. 3. Guessing entropy of the first AES master-key byte ($\log_2$ scale). Left: random (uniform) plaintext scenario. Right: carefully chosen plaintext scenario.

Careful selection of the plaintexts. The previous discussion highlighted that parallelism is not sufficient to guarantee security against side-channel attacks. In this section, we propose to tweak the PRF design with carefully chosen plaintexts, in order to prohibit the application of standard divide-and-conquer strategies. For this purpose, we define our plaintexts as the concatenation of N_s identical values j, i.e. $p_j = j||j||\ldots||j$, with $1 \leq j \leq N_p$ and N_p limited by the S-box

input space. Under conditions discussed later in this section, the effect of this measure is that in a DPA attack, the predictions corresponding to the N_s key bytes cannot be distinguished anymore. That is, all key bytes are targeted at the same time. As a result, and even when increasing N_p, not all the N_s key bytes can be highly ranked by the attack. This effect can be seen in the right part of Figure 3 and is reflected in a higher guessing entropy for the target key byte.

One important consequence of this observation is that the reduced guessing entropy of one key byte does not directly translate towards more key bytes. Indeed, the adversary now has to reconstruct a full key from a single score vector (rather than N_s ones in the random plaintext scenario). Intuitively, the task of reconstructing the full key could be divided into two steps: (1) picking a subset of N_s key bytes and (2) afterwards determining their order. Probably the most important result for this countermeasure is that even if the N_s correct key bytes are always ranked in the N_s first positions, task (2) still has a complexity of $N_s!$, a number which grows super-exponentially. We now discuss the conditions upon which this security parameter can actually be observed:

1. The leakage function L in Equation (1) has to be identical for all S-boxes.
2. Side-channel attacks exploiting unknown ciphertexts should be hard.

As far as the first condition is concerned, it is admittedly a new type of assumption. Therefore we investigated its practicality in Section 6, based on an FPGA case study. Our conclusions can be summarized as follows. (a) This assumption is indeed implementation-dependent. That is, we were able to identify both implementations with close to identical leakage models for all S-boxes, and implementations in which these models exhibit significant differences. (b) Even in the cases where significant differences occur, these differences could not be exploited. Essentially, this is because constructing the models (byte per byte, as imposed by computational constraints) has to be done for uniform plaintexts. That is, we assume parts of the bytes in the implementation to produce independent algorithmic noise (binomially, or approximately Gaussian distributed). By contrast, during an attack, the plaintexts are carefully chosen for all the bytes, hence generating a strongly key-dependent noise that was not characterized during profiling. As a result, the modeled leakage and the leakage during an attack do not match, which prohibits successful key recoveries. Summarizing, our experiments provide good indication that our assumption is sufficiently fulfilled for power measurements. As for EM measurements, it depends on the localization capabilities of the adversary. As discussed in [27], Chapter 3, distinguishing structures of a few hundred gates in complex circuits is a non-trivial task, especially for deep-submicron technologies. Hence, we believe that our countermeasure rules out an important part of low-cost EM attacks, and leave the investigation of advanced localization issues as an interesting question for future research.

As far as the second condition is concerned, first note that only the ciphertexts of the last step in the right construction of Figure 1 are given to the adversary. But for this last iteration, only one public plaintext p can be queried (i.e. the data complexity is bounded to one). For all the other steps, the ciphertexts remain internal intermediate values. In this context, we just observe

that most DPA attacks against block cipher implementations are based on the knowledge of either the plaintexts or the ciphertexts. To our knowledge, the best attacks in fully unknown input conditions are algebraic ones, e.g. [35, 36], which are hardly realistic in large parallel devices. Hence, it is reasonable to assume that the most critical threat against this PRF construction is taking advantage of the carefully chosen plaintexts. This scenario is investigated next.

5 Worst case security analyses

The previous section argued that breaking an AES-based leakage-resilient PRF taking advantage of parallelism could be at least as hard as enumerating a permutation over the AES S-boxes. This would correspond to $16! \approx 2^{44}$ for AES-128, $24! \approx 2^{79}$ for Rijndael-192 and $32! \approx 2^{117}$ for Rijndael-256. Hence, a natural question is to determine whether one can hope for more security, i.e. independent of their order, how difficult is the task of finding the correct N_s key bytes of the PRF? For this purpose, a first strategy is to apply a standard DPA attack and to enumerate the keys from the single score vector it provides. As discussed in Appendix A, this does not lead to any efficient key recovery and suggests large security guarantees. However, it turns out that in view of the design tweaks used in our PRF construction, standard DPA attacks are not anymore the most relevant tool for their security evaluation. In the rest of this section, we discuss two alternative techniques for attacking a PRF implementation. In both cases, the attacks rely on strong assumptions. Namely, we assume that the adversary has a perfect knowledge of the leakage function and can average his measurements in order to obtain noiseless leakages. As discussed in Appendix B, producing such noiseless traces may require significant amounts of measurement data. These conditions are motivated by the goal of investigating worst-case security. In this setting, we first describe an iterative type of DPA attack that significantly improves over the one in Appendix A. Next, we analyze the impact of advanced attacks using lattice reduction. In both cases, the results underline that the PRF construction does not offer much more security than what is bounded by the time needed to enumerate the permutation, if perfect measurements are available.

5.1 An iterative DPA-like attack

The aim of the iterative DPA attack is to recover the correct set of key bytes in the PRF implementation, exploiting the fact that one correct key byte is ranked at position one with high probability (see Appendix A). It works by iteratively removing the algorithmic noise corresponding to the best rated key bytes.

In the beginning of the attack, the adversary has an empty set of recovered key bytes and mounts a first DPA. As a result, he adds a first key byte to the set of recovered key bytes, corresponding to the highest rank in his (single) score vector. Next, he mounts a second DPA, this time adding the algorithmic noise corresponding to the already recovered key byte to his predictions. As a result, he adds a second key byte to the set of recovered key bytes. This procedure

is repeated until a set of N_s key bytes is recovered. Simulated DPA attacks in a noise free scenario and assuming that the adversary exactly knows the leakage model (i.e. in the same worst-case conditions as in Section 4) show that this simple strategy succeeds for $N_s = 16$ and $N_p = 256$ with a probability of $\approx 59\%$. However, as soon as we either increase N_s or decrease N_p, the success rate drops, as exhibited in Table 1. Additionally, there is no obvious way to (1) tell immediately if a wrong key byte was picked and (2) efficiently recover the master key from an incorrect set of key bytes. In the next section, we show that advanced attacks based on lattice reduction provide a more robust and systematic way to exploit the side-channel leakage of our PRF implementation.

Table 1. Success rates for the iterative DPA attack.

	$N_p = 4$	8	16	32	64	128	256
$N_s = 2$	0.0	30.3	81.4	98.6	99.7	99.7	99.5
4	0.0	0.0	13.0	78.3	96.8	97.3	97.8
8	0.0	0.0	0.0	10.1	69.8	89.2	91.1
16	0.0	0.0	0.0	0.0	3.8	36.7	58.8
32	0.0	0.0	0.0	0.0	0.0	0.2	2.9

5.2 Advanced attacks using lattice reduction

Let us first recall Equation (1) that describes our noiseless leakages:

$$l_j = \sum_{i=1}^{N_s} \mathsf{L}(\mathsf{S}(p_j[i] \oplus k[i])).$$

Similarly to the previous section, the goal of a lattice-reduction attack is to recover the vector of key bytes $k = \{k[1], k[2], \ldots, k[N_s]\}$ up to a permutation, from a vector of noiseless leakages $\overline{\mathbf{L}} = \{l_1, l_2, \ldots, l_{N_p}\}$. To simplify the analysis, we first assume that all key bytes in the vector k are distinct[1]. In this context, we denote byte-wise hypothetical leakage values as $l_a^b = \mathsf{L}(\mathsf{S}(b \oplus a))$, where b (resp. a) represents an hypothetical plaintext byte (resp. key byte). Next, we define a N_p-dimension vector $\bar{l}_a = \{l_a^1, l_a^2, \ldots, l_a^{N_p}\}$. Our problem can now be restated as finding a subset $\mathcal{K}$ of $[0, 1, \ldots, 255]$ containing N_s elements such that $\overline{\mathbf{L}} = \sum_{a \in \mathcal{K}} \bar{l}_a$. This turns the initial problem into a vectorial knapsack problem. To solve this knapsack problem, we can either try generic algorithms as in [5, 20, 37], or a lattice-based approach [24]. It is well-known that the lattice-based approach is very efficient for some knapsack problems and fails to work for other parameters. Since our context is quite specific, the parameters we are concerned with are not covered in standard textbooks. Moreover, our parameters are fixed and an asymptotic analysis does not make sense in this case. As a consequence,

[1] With this assumption, an exhaustive search on k can be achieved by trying all choices of N_s key values among 256. Under this exhaustive search attack, the security for $N_s = 16$ is 83 bits, for $N_s = 24$ it is 111 bits and for $N_s = 32$ it is 135 bits.

we decided to investigate the practical performance of a lattice reduction attack. As will be clear next, this lattice reduction approach is surprisingly efficient and the security estimates obtained by analyzing exhaustive search (in footnote 2) are overoptimistic. We note that the lattice based approach also outperforms the results obtained with generic algorithms. Hence, we only focus on this solution in the rest of the section. Taking the wost-case example of $N_p = 256$, we can construct the lattice spanned by the columns of the following matrix:

$$\begin{pmatrix} \kappa\,\bar{l}_0 & \kappa\,\bar{l}_1 & \cdots & \kappa\,\bar{l}_{255} & \kappa\,\overline{\mathbf{L}} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix},$$

where κ is a large enough constant to guarantee that any short vector in this lattice has its 256 first rows equal to 0. There exists a short vector of squared-norm N_s in the lattice which contains 0 in the first 256 rows and such that the next 256 rows are the characteristic vector of the set $\mathcal{K}$, i.e. there is a 1 in row $257 + i$ iff i is in $\mathcal{K}$ (all other rows contain zeros). Hence, if we find a short vector containing exactly N_s 1s, it can be converted into a set of keys which is compatible with the observed leakages. Note that for large values of N_p (e.g. 256), we expect only one solution for $\mathcal{K}$, which is experimentally verified next.

Note finally that if there are collisions in the vector of keys, we can still apply the same method with a minor change: the expected short vector becomes an encoding of a multiset. In particular, a key byte which appears twice is encoded by a 2. As a consequence, the principle of the attack is left unchanged. However, due to the presence of squares in the computation of the norm, the expected short vector has a larger norm which lowers the probability of success.

Experimenting the attack. As in the previous sections, we decided to consider a Hamming weight leakage function in our evaluations. In addition to the previously described case with $N_p = 256$, we again experimented with truncated versions of the vector $\overline{\mathbf{L}}$, i.e. with smaller N_p's. For each pair of parameters N_s, N_p, we performed 100 independent experiments (except for the case $N_p = 256$ where we performed 1000 experiments) and extracted the success rate and average execution time of the LLL algorithm using the FPLLL library [1] of Cadé, Pujol and Stehlé on an Intel Core i7-2820QM processor clocked at 2.30GHz. These results are given in Table 2. Note that a TBD entry means that we have

Table 2. Measured success rates and average timing for the lattice-based attack.

	$N_p = 256$	254	252	251	250	249	248	247	246	245
$N_s = 16$	100	100	100	100	100	100	100	100	100	100
	$1.3s$	$1.4s$	$1.4s$	$1.4s$	$1.5s$	$1.5s$	$3.1s$	$34.8s$	$73.0s$	$131.4s$
24	99.9	100	100	100	100	100	100	100	TBD	TBD
	$1.4s$	$1.4s$	$1.4s$	$1.4s$	$1.5s$	$1.5s$	$3.1s$	$35.5s$	$\approx 88s$	$\approx 143s$
32	79.6	79	79	83	80	79	76	TBD	TBD	TBD
	$1.4s$	$1.5s$	$1.5s$	$1.5s$	$1.6s$	$1.6s$	$3.3s$	$\approx 33s$	$\approx 81s$	$\approx 140s$

only performed a single test in order to determine an approximate running time but no meaningful probability of success. These results clearly exhibit that the LLL-based approach outperforms the heuristic iterative DPA in the previous section. Yet, one can observe that decreasing the number of leakages in large implementations (e.g. for $N_s = 32$) leads to significant increases of the execution times. More detailed results for the $N_p = 256$ case are presented in Table 3, also reflecting the fraction of key vectors containing collisions in our experiments.

Table 3. Additional data for $N_p = 256$.

N_s	Key vectors w/o collisions	Successes w/o collisions	Key vectors w/ collisions	Successes w/ collisions	Overall fraction	Timing
16	610	610	390	390	100 %	1.3s
24	328	328	672	671	99.9 %	1.4s
32	141	137	859	659	79.6 %	1.4s
40	40	23	960	479	50.2 %	1.6s

Improving the success rate. In order to improve the probability of success when N_s grows, we can also combine the lattice reduction approach with a partial exhaustive search. The idea is to guess the contribution of some fixed vector, to subtract this guessed contribution from the target vector and to re-run the attack without the guessed vector and with a smaller short vector.

6 Practical instantiation issues

The previous sections of this paper suggest that a leakage-resilient PRF offers interesting security arguments compared to state-of-the-art countermeasures against side-channel attacks. Motivated by the need to understand the impact of different parameters in a PRF implementation, our analysis was mostly based on idealized leakage functions. In this section, we complement this view with a first discussion of some important practical instantiation issues.

Performance evaluation. We evaluated the hardware performance of our construction based on AES-128 (LRPRF-128) and Rijndael-192 (LRPRF-192). For this purpose, we opted for a fully parallel, encryption-only implementation of the algorithms. In addition to these block ciphers, the PRF designs also contain a register to store the x value and some control logic to operate the blocks. Using Synopsis Design Compiler 2010 and the STM 65nm CMOS standard cell library, this resulted in an area of 9.97 kGE (resp. 14.43 kGE) for LRPRF-128 (resp. LRPRF-192). An encryption takes 10 (resp. 12) cycles plus 2 cycles to load the key and the plaintext. Thus, the complete PRF evaluation with a 16 (resp. 24) byte value for x takes $17 \times 12 + 2 = 206$ (resp. $25 \times 14 + 2 = 352$) cycles, where the additional two are again for loading the key and the x value. This is in fact in line with state-of-the-art protected implementations like the one by Moradi et al. from Eurocrypt 2011 [30]. Their threshold implementation of AES-128 takes 266 cycles at an area of 11.12 kGE. In addition, we mention that given some

memory overheads, the PRF construction gains particular interest when used for encryption in counter mode. It enables starting the PRF evaluation from intermediate results of previous evaluations. For instance, producing a 512-bit keystream can be done in only $(17+2+2+2)\times 12+2=278$ cycles (given that there is no overflow of the least significant byte in the IV).

Investigation of leakage models. In order to analyze the practicality of the requirement that the S-box leakage models must be identical, we performed a case study on the SASEBO evaluation platform [2] and measured the power consumption of two circuits. The first one consisted of two block-RAM based S-boxes and the second one implemented two S-boxes following Canright's approach [8]. For both circuits, we acquired one million traces and built templates from them. That is, for each S-box within a circuit, we characterized 256 Gaussian distributions corresponding to the 256 possible inputs. We then used the mean values of these distributions as the leakage model for an S-box. The extracted leakage models can be seen in Figure 4. For block-RAM based S-boxes, they show a Pearson correlation of 0.996. This means that for carefully designed implementations, the requirement of identical leakage models can indeed be fulfilled. By contrast for the Canright implementation, there was a visible layout difference between the two instances on the FPGA. Therefore, also the models differed and the correlation of the mean values decreased to 0.686. From this case study, we can conclude that for some implementations, there are leakage differences which can be extracted by an adversary with profiling capabilities. In the next paragraph, we discuss whether these differences can be exploited.

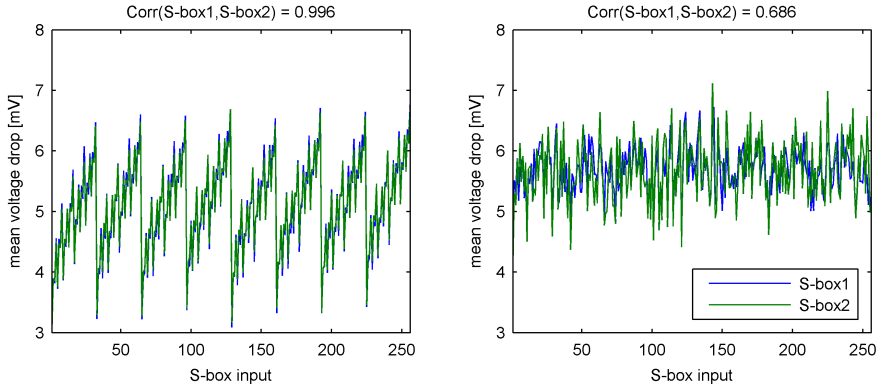

Fig. 4. Leakage models: block-RAM based S-box (left) and Canright S-box (right).

Impact of algorithmic noise. In traditional DPA attacks, the algorithmic noise is considered to be Gaussian and, due to uniformly distributed inputs, averages out for a sufficient number of inputs. In our case on the other hand, all inputs are determined by a single byte and a fixed key. Therefore the algorithmic noise cannot be averaged out and in addition, is fully determined by the unknown part of the key. Clearly, directly profiling such kind of noise is computationally

hard (it corresponds to performing a DPA directly on the full master key). Therefore, and in order to analyze the effect of this key-dependent algorithmic noise, we performed the following simulated experiment. First, we implemented $N_s = \{2, 4, 8\}$ Canright S-boxes on the FPGA (i.e. we considered the most different leakage models). For each S-box S_i with $i \in [1; N_s]$, we measured 400 traces for each input, while keeping the inputs to the other S-boxes at zero. This way we could build precise leakage models L_i without acquiring any algorithmic noise. To simulate real traces where all S-boxes operate in parallel, we then built the overall leakage function as: $\mathsf{L}'(p) = \sum_{i=1}^{N_s} \mathsf{L}_i(\mathsf{S}_i(p[i])) + n$, where p is the N_s-byte input, $p[i]$ is the ith byte of the input, and n an Gaussian distributed measurement noise estimated from our data set. This leakage description was then used to simulate our N_s S-box device from which we could sample traces for arbitrary inputs. From this point, we proceeded as usual. That is, we built N_s templates (now including algorithmic noise) by sampling 100 million traces from $\mathsf{L}'(\cdot)$. Next, we launched template attacks by sampling 300 times 256 000 traces (for 300 different keys). The results of these attacks can be seen in Figure 5.

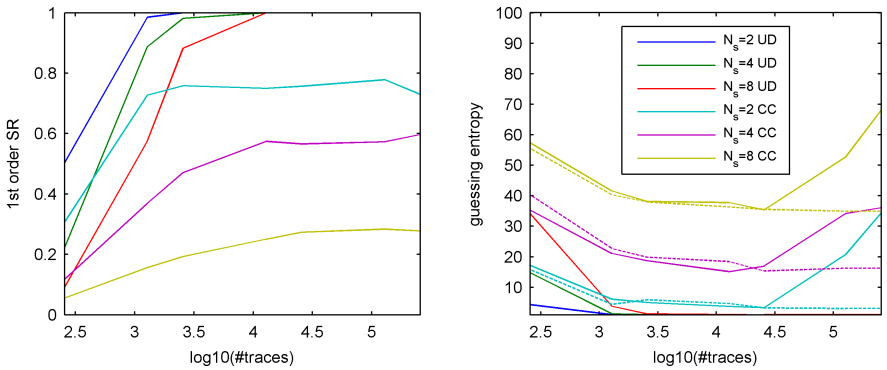

Fig. 5. Impact of algorithmic noise reflected by the success rate and guessing entropy.

For the uniformly distributed (UD) plaintexts, all subkeys are recovered correctly after 3 000 and 13 000 traces respectively, indicated by a guessing entropy of one and a first-order success rate of one. Both metrics are averaged over the N_s S-boxes. By contrast, for the carefully chosen (CC) plaintexts, it can be observed that the success rates stagnate at the same time as when they reach one for the uniformly distributed plaintexts. This is because for some subkeys the models will fit. Hence, those subkeys can be recovered with good probability. But for the remaining subkeys it is not possible to carry out a successful recovery. In the case of a template attack (represented with plain curves in the figure), this means that the probability of the correct key will diminish at some point, which is the reason why the guessing entropy increases again after 26 000 traces. In the case of a correlation attack using the templates' mean value as model (instead of the usual Hamming weight model [7]), this effect vanishes, as represented by the dotted lines. That is, the "hard to recover" subkeys then stagnate at fixed

ranks in the lists (corresponding to a fixed correlation coefficient value), rather than decreasing towards a probability zero, due to an incorrect model. Thus, we can conclude that even if there are actual differences in the leakage models of the different S-boxes of a PRF implementation, and strong profiling is possible for the adversary, the key-dependent algorithmic noise prevents the building of a sound leakage model. For example, already for 8 parallel S-boxes, each subkey remains with a guessing entropy of ≈ 35 in our case study. It would further increase with more parallel S-boxes (the previously described PRF implementations would have at least 16 and 24 ones, respectively). Hence, the only way to perform successful key recoveries in these cases would be to build templates for the full key, which is unrealistic for computational reasons.

Preventing DPA attacks against MixColumns. The central result of our security analyses is that performing a side-channel attack against our parallel PRF implementation should at least require to enumerate a permutation over N_s S-boxes. However, this implicitly assumes that the only path for performing a DPA is this operation, which neglects the possibility to mount attacks against MixColumns. In general, such attacks are more computationally intensive, as they require guessing 2^{32} key candidates. Yet, this remains achievable with modern computers. Given that such attacks succeed, it would only remain to enumerate a permutation of the MixColumns operations (i.e. $4! \times 4$ for the AES-128, where the factor 4 relates to the fact that the adversary would recover the 32-bit subkeys up to a byte-wise rotation). However, this attack may not always be applicable in practice, and can be made more computationally intensive, as we now discuss. First note that in a hardware implementation, the adversary may have to target the Hamming distance between the state register values before and after the first round. But one byte of this value depends on five key bytes and four bytes of this value depend on eight key bytes, which is harder to guess. In addition, there is a simple and general trick to increase the amount of bytes to guess after the MixColumns transform. Namely, one just has to switch the order of MixColumns and AddRoundKey. This requires that the key schedule applies the inverse MixColumns operation to the round keys before outputting them. Since the only non-linear operation in the key schedule is SubWord, the key schedule can operate on accordingly recoded round keys and, instead of applying SubWord, apply the sequence MixColumns, SubWord, InvMixColumns. The costs of this accounts for one InvMixColumns and one MixColumns unit. Finally, any attack against larger subkeys would still have a bounded data complexity of 256 with key-dependent algorithmic noise. Summarizing, the protection of MixColumns against DPA can be enhanced by different architectural means. Besides, and quite interestingly, this discussion highlights that the AES is not the best suited algorithm for integration in our leakage-resilient PRF. Hence, it suggests the design of block ciphers with more convenient diffusion layers for this purpose as another interesting scope for further research.

Security against fault attacks. Finally, there is an additional advantage to our construction. Usually, the resources for side-channel attacks and fault protections cannot be shared. For leakage-resilient PRFs, on the other hand, we can

provide a first-order fault protection based on temporal redundancy, by just repeating the last step of the construction. Taking the LRPRF-128 as an example, we would perform 18 instead of 17 encryptions. This accounts for an overhead of only 5.8%, rather than the usual 100% for block ciphers.

7 Conclusions & consequences for block cipher design

This paper describes tweaks to improve both the practical security and the efficiency of leakage-resilient PRFs. They allow quantifying physical security with a parameter that has super-exponential impact on the time complexity of a successful attack. They also open the paths towards real world applications, as their performance overheads are in line with other countermeasures against side-channel attacks. In particular, the only known countermeasure with an exponential security parameter is masking. But increasing the number of masks in a block cipher implementation is generally (much) more expensive than increasing its parallelism. Next, our results suggest interesting challenges for the design of new block ciphers, as the AES Rijndael appears not to be an ideal candidate for integration in leakage-resilient constructions. Possible tracks for investigation include modifying the number and size of S-boxes (that directly affect the security vs. efficiency tradeoff of the PRF), reducing the number of rounds in the inner steps of the construction, and improving diffusion layers in order to avoid the possible attacks after the diffusion layer described in Section 6.

References

1. http://perso.ens-lyon.fr/xavier.pujol/fplll/.
2. http://staff.aist.go.jp/akashi.satoh/sasebo/en/board/sasebo.html.
3. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
4. R. J. Anderson and M. G. Kuhn. Low cost attacks on tamper resistant devices. In B. Christianson, B. Crispo, T. M. A. Lomas, and M. Roe, editors, *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.
5. A. Becker, J.-S. Coron, and A. Joux. Improved generic algorithms for hard knapsacks. In Paterson [32], pages 364–385.
6. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.
7. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
8. D. Canright. A Very Compact S-Box for AES. In J. R. Rao and B. Sunar, editors, *CHES*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.
9. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. K. Jr., Çetin Kaya Koç, and C. Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

10. G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In Halevi and Rabin [19], pages 225–244.
11. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In M. Mitzenmacher, editor, *STOC*, pages 621–630. ACM, 2009.
12. Y. Dodis and K. Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In T. Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.
13. S. Dziembowski. Intrusion-resilience via the bounded-storage model. In Halevi and Rabin [19], pages 207–224.
14. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.
15. S. Faust, K. Pietrzak, and J. Schipper. Practical leakage-resilient symmetric cryptography. in these proceedings, 2012.
16. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In Naor [31], pages 258–277.
17. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
18. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In P. C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.
19. S. Halevi and T. Rabin, editors. *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*. Springer, 2006.
20. N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010.
21. Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.
22. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
23. A. Joux, editor. *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*. Springer, 2009.
24. A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology*, 11(3):161–185, 1998.
25. P. C. Kocher. Leak resistant cryptographic indexed key update. US Patent.
26. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
27. V. Lomne. Power and electro-magnetic side-channel attacks: Threats and countermeasures. PhD thesis, Universite de Montpellier II, 2010.
28. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
29. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In Naor [31], pages 278–296.
30. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the limits: A very compact and a threshold implementation of aes. In Paterson [32], pages 69–88.

31. M. Naor, editor. *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*. Springer, 2004.
32. K. G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.
33. C. Petit, F.-X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In M. Abe and V. D. Gligor, editors, *ASIACCS*, pages 56–65. ACM, 2008.
34. K. Pietrzak. A leakage-resilient mode of operation. In Joux [23], pages 462–482.
35. M. Renauld and F.-X. Standaert. Algebraic side-channel attacks. In F. Bao, M. Yung, D. Lin, and J. Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.
36. M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon. Algebraic side-channel attacks on the aes: Why time also matters in dpa. In C. Clavier and K. Gaj, editors, *CHES*, volume 5747 of *LNCS*, pages 97–111. Springer, 2009.
37. R. Schroeppel and A. Shamir. A t=o($2^{n/2}$), s=o($2^{n/4}$) algorithm for certain np-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981.
38. F.-X. Standaert. How leaky is an extractor? In M. Abdalla and P. S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 294–304. Springer, 2010.
39. F.-X. Standaert. Leakage resilient cryptography: a practical overview. ECRYPT Workshop on Symmetric Encryption (SKEW 2011), Copenhagen, Denmark, 2011. http://perso.uclouvain.be/fstandae/PUBLIS/96_slides.pdf.
40. F.-X. Standaert and C. Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.
41. F.-X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In Joux [23], pages 443–461.
42. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. In A.-R. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer Berlin Heidelberg, 2010.
43. Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS*, pages 141–151. ACM, 2010.

A Security against standard DPA attacks

The result of a standard (template-based) DPA attack against our scheme is a single vector, in which all possible subkeys are ranked according to their probability. From this, a full key consisting of N_s bytes has to be reconstructed. Ideally, the set of the N_s correct subkeys would be ranked first and all other subkeys would have a low probability. However, looking at the distribution of the subkeys within such a vector after a noise-free attack shows that this is not the case (mainly because of the algorithmic noise). Figure 7 illustrates where the correct

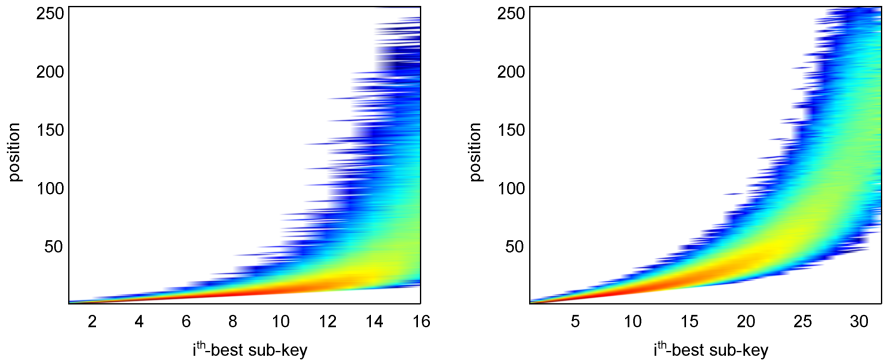

Fig. 6. Distribution of the correct key bytes within a probability vector.

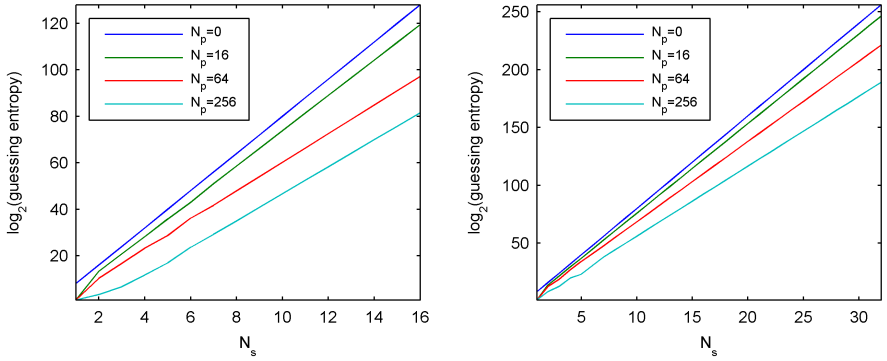

Fig. 7. Estimated and extrapolated guessing entropy of the full key.

subkeys can be found within the vector on average. Whereas the best-ranked correct key byte can be almost with certainty found at position one, some of the correct subkeys are ranked much lower. Starting from such a vector, the optimal adversarial strategy is to enumerate full keys according to their probability, where all up-to-permutation-identical keys have the same probability. Following this strategy, we estimated and extrapolated the guessing entropy. This was done by generating a probability vector for N_s values between one and seven with a constant noise variance which would correspond to an algorithmic noise of 16 or 32 parallel S-boxes. From each vector we sampled 2^{30} random full keys and checked the position of the correct full key within this set. Afterwards, we scaled this position to 256^{N_s} and added the complexity for the permutation. Finally, since we could observe a power-law for the guessing entropy, we extrapolated these values up to 16 and 32 S-boxes using the slope in log-scale. The very fact that the end points of these extrapolations suggest a security of 2^{84} and 2^{185} show that standard DPA cannot be the optimal strategy.

B Averaging effort to obtain noiseless traces

As the security evaluations in Section 5 both consider noiseless traces, an interesting question is to determine the averaging effort that would be needed to obtain such high quality information from an actual implementation. For this purpose, we measured a fully parallel AES-128 FPGA implementation on the SASEBO evaluation platform [2]. We considered traces close to noise-free if we can correctly identify the 128-bit Hamming distance value for 256 measurements, with a probability of 0.90. Thus, each of the 256 measurements must be classified correctly with a probability of $(0.9)^{1/256} = 0.9996$. This in turn corresponds to a confidence interval of $3.54\,\sigma$, assuming a normal distribution of the noise. Thus, to allow error-free decoding, we need the mean values for the Hamming weights to be twice that value apart, meaning $7.08\,\sigma$. Given the distance between the mean values $\Delta\mu$ and the standard deviation of our measurements, we can calculate the number of traces to average as:

$$n = \left(\frac{7.08\,\sigma}{\Delta\mu}\right)^2 .$$

As we get only sample means from our measurements, we calculated n for the average and the minimum value of $\Delta\mu$. The latter one is also motivated by the fact that, if due to the power model the $\Delta\mu$ values are not equidistant, then the smallest distance determines n. For the average $\Delta\mu$ we found $n = 6.9 * 10^3$ and for the minimum value we found $n = 8.8 * 10^6$. To get a close to noise-free mean trace for every plaintext, we additionally need to multiply this number by 256, thus we need to acquire a total number of $1.78 * 10^6$ and $2.27 * 10^9$ traces for the minimum and average $\Delta\mu$, respectively. The actual number of traces to acquire most likely lies somewhere between these two extreme values. Hence, it suggests that the averaging effort can be expected to be non-negligible.

C Additional figure

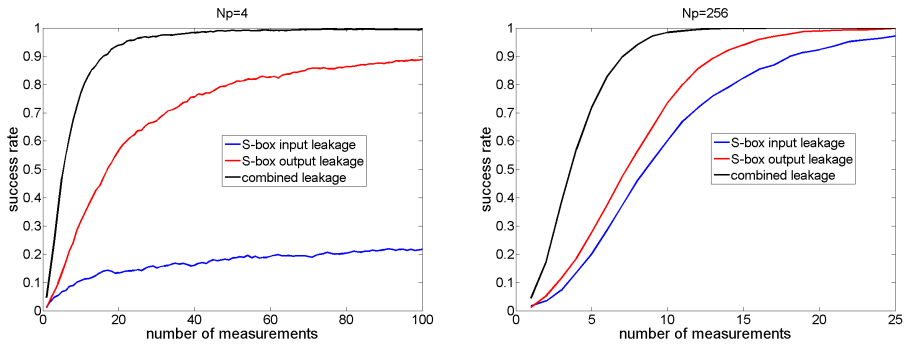

Fig. 8. Experimental attacks against the AES with bounded data complexity.