Leakage Resilient Primitive

Document version: 1.1

Marcel Medwed Ventzislav Nikov Martin Feldhofer
Bruce Murray Mario Lamberger

NXP Semiconductors

March 27, 2017

In this work we present a practical implementation of a leakage resilient pseudo-random function (PRF) and analyze its security. The assumed adversary against whom the scheme is shown secure is able to profile the device and has no bound on the number of traces to be used in the attack. The construction is based on the one by Goldreich, Goldwasser and Micali (known as GGM construction [12]) but has two additional steps, namely input pre-processing (and output post-processing). In the outer layer for the pre-processing (and post-processing), an attacker might know the input (or output), but is limited to two such known inputs (or to one such known output) per key. Hence, he faces a simple power analysis (SPA) scenario. In the inner layer where the actual GGM tree is processed, no known values are processed, hence an adversary needs to mount an unknown-input attack. To the best of our knowledge, this is the first practical leakage resilient construction which can be based on a standard block cipher like AES-128. This is because we neither need to rely on the number of parallel S-boxes to be as large as 24 or even 32, nor does our leakage resilient property only hold for the first round of the cipher like in previous works [3, 20]. Furthermore, this is the first paper to analyze unknown input attacks. We show that they are easy to conduct against a single S-box, but become intractable as soon as an adversary faces four and more parallel S-boxes.

Contents

Chapter 1

Side-Channel Foundations

1.1 Introduction

Countermeasures against side-channel attacks always imply implementation overheads and rely on physical assumptions. So designing such countermeasures comes with the equally important goals of maximizing security, while minimizing the overheads and relying on physical assumptions that are easy to fulfill by cryptographic engineers. Mainstream masking schemes (i.e. data randomization based on secret sharing) are a typical example of this tradeoff, where security is exponential in the number of shares, performances are quadratic in the number of shares, and implementers need to guarantee that the leakages of the shares are independent and sufficiently noisy [6, 9, 14, 24]. (Note that the condition of independent leakages is typically hard to guarantee, both in software and hardware implementations [1, 7, 17, 18]). Threshold implementations are a specialization of masking that reduces the independence requirement (by ensuring that glitches do not harm the security of the masked implementations) [4, 22], which can also lead to some performance gains with low number of shares [5, 21].

At CHES 2012, a quite different tradeoff was introduced. Namely, and starting from the observation that leakage-resilience via re-keying alone is not sufficient to efficiently protect stateless symmetric cryptographic primitives such as block ciphers (later formalized in [2]), Medwed et al. proposed a tweaked construction of AES-based leakage-resilient PRF, inspired from more formal works such as [8, 10, 26, 29], which additionally requires that the AES is implemented in parallel and that its S-boxes have similar leakage models [20]. In this respect, and while the parallel implementation setting is easy to guarantee (and can even be emulated thanks to shuffling [13]), the "similar leakage assumption" turned out to be harder to evaluate. Later results showed that despite not easy to attack, such a solution may not be best suited to the standard AES cipher [3].

In this paper, we aim to improve the tradeoff between security, performance and physical assumptions for the CHES 2012 construction. For this purpose, our main ingredient

is to replace the similar leakage assumption by an easier-to-guarantee requirement of unknown plaintexts. Interestingly, this requirement can be easily satisfied by exploiting a leakage-resilient stream cipher in order to generate these plaintexts (we use the efficient construction from [25] for this purpose). As a result, our contributions are as follows. We first describe our new construction of leakage-resilient PRF based on unknown plaintexts. Second, we analyze its security in front of standard side-channel attacks where the adversary can observe noisy Hamming weight leakages (and compare it with the CHES 2012 proposal). Third, we evaluate the impact of implementation issues such as deviations from the Hamming weight leakages and leakages due to transitions between registers. Finally, we discuss alternative attack paths and put forward the good performances of our new construction. As part of our investigations, we also highlight the interesting security guarantees offered by the combination of unknown cipher inputs and parallel implementations for side-channel resistance, which is of independent interest.

1.2 Background: the CHES 2012 leakage-resilient PRF

We start with the description of the standard GGM PRF [12], depicted in the left part of Figure 1.1, on which the CHES 2012 PRF is based. 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 shows the example of $\mathsf{E} =$ AES-128 with $1 \leq i \leq 128$ and $0 \leq j \leq 1$). Let also $x(i)$ denote the i^{th} 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 per intermediate key (p_0^i and p_1^i). 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 already discussed in [20], such a tradeoff scales badly and very rapidly decreases the side-channel security of an implementation (as it typically allows DPA with N_p observable plaintexts).

To avoid this drawback, an efficient alternative (also proposed in [20]) is illustrated in the right part of Figure 1.1. 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 $N_p = 1$. 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).

The security of this second construction is based on the combination of parallelism with carefully chosen plaintext values, in order to prohibit the application of standard divide-and-conquer strategies. For this purpose, plaintexts of the form $p_j = \{j-1\}^{N_s}$, with

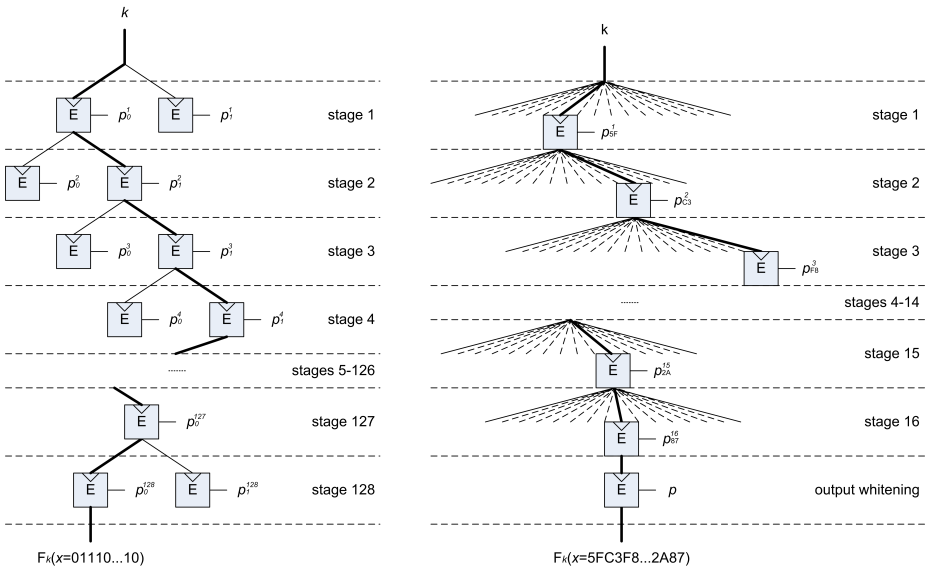

Figure 1.1: Leakage-resilient PRFs: straight GGM (left) and efficient alternative (right).

$1 \leq j \leq N_p$ and N_p being limited by the S-box input space were considered. Given that all S-boxes leak in parallel, the effect of this measure is that in a DPA attack, the predictions corresponding to the N_s key bytes cannot be distinguished anymore, because these key bytes have to be 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. (We will re-detail this effect in Section 1.4.1, which is reflected by the higher guessing entropy of the targeted key bytes in Figure 1.3). In [20], it was even shown that slight differences in the implementation – and therefore in the leakage of the N_s S-boxes – are not easily exploitable. Eventually, if N_s becomes sufficiently large, ordering the N_s recovered subkeys becomes has a cost of $N_s!$, meaning that even after seeing all leakages without noise, the adversary cannot fully recover the key.

Unfortunately, and despite conceptually appealing, this construction has several drawbacks which limit its applicability. First, the security parameter N_s is defined by the number of S-boxes of the underlying block cipher. For some of the currently standardized block ciphers N_s is not large enough (e.g. $N_s = 16$ for the AES-128, which corresponds to an insufficient $N_s! \approx 2^{44}$). Second, if intermediate values other than the first round's S-box outputs are targeted, the leakages might be sufficiently independent such that divide-and-conquer strategies work again. While this generally requires more computational power, recent results on multi-target attack DPA show that it is not out of reach [19]. (This is in fact the reason why attacks on the ciphertext need to be prevented by the whitening step in the CHES 2012 proposal). Finally, the size of the S-box defines the maximum

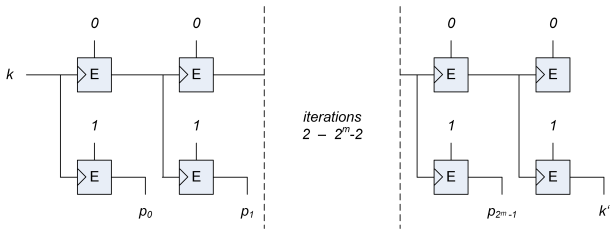

Figure 1.2: Leakage-resilient PRG used to generate the 2^m secret plaintexts.

value of N_p and hence the maximum throughput.

1.3 New leakage-resilient PRF construction

We now present a new construction which improves over the one in [20] in terms of performance and security, at the cost of higher memory requirements. For this purpose, we introduce a pre-computation step in which we generate N_p secret, distinct plaintexts. This step can be seen in Figure 1.2. It essentially uses the leakage resilient PRG from [25] to generate 2^m secret plaintexts $p_0 \dots p_{2^m-1}$ as well as an updated key k'. These secret plaintexts and updated key are then simply used in a tree-based PRF such as in the right side of Figure 1.1. The output whitening step stays the same. By design, this new construction has the advantage (compared to the CHES 2012 one) that the plaintexts are secret and of no particular form. This implies that their number is not bounded by the S-box size, allowing for smaller trees of depth $128/m + 1$. From a security point of view, it also comes with interesting implications:

1. Since the plaintexts are unknown, a straight-forward unprofiled DPA is ruled out. Instead, an adversary has to build templates (for instance for the bi-variate variable made of the plaintext and S-box output leakages).

2. For a similar reason, there is no straightforward way to verify a key candidate: for this purpose, one would not only need to recover the key but also at least one secret plaintext. In the worst case where the information leakages are not sufficent (i.e. if a successful attack requires additional key/plaintext enumeration [28]) this squares the attack time complexity.

3. As for the CHES 2012 construction using carefully chosen plaintexts, the adversary has no way of separating the leakages from the different subkeys. But contrary to this previous work, this feature now applies to any intermediate variable within the algorithm (not only to the first round leakages).

1.4 Security analysis w.r.t. basic side-channel attacks

We now detail our security analysis against standard side-channel attacks and use the following notations. First, k denotes a key, k^* denotes a key candidate and k_j the j^{th} byte of a key. Next, $p_{i,j}$ is the j^{th} byte of the i^{th} plaintext out of q ones that are available to an adversary. For p and k, j is assumed to be in the range $1, \ldots, N_s$ where $N_s = 16$ for AES. Further, t_i is a trace (aka leakage) vector, corresponding to the i^{th} plaintext. A trace may contain several leakage points, denoted by $t_{i,j}$. L denotes the leakage function, e.g. the Hamming weight function in our examples below. Finally, $\mathsf{L}(\mathsf{S}(k_1 \oplus p_{2,1}))$ denotes the leakage of the S-box output corresponding to S-box 1 for the 2nd plaintext. The set of all plaintexts is denoted as P and the set of all traces as T.

In a standard DPA attack, the adversary computes the correct key as:

$$\tilde{k} = \arg\max_{k^*} \Pr(k^* | p_1 \ldots p_q, t_1 \ldots t_q).$$

The attack is successful if $\tilde{k} = k$. In a parallel hardware scenario, t_i consists of a single leakage point that we approximate as:

$$t_{i,1} = \sum_{j=1}^{N_s} \mathsf{L}(\mathsf{S}(k_j \oplus p_{i,j})). \tag{1.1}$$

Nevertheless, even in this parallel scenario, an adversary can always target a single key byte at a time by computing:

$$\tilde{k_j} = \arg\max_{k_j^*} \Pr(k_j^* | p_{1,j} \ldots p_{q,j}, t_1 \ldots t_q).$$

In this case, by just looking at a specific S-box or byte of the key, an adversary neglects the other key bytes and their contribution to the leakage is interpreted as (algoirithmic) noise, which eventually averages out if plaintexts are uniformly distributed. As already discussed in [20], for carefully chosen plaintexts, $p_{1,1} = p_{1,j}$ for all $j = 1 \ldots N$. Therefore, the equation becomes:

$$\tilde{k_j} = \arg\max_{k_j^*} \Pr(k_j^* | p_{1,1} \ldots p_{q,1}, t_1 \ldots t_q) \tag{1.2}$$

and all $\tilde{k_j}$ are the same. That is, since the probability condition is no longer dependent on j, only one joint score vector can be obtained, which contains the information about all the N_s target key bytes at once. For unknown-plaintext attacks, an adversary finally faces the problem of finding:

$$\tilde{k_j} = \arg\max_{k_j^*} \Pr(k_j^* | (t_{1,1}, t_{1,2}) \ldots (t_{q,1}, t_{q,2})), \tag{1.3}$$

where he has no direct access to plaintext information, and therefore must extract this information from the traces as well (reflected by the second sample of the traces in the equation). We assume that this information is separately available and that the traces take the form:

$$(t_{i,1}, t_{i,2}) = \left(\sum_{j=1}^{N_s} \mathsf{L}(p_{i,j}), \sum_{j=1}^{N_s} \mathsf{L}(\mathsf{S}(k_j \oplus p_{i,j})) \right). \tag{1.4}$$

This has the following important implications on the attack:

1. The adversary cannot apply a divide-and-conquer brute-force attack anymore. As in the case of carefully chosen plaintexts, also here the probability's condition becomes independent of j, which results in only a single score vector containing the information for all the N_s subkeys.

2. Successful attacks have to be bi-variate ones, in which a second-order moment of the leakage distribution is exploited. This makes them more sensitive to noise. Furthermore, as for the CHES 2012 construction as well, $N_s - 1$ contributors for each leakage point represent key-dependent algorithmic noise, and cannot be averaged out like in the case of masking (as noted in [2]).

In the following we present three experiments. In the first one we recap the security of the CHES 2012 scheme in order to allow for a later comparison. We do so by estimating the guessing entropy and the subkey rank distribution as a function of N_s after seeing all possible traces. In the second experiment, we do the same for our improved proposal. This allows us to highlight the security improvement. In a third experiment we look at the model errors which are the reason for the security improvement. All experiments are carried out based on template attacks as this represents the most powerful side-channel adversary. We used discrete histograms (instead of continuous distributions) for our templates since the leakage function (aka power model) used in our experiments is also discrete and no noise is added. Hence, the number of bins is determined automatically and the histograms capture all the available information. Finally, we evaluate our metrics for increasing number of traces (but bounded number of plaintexts in the case of the CHES 2012 and our new construction).

1.4.1 Security based on carefully chosen plaintexts

For the CHES 2012 scheme, the plaintexts are known, the target function is the AES S-box and the assumed power model is the Hamming weight model. Thus, the leakages are in the form of Equation (1.1). Knowing this, we can generate a template $\mathcal{D}^i$ for each of the subkey candidates, assuming the plaintext to be zero. In our simulations we look at N_s parallel AES S-boxes and the leaking variables are 8-bit valued. Therefore, each template is a histogram with $8 \cdot N_s + 1$ bins, starting at bin $\mathcal{D}^i(0)$ which indicates the probability that for a subkey $k = i$, the leakage sample has a value of 0. The templates are built according to Algorithm 8.

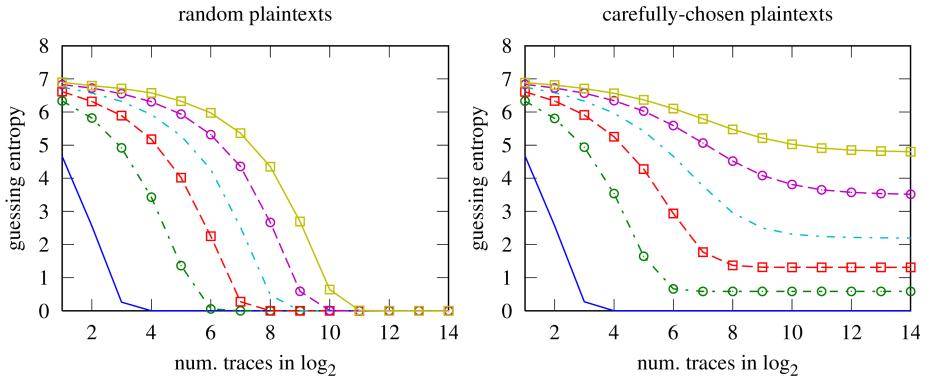

Figure 1.3: Average guessing entropy after attacks with known plaintexts for $N_s = 1$ (blue,s/), 2 (green,dd/c), 4 (red,d/s), 8 (cyan,dd/), 16 (magenta,d/c), and 32 (yellow,s/s) with {s=solid,d=dashed,dd=dotted dashed}/{c=circle,s=square}.

During the attack phase, the templates have been permuted according to the plaintext byte, that is, the probability for a certain leakage given a certain plaintext was calculated as $\Pr(t_1|p_{1,j}, k_j^*) = \mathcal{D}^{k_j^* \oplus p_{1,j}}(t_1)$.

The result of the known plaintext attack can be seen in Fig. 1.3. The left plot represents a scenario where the plaintexts where not carefully chosen and therefore, the S-boxes leak independently. This just serves as a reference for the right plot, where the actual CHES 2012 scheme with carefully chosen plaintexts was analyzed. The y-axes represent the average key rank of k_1 in $\log_2$-scale. A random guess would result in an average key rank of 128 and thus a 7 in $\log_2$-scale indicates that no information was retrieved via the side channel. Zero on the other hand indicates that the correct key was ranked first and thus it has been recovered with certainty. The x-axis shows the number of required traces to reach a certain average key rank, again in $\log_2$-scale. The different curves represent different numbers of parallel S-boxes ranging from 1 to 32 in powers of two. Each curve has been averaged over 10k attacks. On the right side we can observe a stagnation of the average rank at approximately $(N_s+1)/2$ for $N_s \leq 8$ (in $\log_2$ this results in 0, 0.6, 1.3, and 2.2). As the adversary targets all subkeys at the same time, this is what one would expect intuitively. However, for 16 and 32 S-boxes, the average rank becomes higher, namely $\log_2(11.2) = 3.5$ (instead of 3.1) and $\log_2(27.1) = 4.8$ (instead of 4.0). This may look surprising, since due to the higher probability of collisions (i.e. repetitions within the N_s subkey values for large values of N_s) the rank could be expected to be below $(N_s + 1)/2$. However, as the number of S-boxes increases, the key-depedent algorithmic noise also increases and starts to dominate, implying that incorrect keys start to be ranked amongst the most likely ones in this case.

Next to the average guessing entropy, it is also insightful to look at the rank distribution after seeing all possible leakages. This is done by analyzing the device's leakage

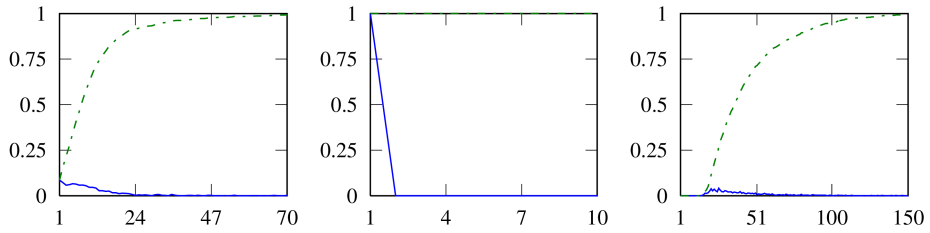

Figure 1.4: Rank distributions for carefully-chosen plaintexts with $N_s = 16$. Average rank of subkey k_1 (left), average minimum rank amongst all k_j's (middle) and average maximum rank amongst all k_j (right).

distribution that we denote as $\mathcal{D}$. For instance, given two S-boxes and two subkeys k_1 and k_2, the exact leakage distribution of such device can be computed as $\mathcal{D} = conv(\mathcal{D}^{k_1}, \mathcal{D}^{k_2})$ (using convolutions reduces the complexity of computing $\mathcal{D}$ for an 8-bit S-box from $2^{8 \cdot N_s}$ for the naive approach to $(8 \cdot N_s + 1)^2$). The outcome of this experiment for 1000 random keys can be seen in Fig. 1.4. The plots show the PMF (in solid blue) and the CDF (in dotted dashed green) for the rank distribution after seeing all possible traces for $N_s = 16$ with carefully chosen plaintexts. The x-axis corresponds to the key ranks and the y-axis corresponds to the probabilities. This figure confirms the previous observations with additional intuitions. First for the left plot, since the median is at rank ≈ 8 for each subkey in this case (as concluded in Fig. 1.3), an adversary would have a success rate of 0.5 to find the subkey within the ≈ 8 most likely candidates. Next, in the middle plot, we show the distribution of the minimum rank within the 16 subkeys k_j. It can be clearly seen that the subkey ranked first is almost surely one of the correct ones. This is an important observation and will allow us to construct an advanced attack in Section 1.6.1. As for the distribution of the maximum rank within the 16 subkeys k_j in the right plot, it can be seen that below rank 16, the success rate is almost zero since this can only happen (but is not given) if two subkeys are equal. Finally, in order to have a success rate of 0.5 to see the lowest ranked k_j (and therefore also seeing all other correct subkeys), the adversary would need to look at the first 37 most likely candidates.

1.4.2 Security based on unknown plaintexts

For the unknown plaintext scenario we targeted leakages in the form of Equation (1.4) and generated the templates as two-dimensional histograms. Each dimension has $8 \cdot N_s + 1$ bins, starting at bin $\mathcal{D}^i(0, 0)$ which indicates the probability that for key $k = i$, both leakage samples have a value of 0. The templates are built according to Algorithm 9.

The left side of Fig. 1.5 again shows a reference result for independent noise. Since, in the unknown plaintext scenario, we cannot decouple the noise by simply randomizing the plaintexts, we had to use a trick. Namely, we only fixed k_1 and randomly drew q different values for each k_j with $j \in 2, \ldots, N_s$. It can be seen that a recovery for $N_s = 1$ S-boxes

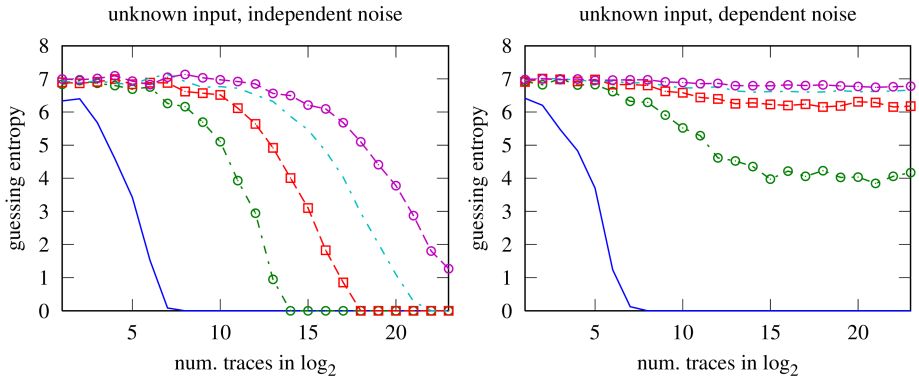

Figure 1.5: Average guessing entropy after attacks with unknown plaintexts for $N_s =$ (blue,s/), 2 (green,dd/c), 4 (red,d/s), 8 (cyan,dd/), 16 (magenta,d/c).

requires around 2^8 traces, whereas for $N_s = 16$ around 2^{27} traces can be expected.

The right side of the figure represents the unknown-plaintext scenario where the subkeys are constant over all traces within one instance of the experiment. It can be seen that dependent noise renders the model represented by the templates incorrect and therefore leads to a stagnation of the correct subkey's rank. This is similar to the carefully-chosen plaintext case and expected. However, the important difference compared to the previous experiment is that the stagnation does not take place at $y \approx \log_2((N_s + 1)/2)$ but much earlier. In order to get the full picture, we again look at the rank distributions in Fig. 1.6. First, we can observe in the left plot that the subkey ranks (from 40 000 experiments) look close to uniformly distributed (that would be reflected by a straight line), with a median rank at ≈ 102 (instead of 128 for the uniform distribution). For the minimum rank distribution (middle plot), the median rank is at ≈ 6, which has to be compared to a value of 10 that would be obtained for a uniform distribution with $N_s = 16$. As for the median of the maximum rank, it moved to ≈ 240 (whereas it would be at 245 for a uniform distribution with $N_s = 16$). In our experiments, the lowest maximum rank value found was 110. This essentially means that with a search complexity of $\binom{110}{16} \cdot 16! \approx 2^{107}$, the correct key is found with probability $\approx 1/40000 \approx 2^{-15}$. In fact, already for $N_s > 4$ and even when seeing all possible leakages in a noise free Hamming weight scenario, the guessing entropy is close to 7 and the rank distribution close to uniform.

1.4.3 Explaining the results: analysis of model errors

Both for the carefully-chosen plaintext scenario as well as for the unknown-input scenario, we are not able to perfectly model the leakage distribution without knowing the key. This is because, we have no means of marginalizing the distributions for the not-targeted subkeys as explained by Equations (1.2) and (1.3). That is, due to the key-dependent

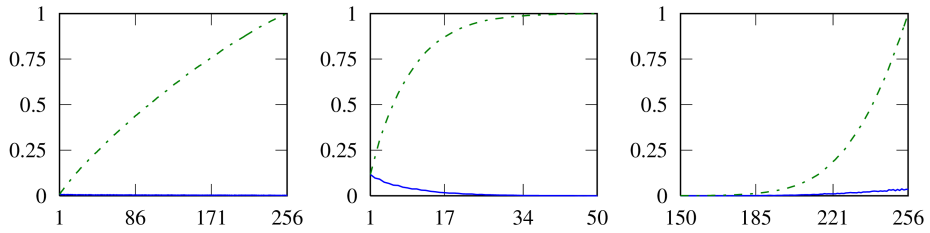

Figure 1.6: Rank distributions for unknown plaintexts with $N_s = 16$. Average rank of subkey k_1 (left), average minimum rank amongst all k_j's (middle) and average maximum rank amongst all k_j (right)

algorithmic-noise, we inevitably build incorrect templates. The question is, how bad our model errors become. For the carefully-chosen plaintext scenario, we saw that for small values of N_s (≤ 8), the average rank was at an optimum of $(N_s + 1)/2$. This suggests that the model errors were somewhat tolerable as illustrated in the left and middle plot of Figure 1.7. They show the distance between $\mathcal{D}^*$ and $\mathcal{D}^i$ for $N_s = 4$ and $N_s = 8$. For measuring the distance we used columns of the mutual information matrix as defined in [9] and in particular plot the column for the used key.[1] That is, the higher the value, the smaller is the distance. The metric was chosen because it directly reflects what will happen in a template attack: The key $\mathcal{D}^i$ which is closest to $\mathcal{D}^*$ will eventually be rated first. The distances between $\mathcal{D}^*$ and $\mathcal{D}^{k_j^*}$ are circled in red. In particular, it can be seen in the leftmost and middle plot for $N_s = 4$ and $N_s = 8$ that the distributions for the correct subkeys are closest. This does not hold anymore for $N_s = 16$ in the rightmost plot, only seven of the 16 correct subkeys are ranked first. Although these plots only show the effect for a specific set of subkeys, it already shows that the average rank has to be higher than $(N_s + 1)/2$.

In the unknown plaintext scenario, we need to estimate a second-order moment of a bi-variate distribution. From studies of masking, we know that such distributions are much more susceptible to noise [27]. Furthermore, in our case the relation between the leakage samples is not straightforward, similar as for affine or multiplicative masking [11]. Both circumstances suggest that the key dependent noise will cause more severe model errors and indeed this is what can be observed in Figure 1.8. Be aware that this time, the leftmost plot depicts the case for $N_s = 2$ and even there already non of the correct keys is ranked first. As we move to higher values for N_s, it can also be seen that the distances themselves become much smaller. As a consequence, measurement noise (remember that until now all experiments were performed without noise) and the inability to calculate the templates will make attacks even harder, as will be discussed in Section 1.5.3.

[1] Be aware that in our case the matrix has 2^8 rows and $2^{8 \cdot N_s}$ columns, however, we only plot the column for a single key. Would it also have $2^{8 \cdot N_s}$ rows, then we would have no model errors, but this would be equal to exhausting the full key.

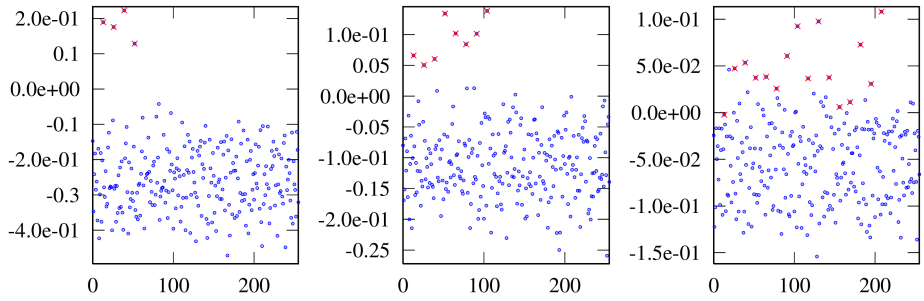

Figure 1.7: Distance between $\mathcal{D}^*$ and $\mathcal{D}^{k_j^*}$ for carefully-chosen plaintexts. The device holds the subkeys k_j^* (marked by the red x). As the distance is measured by the entries of the mutual information matrix, a higher value on the y-axis indicates a smaller distance. From left to right the scenarios for $N_s = 4, 8$, and 16 are depicted.

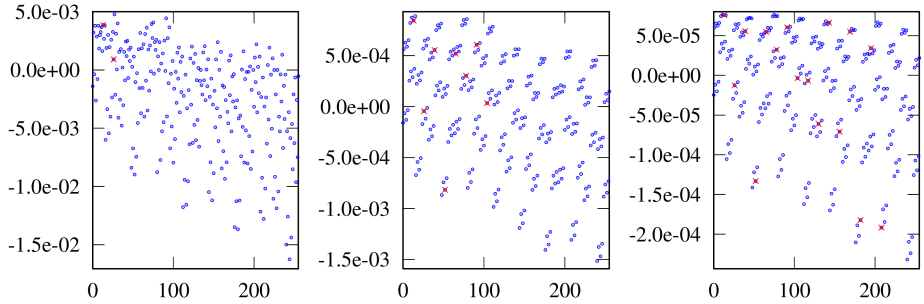

Figure 1.8: Distance between $\mathcal{D}^*$ and $\mathcal{D}^{k_j^*}$ for unknown plaintexts. From left to right the scenarios for $N_s = 2, 8$, and 16 are depicted.

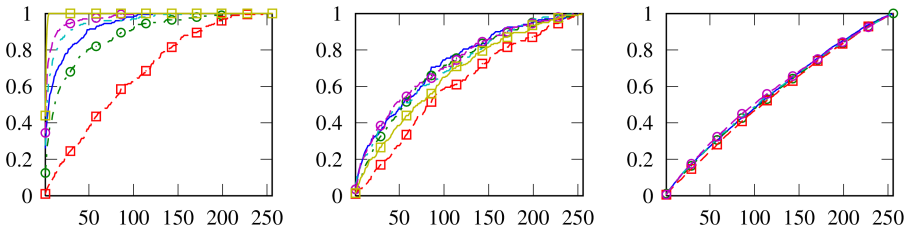

Figure 1.9: Comparison of different combinations of target functions and power models: AES+id (yellow,s/s), AES+quad (magenta,d/c), AES+nlhw (cyan,dd/), AES+hw (blue,s/), ID8+nlhw (green,dd/c), ID8+hw (red,d/s). From left to right the scenarios for $N_s = 2, 4$, and 16 are depicted.

1.5 Implementation and attack issues

Until now, we assumed bi-variate noise-free Hamming weight leakage and perfectly calculated templates. In this section we want to address the violation of these assumptions in a real world implementation and attack.

1.5.1 Deviations from the Hamming weight leakage function

All our previous simulations were based on the Hamming weight model. In this section we show that this model is appropriate and sufficient to argue about the security, even if it is not accurately met in a real world application. We do so by exploring different power models. In particular, we choose power models with low and high resolution and with low and high non-linearity. As for the resolution we choose the leakage functions as the Hamming weight function (hw), as the Hamming weight function plus quadratic terms (quad), and as the identity function (id). As for the non-linear leakage function we chose the Hamming weight function preceeded by an AES S-box (nlhw). In addition, we target two kinds of S-boxes, the AES S-box (AES) and an identity function S-box (ID8). The latter one would correspond to directly attacking the key addition layer of AES. In Figure 1.9 we compare the rank distributions for various scenarios for $N_s = 2, 4$ and 16. For $N_s = 2$ it can be seen that the non-linearity of the target function is of higher importance than the one of the leakage function. The non-linearity of the leakage function only helps significantly for linear target functions (ID8), for (AES) the impact is minor, yet it helps improving slightly. Furthermore, for low noise, the resolution of the leakage function is of great importance, but is as noise prone as the Hamming weight function. Especially, for high algorithmic noise scenarios, almost no difference can be seen. Finally, it is important to note, that the S-box is indeed the best target function to show the side-channel resistance of the proposed scheme. Note, that due to the computational complexity ($(2^8 \cdot N_s)^2$ bins per template), the id leakage function was omitted for $N_s = 16$, yet, already for $N_s = 4$ it performs worse than the Hamming weight leakage function.

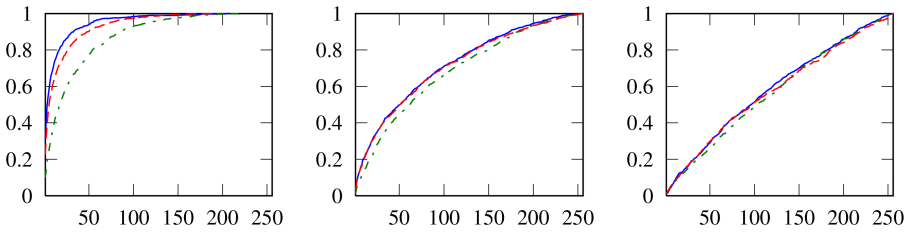

Figure 1.10: Comparison of the rank distribution when attacking a standard bi-variate (red,d), a Hamming distance-based (green,dd) and a normalized product combining (blue,s) based leakage distribution. From left to right the scenarios for $N_s = 2, 4$, and 16 are depicted.

1.5.2 Distance-based leakages

In practice, a cryptographic implementation can be flawed because an adversary sees leakages which are not covered by the theoretical analysis. This can be due to glitches, early propagation, or most deadly for Boolean masking, unintentional distance-based leakage. That is, a secret shared as $(s \oplus m, m)$ leaks via $HD(s \oplus m, m)$. Such leakage can occur if a register holding the first share is overwritten with the second share. Another scenario, where the adversary might get an advantage is if he can perform a normalized product combining before summing up the leakage points. This can be the case for a weakly shuffled software implementation which handles the key addition and the S-box operation together. Interestingly, none of these implementation issues represent a threat in the unknown-input case. From Figure 1.10 we see that the Hamming distance case already performs badly for small values of N_s, whereas the normalized product combining still gives a slight advantage due to the reduced noise impact. Yet, the higher the value of N_s becomes, the more forgiving the scheme becomes w.r.t. implementation weaknesses.

1.5.3 Bounded template estimation

In practice, there is another source of model errors, namely poor template estimation. Usually, one exhaustively acquires traces for all inputs. In practice, this is not possible as the number of inputs grows exponentially with N_s, but usually good enough if the number of traces is sufficiently large.[2] In Figure 1.11 we can see that this is not the case for unknown inputs. We compare the rank distribution for an attack with dependent noise to an attack with independent noise but with insufficiently sampled templates. For the left plot with $N_s = 2$, 2^{26} traces for template building yield a smaller error than key dependent noise, but still do not allow to recover the key with certainty as in the left

[2]One could overcome this insufficiency by building the templates for the S-boxes independently and afterwards combine them like we did in our simulations. However, the errors for the $\mathcal{D}^i$s will multiply when calculating the overall template and therefore the overall error will grow exponentially with N_s.

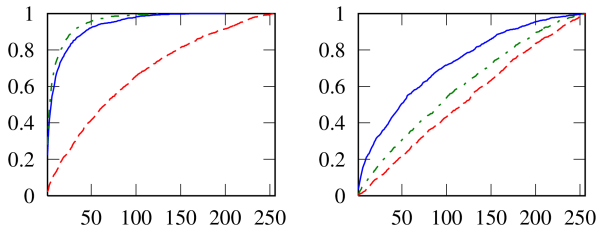

Figure 1.11: Rank distribution for calculated templates with dependent noise (blue,s) and estimated templates with independent noise. For the dotted dashed green line the templates were estimated using 2^{26} traces, for the dashed red one using 2^{22} traces. $N_s = 2$ in the left plot and $N_s = 4$ in the right plot.

plot of Figure 1.5 where the templates where calculated. Using only 2^{22} traces already leads to a larger model error than dependent noise. Finally, for $N_s = 4$ the calculated templates for dependent noise perform already best.

1.6 Alternative attack paths

1.6.1 Advanced attacks and key verification

In [20], an iterative attack was described which allows to recover the 16 subkeys up to their order by successively removing the dependent noise in an iterative DPA. In this attack the authors exploited the fact that the first ranked key was always one of the correct ones and thus could be used to model the key dependent noise in the next iteration. Thus virtually, the parameter N_s was reduced by one in each iteration.

In the unknown input case we could follow a similar strategy, just that we would need to construct the templates freshly in every iteration. On top of that we cannot just take the first subkey candidate but exhaust the lists up to a certain threshold.[3] To estimate the effort of this, we multiply the medians of the ranks for the best ranked k_j^* for $N_s = 1 \dots 16$. The result is that with a probability of 2^{-16} we recover the correct key set after $\approx 2^{27}$ iterations. Each iteration comprises 16 template building and attack operations which in turn has a complexity of $\approx 2^{28}$(at least 2^{20} traces and 2^8 keys) each. Thus, investing around 2^{60} one can recover the subkey bytes up to permutation. Ordering them costs again $16! \approx 2^{44}$. Finally, after going through all this effort, one still has no means of verifying whether the correct key was found as one needs at least one secret plaintext to verify the key based on a known answer. As recovering a plaintext is as hard as recovering a key and both need to be jointly verified, the effort squares.

[3] Be aware that key enumeration algorithms do not work here since the lists are dependent and thus no full key sorting according to probabilities is possible.

1.6.2 Attacks on the plaintexts

Attacks on the key are restricted to N_p traces in practice. As N_p needs to be precomputed, in practice it will take values between 2^4 and 2^{16}. Although an adversary cannot launch a meaningful attack on the key with this restriction, he could as well target the plaintexts (by fixing the plaintext for the last iteration of the tree, hence randomizing the key for this iteration, and thus switching the role of the key and the plaintext in the attack). Having sufficient plaintexts, a standard DPA on the key could be mounted.

Even with unlimited traces, one is far from recovering a key or a plaintext. Thus, for the standard DPA, the plaintext bytes have to be guessed. Let us first assume, that only one subkey byte is targeted. The adversary then needs to pick the plaintext byte for each trace from a set. Without side-channel information, this set would have a size of 256. From Figure 1.6 we know that with a 50% probability, the plaintext byte is contained in a set of 240 entries after an unbounded attack. Thus, overall in an attack where r plaintexts are used, $256 \cdot 240^r$ hypotheses have to be built. Even then, the probability that the correct plaintext bytes are contained is 2^{-r}. Therefore, this seems to be a rather futile attack path.

1.7 Conclusions

In this work we presented a leakage resilient PRF which makes use of parallel block cipher implementations with unknown inputs. To the best of our knowledge this is the first work to study and exploit this form of key dependent algorithmic noise. It turns out that it renders the problem of side-channel key recovery intractable, even in a noise free setting and independent of the number of traces and the used power model.

Thanks to this security improvement over the CHES 2012 construction, standardized algorithms like AES can be used in our construction. Moreover, the analysis suggests that even localized EM attacks can be tolerated to some degree. That is, even if an EM probe would only catch the signal of 8 or 4 S-boxes, the attack would not suddenly become trivial. On top of that, we showed that opposed to the previous construction, the strong side-channel resistance holds throughout the entire algorithm and not only for the first round's S-box layer.

We also showed that in practice, the inability to build perfect templates due to a limitation in the number of traces and electrical noise, will lead to much worse results than in our analysis. Finally, we also investigated the impact of effects like unintentional Hamming distance leakage and multiplicative leakage effects and could confirm that they only lead to minor advantages or even make things worse.

From a performance point of view it allows to use larger values for N_p than the size of the S-box. In practice, it will be very application specific whether large values for N_p pay off. However, for block ciphers which use small S-boxes, like e.g. PRESENT, the construction can definitely lead to a performance increase over the CHES 2012 one.

Chapter 2

Algorithms and Cryptographic services

In this chapter we will describe the algorithms which have to be used to implement LRP and the cryptographic services using LRP.

2.1 Notation

In the following we will use the function $len(x, 8)$ to get the byte-length of the operand x and more generally $len(x, m) := \lceil log_{2^m}(x) \rceil$.

2.2 Key generation

The key generation is optional and can be used to ensure that a key can generate 2^m distinct plaintexts using Algorithm 2. In practice it has to be judged whether the key generation is really needed or whether the probability of a random key fulfilling this requirement is high enough. In our setup, the probability is approximately 2^{-120}. The check can be done in a non-leaking way by checking that the keystream for 2^m blocks is distinct. Likewise for all keys it can be checked that the keystreams for a constant counter are the same.

The checksum generated at the end of this function is not defined in this document as it can be application or platform specific. However, in order to counteract fault attacks, it must be verified after running Algorithms 2 and 3 on a static key for which a checksum is available.

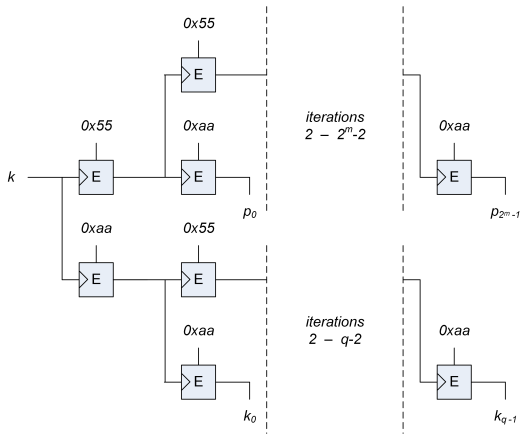

Figure 2.1: Generation of secret plaintexts and updated keys.

Algorithm 1 $\{k, cs\} = \text{keyGen}(m)$

Require: Parameter m

Ensure: Key k which produces 2^m distinct plaintexts and q different updated keys under Algorithms 2 and 3.

repeat

$k \leftarrow \mathbb{F}_{2^n}$

$\{p_0, \ldots, p_{2^m-1}\} = \text{generatePlaintexts}(m,\ k)$

$\{k_0, \ldots, k_{q-1}\} = \text{generateUpdatedKeys}(q,\ k)$

until $\text{isPairwiseDistinct}(\{p_0, \ldots, p_{2^m-1}\}) \wedge \text{isPairwiseDistinct}(\{k_0, \ldots, k_{q-1}\})$

return $\{k,\ \text{checksum}(p_{2^m-1},\ k_{q-1})\ \}$

2.3 Plaintext and key generation

As can be seen in Figure 2.1, the generation of the plaintexts and the updated keys deviates from the construction in Section 1.3. This design choice was made for two reasons. First it allows to generate q updated keys for different purposes, as it is often the case for session keys. Second, if q is small these keys could also be generated on the fly from k without much performance penalty. That is, theoretically, one would only need to store one session key root key k from which everything can be generated when needed.

2.3.1 Plaintext generation

During the plaintext generation we choose the vectors $\{\texttt{0x55}\}^{16}$ and $\{\texttt{0xaa}\}^{16}$ in order to have different leakages for all S-boxes.

Algorithm 2 $\{p_0, \ldots, p_{2^m-1}\}$ = generatePlaintexts(m, k)

Require: Parameter m, key k.
Ensure: Plaintexts $\{p_0, \ldots, p_{2^m-1}\}$.
  $h = k$
  $h = E_h(\{\texttt{0x55}\}^{n/8})$
  for $i = 0 \ldots 2^m - 1$ **do**
    $p_i = E_h(\{\texttt{0xaa}\}^{n/8})$
    $h = E_h(\{\texttt{0x55}\}^{n/8})$
  end for
  return $\{p_0, \ldots, p_{2^m-1}\}$

2.4 Updated key generation

The generation of the updated keys is analogous to the secret plaintext generation with the only difference being derivation of the inital key h.

Algorithm 3 $\{k_0, \ldots, k_{q-1}\}$ = generateUpdatedKeys(q, k)

Require: Parameter q, key k.
Ensure: Updated keys $\{k_0, \ldots, k_{q-1}\}$.
  $h = k$
  $h = E_h(\{\texttt{0xaa}\}^{n/8})$
  for $i = 0 \ldots q - 1$ **do**
    $k_i = E_h(\{\texttt{0xaa}\}^{n/8})$
    $h = E_h(\{\texttt{0x55}\}^{n/8})$
  end for
  return $\{k_0, \ldots, k_{q-1}\}$

2.5 Get plaintext

Simply returns the correct plaintext.

Algorithm 4 p = getPlaintext(c, $\{p_0, \ldots, p_{2^m-1}\}$)

Require: an input chunk $c \in \mathbb{F}_{2^m}$, a set of plaintexts $\{p_0, \ldots, p_{2^m-1}\}$.
Ensure: A plaintext $p \in \mathbb{F}_{2^n}$
  return p_c

2.6 Function evaluation

The final transformation should always be checked for fault attacks by verifying that the decryption of the result equals 0^n. However, since this is a pure functional description, this detail was left out here.

Algorithm 5 $y = \text{evalLRP}(m, \{\{p_0, \ldots, p_{2^m-1}\}, k'\}, l, x, final)$

Require: The dimension m, a set of plaintexts $\{p_0, \ldots, p_{2^m-1}\}$, an updated key k', the length $l = len(x, m)$, the input $x = x_0, \cdots, x_{l-1}$ with $x_i \in \mathbb{F}_{2^m}$, an indicator for the final transformation $final$.
Ensure: Result of (partial) LRP evaluation y

```
y = k'
for i = 0 to l − 1 do
    p = getPlaintext(x_i, {p_0, ..., p_{2^m−1}});
    y = E_y(p)
end for
if final = true then
    t = y
    y = E_y(0^n)
end if
return y
```

2.7 Encryption

Encryption is done similar as Construction 3.30 from [16]. However, instead of

$$c := \langle r, F_k(r) \oplus m \rangle ,$$

where $F_k(r)$ is the family of pseudo-random functions indexed by key k, we perform

$$c := \left\langle r, E_{F_k(r)}(m) \right\rangle .$$

If $F_k(r)$ is instantiated with the previously proposed LRP, we refer to this mode of operation as **leakage resilient indexed codebook (LRICB)**. Decryption is straightforwardly defined as

$$m := \left\langle r, D_{F_k(r)}(c) \right\rangle .$$

To keep the overhead of transporting r to a minimum, encryption is done in counter mode. Running encryption in counter mode also has the advantage that for a w-block output $c_1, ..., c_w$, $r_1, ..., r_w$ have up to $w' = len(n, m) - len(w, m)$ equal most significant chunks. This means that the tree of $F_k(r)$ can in the first step only be evaluated until depth w' which takes w' encryptions. Then for every r_i, the remaining evaluation effort is $len(w, m) + 1$.Therefore, the effort for encrypting w blocks can be stated

as $w' + (len(w, m) + 3) * w$ encryptions. Note, that the value 3 accounts for the ECB encryption/decryption with a fault protective backwards calculation. To allow such optimizations the counter is assumed to be in big-endian as it is processed from left to right by evalLRP. However, the optimizations themselves are not included in the depicted algorithm.

2.7.1 Background

For counter mode encryption it is possible to mount a DPA in case the plaintext is constant. In this scenario the plaintext acts as the key one wishes to recover. In particular, we assume that the key stream (ks) changes on every encryption but the plaintext (pt) you encrypt is constant. Encryption works as ct = pt xor ks. The adversary sees the ciphertext (ct) and can thus build his hypotheses for the key stream as hypo = ct xor pt_guess. If one of his hypothesis correlates with the key stream, he found the plaintext. To counteract this, we decided to go for Algorithm 6 instead of pure counter mode encryption. Note, that LRICB relies on full blocks again and therefore potentially needs padding.

Algorithm 6 LRICB encryption: $\{r, ct\}$ = LRICBEnc(m, $\{\{p_0, \ldots, p_{2^m-1}\}, k'\}$, w, r, pt, pad)

Require: Parameter m, $\{p_0, \ldots, p_{2^m-1}\}$, an updated key k', the length $l = len(r, m)$, the least significant l digits of the counter $r = r_{l-1}, \cdots, r_0$ with $r_i \in \mathbb{F}_{2^m}$, the plaintext pt, an indicator whether 0x80 padding is used pad.
Ensure: The ciphertext $ct = ct_0 \ldots ct_{w-1}$ and an updated counter r.
  if $pad = 1$ **then**
    Apply padding by adding a 0x80 byte and filling up the block with 0x00
  else if $len(pt, 8) \mod 16 \neq 0$ **then**
    return FAIL
  end if
  $w = len(pt, 128)$
  for $i = 0 \ldots w - 1$ **do**
    y = evalLRP(m, $\{\{p_0, \ldots, p_{2^m-1}\}, k'\}$, l, r, $true$)
    $ct_i = E_y(pt_i)$
    $r = r + 1$
  end for
  return $\{r, ct\}$

2.7.2 Important implementation notes

For encryption, the key used in the ECB stage is unique as the counter needs to be unique to provide IND-CPA. However, if an adversary can decrypt, he will be able to keep the counter constant and choose the ciphertext to decrypt. This would create a DPA scenario where the adversary can recover the key (used in the ECB stage) associated with a given

counter value. To counteract this attack, any ciphertext needs to be authenticated before decryption using a leakage resilient MAC as presented in Section 2.8.

2.7.3 Note with respect to repeated counter values and fault attacks

Whenever a counter value and a message (plaintext or ciphertext) can be repeated and the output is seen, LRICB could be subject to DFA.In practice, this rarely applies. This is because for encryption, the counter has to change. For decryption, it would only apply, if the result of a decryption is actually seen by the adversary. This neither is the case for secure messaging, where no plaintexts are transmitted nor for encrypted memory, where there would be no need for encrypting the memory if the plaintext would afterwards be transmitted.

2.8 CBC-MAC and CMAC

In general, CBC-MAC and CMAC can be constructed using a PRF as given in Construction 4.11 from [16]. The mentioned construction shows only CBC-MAC which is only secure for a fixed length (per key). In order to obtain a variable length construction from CBC-MAC we either need to perform a prefix-free encoding which increases the number of blocks to process, or a construction like CMAC is used. CMAC can be seen as an improvement of CBC-MAC where the last block is padded and xored with one out of two keys depending on whether the last block is a full one.

The CMAC keys K_1 and K_2 are generated as follows:

$$K_0 = \text{evalLRP}(m, \{\{p_0, \ldots, p_{2^m-1}\}, k'\}, l, 0^n, final)$$

$$K_1 = K_0 \cdot x \mod x^{128} + x^7 + x^2 + x + 1$$

$$K_2 = K_0 \cdot x^2 \mod x^{128} + x^7 + x^2 + x + 1$$

Algorithm 7 $y = \text{CMAC\_LRP}(m, \{\{p_0, \ldots, p_{2^m-1}\}, k', K_1, K_2\}, l, x)$

Require: Parameter m, the plaintexts $\{p_0, \ldots, p_{2^m-1}\}$, an updated key k', the byte length $l = len(x, 8)$, the input $x = x_0, \cdots, x_{l-1}$, the CMAC keys K_1 and K_2, the block size $BS = len(K_1, 8)$.

Ensure: The CMAC result y.

$y = 0^n$
$i = 0$
while $l > BS$ **do**
  $y = y \oplus x_i \ldots x_{i+BS-1}$
  $y = \text{evalLRP}(m, \{\{p_0, \ldots, p_{2^m-1}\}, k'\}, y, true)$
  $l = l - BS$
  $i = i + BS$
end while
$y = y \oplus x_i \ldots x_{i+l-1} 0^{BS-l}$
if $l = BS$ **then**
  $y = y \oplus K_1$
else
  $x_{i+l} = \texttt{0x80}$
  $y = y \oplus K_2$
end if
$y = \text{evalLRP}(m, \{\{p_0, \ldots, p_{2^m-1}\}, k'\}, y, true)$
return y

Bibliography

[1] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Joye and Moradi [15], pages 64–81.

[2] Sonia Belaïd, Vincent Grosso, and François-Xavier Standaert. Masking and leakage-resilient primitives: One, the other(s) or both? *Cryptography and Communications*, 7(1):163–184, 2015.

[3] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient prfs: cipher design principles and analysis. *J. Cryptographic Engineering*, 4(3):157–171, 2014.

[4] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[5] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient AES threshold implementation. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014.

[6] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In 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*, pages 398–412. Springer, 1999.

[7] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.

[8] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.

[9] Alexandre Duc, Sebastian Faust, and Franois-Xavier Standaert. Making masking security proofs concrete or how to evaluate the security of any leaking device (extended version). Cryptology ePrint Archive, Report 2015/119, 2015. `http://eprint.iacr.org/`.

[10] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Prouff and Schaumont [23], pages 213–232.

[11] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In *Proceedings of the 17th International Conference on Selected Areas in Cryptography*, SAC'10, pages 262–280, Berlin, Heidelberg, 2011. Springer-Verlag.

[12] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[13] Vincent Grosso, Romain Poussier, François-Xavier Standaert, and Lubos Gaspar. Combining leakage-resilient prfs and shuffling - towards bounded security for small embedded devices. In Joye and Moradi [15], pages 122–136.

[14] 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.

[15] Marc Joye and Amir Moradi, editors. *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*. Springer, 2015.

[16] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2014.

[17] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[18] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In 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*, pages 157–171. Springer, 2005.

[19] Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 243–261. Springer, 2014.

[20] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient prfs. In Prouff and Schaumont [23], pages 193–212.

[21] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In 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*, pages 69–88. Springer, 2011.

[22] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[23] 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.

[24] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[25] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[26] 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 - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[27] 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, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

[28] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.

[29] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.