SCARE of Secret Ciphers with SPN Structures*

Matthieu Rivain[1] and Thomas Roche[2]

[1] CryptoExperts, France
matthieu.rivain@cryptoexperts.com

[2] ANSSI, Fance
thomas.roche@ssi.gouv.fr

Abstract. Side-Channel Analysis (SCA) is commonly used to recover secret keys involved in the implementation of publicly known cryptographic algorithms. On the other hand, Side-Channel Analysis for Reverse Engineering (SCARE) considers an adversary who aims at recovering the secret design of some cryptographic algorithm from its implementation. Most of previously published SCARE attacks enable the recovery of some secret parts of a cipher design –*e.g.* the substitution box(es)– assuming that the rest of the cipher is known. Moreover, these attacks are often based on idealized leakage assumption where the adversary recovers noise-free side-channel information. In this paper, we address these limitations and describe a generic SCARE attack that can recover the full secret design of any iterated block cipher with common structure. Specifically we consider the family of Substitution-Permutation Networks with either a classical structure (as the AES) or with a Feistel structure. Based on a simple and usual assumption on the side-channel leakage we show how to recover all parts of the design of such ciphers. We then relax our assumption and describe a practical SCARE attack that deals with noisy side-channel leakages.

1 Introduction

Side-Channel Analysis for Reverse Engineering (SCARE) refers to a set of techniques that exploit side-channel information to recover secret algorithms, software, or hardware designs. One of the main application of SCARE is the recovery of symmetric ciphering algorithms of private design, as often used in Pay-TV and GSM authentication protocols. The first SCARE attack in this context was introduced by Novak [26], who showed how to recover one out of two s-boxes from a secret instance of A3/A8 algorithm (used in GSM protocol). This work was subsequently improved by Clavier [11] who described how to recover both s-boxes altogether with the secret key used by the cipher. In parallel to these results, Daudigny *et al.* [14] showed that simple secret modifications of the DES cipher could also be recovered from side-channel observations. In a more recent work, Réal *et al.* [28] took a closer look at Feistel schemes in a more general sense. They showed how an adversary that gets the Hamming weight of some intermediate result can interpolate the round function of the cipher. Eventually, a SCARE attack on stream ciphers was proposed by Guilley *et al.* [20]. They showed how to retrieve the overall design when either the linear or the nonlinear part of the cipher is known.

Our Contribution. In this paper, we introduce a SCARE attack that recovers the full secret design of an iterated Substitution-Permutation Network (SPN for short), namely an

* A short version of this paper appears in the proceedings of ASIACRYPT 2013.

iterated cipher composed of substitution boxes (or s-boxes), linear layers and key additions. As in [11, 26], our attack is based on the simple assumption that the side-channel leakage enables the detection of colliding s-box computations. Specifically, the attacker is able to select strips of side-channel traces where the s-box computations are located and decide on collisions between the processed values from the observation of these traces. This assumption has been the basis of various previously published side-channel key-recovery attacks (see for instance [3–7, 12, 18, 25, 31, 32]). We first show how a perfect detection of colliding s-box computations enables an efficient recovery of a secret cipher with *classical* SPN structure as the one of the AES [15]. Roughly speaking, the collision detection mechanism allows us to build simple linear equation systems involving the different unknowns of the cipher algorithm (*i.e.* the s-box values, the linear layer coefficients, the secret round key coordinates). We then extend our basic attack to relax as much as possible the constraints on the design, allowing several different s-boxes, binary linear layers, and Feistel structures, in order to cover a wide spectrum of usual block cipher designs. In the second part of this paper, we address the practical aspects of our attack and relax the perfect detection assumption. We describe a practical SCARE attack working in the presence of noise in the side-channel leakage and we provide experimental results showing its practicability.

Related Work. In a recent independent work [13], Clavier *et al.* present a SCARE attack against AES-like block ciphers. The authors consider a chosen-plaintext and known-ciphertext scenario with perfect detection of colliding s-boxes. Under these assumptions, they show how to efficiently recover the secret parameters of a modified AES. They further address the case of protected implementations with common software countermeasures against side-channel attacks. In comparison, our attack targets a wider class of SPN ciphers, including modified AES ciphers as a particular case. Moreover, we extend our attack to deal with noisy leakages, hence relaxing the perfect detection assumption. However, we do not deal with the case of protected implementations (though we give a few insights about it in Section 8).

Paper Organization. In the first section we describe the families of target algorithms: the classical SPN and Feistel structures. Then we present our basic SCARE attack on classical SPN ciphers in Section 3, followed by the attack extension to more general SPN structures in Section 4 and to Feistel schemes in Section 5. The practical SCARE attack dealing with noisy leakages is described in Section 6, and experimental results are presented in Section 7. Finally, we give some discussions and perspectives in Section 8.

2 Substitution-Permutation Networks

We consider a block cipher E computing an ℓ-bit ciphertext block c from an ℓ-bit plaintext block p through the repetition of a key-dependent permutation ρ, called *round function*. Each round is parameterized by a different round key k_i derived from the secret key k through a key scheduling process. Let r denote the number of rounds, the ciphertext block is then

defined as

$$c = E_k(p) = \rho_{k_r} \circ \rho_{k_{r-1}} \circ \cdots \circ \rho_{k_1}(p) \ .$$

In an SPN block cipher, the round function is composed of linear permutations and nonlinear substitutions, and the key is introduced by addition. The addition and linearity are considered over the vectorial space $\mathbb{F}_2^\ell$. Namely round keys are introduced by a simple exclusive-or (xor), and linear permutations are homomorphic with respect to the xor operation. Non-linear substitutions are applied on small blocks of bits which are replaced by new blocks looked-up from a predefined table usually called *s-box* (for substitution-box). In what we shall call a *classical SPN structure*, the different s-boxes and linear transformations are bijective (*e.g.* the Advanced Encryption Standard [15]). In the presence of non-bijective transformations, it is common to use a so-called *Feistel scheme* in order to make the round function, and hence the overall cipher, invertible (*e.g.* the Data Encryption Standard [16]). We present the two different structures hereafter.

2.1 Classical SPN Structure

In a classical SPN structure, the plaintext is considered as a n-dimensional vector of m-bit coordinates: $p = (p_1, p_2, \ldots, p_n)$, with $\ell = nm$. The round function is composed of a key addition layer σ_{k_i}, a nonlinear layer γ, and a linear layer λ, that is

$$\rho_{k_i} = \lambda \circ \gamma \circ \sigma_{k_i} \ .$$

The key addition layer is a simple xor-ing of the round key:

$$\sigma_k(p) = p \oplus k \ .$$

The nonlinear layer consists of the parallel application of an $m \times m$ s-box S:

$$\gamma(p) = (S(p_1), S(p_2), \ldots, S(p_n)) \ ,$$

And the linear layer is a linear transformation over $(\mathbb{F}_{2^m})^n$:

$$\lambda(p) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \tag{1}$$

where the a_{ij} and the p_j are considered as elements of $\mathbb{F}_{2^m}$.

Remark 1. The final round sometimes skips the linear layer and an additional key addition is often performed after the final nonlinear layer. The attack described in this paper works as well for these variants.

Generalizations. The attack described in this paper is extended to deal with more general SPN structures. In particular, the s-box may not be the same for each subpart of the state and the linear transformations may be defined at the binary level *i.e.* over $\mathbb{F}_2$ rather than over $\mathbb{F}_{2^m}$. Namely, we consider the two following variants of the previous structures:

- **Multiple s-boxes setting** – In this variant, the nonlinear layer γ is defined as

$$\gamma(p) = (S_1(p_1), S_2(p_2), \ldots, S_n(p_n)) \ ,$$

  where the S_i's are different bijective nonlinear functions defined over $\{0,1\}^m$.

- **Binary linear layer setting** – In this variant, λ is defined as a linear transformation over $\mathbb{F}_2$. Namely the input state is considered as a vector over $(\mathbb{F}_2)^\ell$ and is multiplied by an $\ell \times \ell$ binary matrix. An interesting particular case of binary linear transformation is the bit-permutation which is often used to get compact hardware implementations (see for instance the block cipher PRESENT [8]).

2.2 Feistel Structure

In a Feistel scheme, the encrypted block is divided in two parts p and q of equal bit-length $\ell/2$. The round function is then defined as

$$\rho_{k_i} : (p, q) \mapsto (q, p \oplus f_{k_i}(q)) \ ,$$

where f is a function from $\ell/2$ bits to $\ell/2$ bits called the *Feistel function*. The Feistel function can take many forms; we will assume here that it is composed of a key addition followed by a nonlinear layer both surrounded by linear layers. Namely it is defined as

$$f_{k_i} = \lambda_2 \circ \gamma \circ \sigma_{k_i} \circ \lambda_1 \ .$$

Note that f is not necessarily invertible since the overall invertibility of the cipher comes from the Feistel structure.

The linear layers λ_1 and λ_2 are linear transformations from $(\mathbb{F}_{2^m})^n$ to $(\mathbb{F}_{2^m})^s$ and from $(\mathbb{F}_{2^m})^s$ to $(\mathbb{F}_{2^m})^n$ respectively, for some $s \geq n$. That is we have

$$\lambda_1(q) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s,1} & a_{s,2} & \cdots & a_{s,n} \end{pmatrix} \cdot \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \quad \text{and} \quad \lambda_2(q) = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,s} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,s} \end{pmatrix} \cdot \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_t \end{pmatrix} \qquad (2)$$

where the a_{ij}, b_{ij} and the q_j are considered as elements of $\mathbb{F}_{2^m}$. These functions are usually called the *linear extension* and *linear compression* transformations when $s > n$. Although not mandatory from a functional point of view, these functions are usually taken of full rank for security reasons.

As for the classical SPN structure, the key addition layer is a simple xor-ing of the round key (of size $s \times m$ bits):

$$\sigma_k(q) = q \oplus k \ .$$

And the nonlinear layer consists of s parallel applications of an $m \times m$ s-box S, that is

$$\gamma(q) = (S(q_1), S(q_2), \ldots, S(q_s)) \ .$$

It is common for Feistel schemes to use non-injective $m \times m'$ s-boxes where $m' < m$. Such s-boxes can be seen as $m \times m$ s-boxes by padding their ouputs with arbitrary bits which are simply discarded by the subsequent linear layer. So there is no loss of generality in only considering $m \times m$ s-boxes.

3 Basic SCARE of Classical SPN Structures

3.1 Attacker Model

We present a generic SCARE attack in a known-plaintext scenario, and we show how its complexity can be lowered in a chosen-plaintext scenario. Our attack does not require the knowledge of the ciphertext but only exploits the side-channel leakage of the cipher execution. Moreover, it is assumed that *colliding s-box computations can be detected from the side-channel leakage.* Specifically, we assume that the attacker is able to

(i) identify the s-box computations in the side-channel leakage trace and extract the leakage corresponding to each s-box computation,

(ii) decide whether two s-box computations $y_1 \leftarrow S(x_1)$ and $y_2 \leftarrow S(x_2)$ are such that $x_1 = x_2$ or not from their respective leakages.

Remark 2. This assumption implicitly means that the cipher implementation processes the s-box computations in a sequential way and that two s-box computations of the same input at two different points in the execution produce identical side-channel leakages. These constraints are further discussed in Section 8.

Under the above assumption, the attacker can identify r different groups of n s-box computations, and hence recover the number r of rounds, the number n of s-boxes per round and hence the s-box size $m = \ell/n$, where ℓ is the block size. We will therefore assume these parameters to be known in our attack description.

In what follows, we first show how the above assumption enables the complete recovery of a secret cipher with SPN structure as described in Section 2.1. In Section 6, we relax this assumption and extend our attack to deal with noisy leakages which can lead to decision errors in the collision detections.

3.2 Equivalent Representations

Several equivalent representations are possible for an SPN cipher such as considered here. For instance one can change the s-box S for the s-box S' defined as

$$S'(x) = S(x \oplus \delta)$$

for some $\delta \in \mathbb{F}_{2^m}$, and replace every round key $k_i = (k_{i,1}, k_{i,2}, \ldots, k_{i,n})$ by

$$k'_i = (k_{i,1} \oplus \delta, k_{i,2} \oplus \delta, \ldots, k_{i,n} \oplus \delta) \ .$$

The two representations are clearly equivalent in a functional sense. Moreover, the ability of detecting collisions in s-box computations does not make it possible to distinguish between two different equivalent representations.

Another way to obtain equivalent representations is by changing the s-box S for the s-box S' defined as

$$S'(x) = \alpha \cdot S(x)$$

for some $\alpha \in \mathbb{F}^*_{2^m}$, and by replacing the linear layer λ defined in (1) by the linear layer λ' obtained from the matrix $(a'_{i,j})_{i,j}$ whose coefficients satisfy

$$a'_{i,j} = \frac{a_{i,j}}{\alpha}$$

for every i and j.

In our attack, we fix the first round key coordinate $k_{1,1}$ to 0 and we fix the coefficient $a_{1,1}$ to 1, which is equivalent to fixing the variables δ and α. Note that $a_{1,1}$ may equal 0 (which is revealed by the attack), in which case we try fixing $a_{1,2}$, then $a_{1,3}$, and so on. We describe hereafter the successive stages of the attack.

3.3 Stage 1: Recovering k_1

Since we have fixed $k_{1,1} = 0$, we aim to recover the $n-1$ remaining subkeys $k_{1,2}$, $k_{1,3}$, …, $k_{1,n}$. Let $\mathcal{I}$ denote the set of indices i for which $k_{1,i}$ is known. At the beginning of the attack $\mathcal{I} = \{1\}$. Then for any collision between two s-box computations $y_i \leftarrow S(p_i \oplus k_{1,i})$ and $y_j \leftarrow S(p_j \oplus k_{1,j})$ for some $i \in \mathcal{I}$ and $j \notin \mathcal{I}$, one deduces

$$k_{1,j} = p_j \oplus p_i \oplus k_{1,i} \ ,$$

and the index j is added to $\mathcal{I}$. We expect to retrieve all subkeys with less than $2^{m/2}$ encryptions.

3.4 Stage 2: Recovering λ, S and k_2

Once k_1 has been recovered, one knows the inputs of the s-box in the first round. Let us define $x_i = S(i)$ for every $i \in \{0, 1, \ldots, 2^m - 1\}$, so that recovering the s-box means recovering the 2^m unknowns x_0, x_1, …, x_{2^m-1}. The attack consists in constructing a set of equations

in the x_i's, the $a_{i,j}$'s and the $k_{2,i}$'s. Solving the obtained system hence amounts to recover λ, S and k_2.

The first step of this stage consists in collecting the leakages ℓ_β from s-box computations $\mu \leftarrow S(\beta)$ for every $\beta \in \mathbb{F}_{2^m}$. We shall denote by $\mathcal{B}$ the obtained leakage basis $\{\ell_\beta \mid \beta \in \mathbb{F}_{2^m}\}$. Such a basis can be constructed since k_1 is known from the first stage, hence the inputs of the s-box computations in the first round are known. This basis is then used to detect collisions between s-box computations in the second round and s-box computations $\mu \leftarrow S(\beta)$. Let w_i be the ith s-box input before key addition in the second round (*i.e.* w_i is the ith m-bit output of the first round), in the encryption of some plaintext p. Then w_i satisfies

$$w_i = a_{i,1}\, x_{j_1} \oplus a_{i,2}\, x_{j_2} \oplus \cdots \oplus a_{i,n}\, x_{j_n} \ ,$$

where $j_t = p_t \oplus k_{1,t}$ is a known index. If the corresponding s-box computation $y_i \leftarrow S(w_i \oplus k_{2,i})$ collides with some s-box computation $\mu \leftarrow S(\beta)$ from $\mathcal{B}$, then we get the following quadratic equation:

$$a_{i,1}\, x_{j_1} \oplus a_{i,2}\, x_{j_2} \oplus \cdots \oplus a_{i,n}\, x_{j_n} \oplus k_{2,i} = \beta \ .$$

Once several such equations have been collected, one can solve the system and recover all the unknowns (*i.e.* the x_i's, the $a_{i,j}$'s and the $k_{2,i}$'s).

Solving the system. In order to solve the quadratic system obtained from all the collected equations, one can use the linearization method. The monomial $a_{i,j}\, x_u$ is replaced by a new unknown y_t for every triplet $t \equiv (i, j, u)$ where $1 \leq i, j \leq n$ and $0 \leq u \leq 2^m - 1$. We get a linear system with $2^m n^2 + n$ unknowns (the y_t's and the $k_{2,i}$'s), which can be solved based on $2^m n^2 + n$ independent equations. Since every encryption provides n new equations, the required number of encryptions is $2^m n + 1$.

However, using linearization is not mandatory and we show hereafter that the system can be directly rewritten as a linear system. To do so, we consider the n equations obtained for the different s-box computations at the same time. Let $\beta_1, \beta_2, \ldots, \beta_n$ be the values such that $y_i \leftarrow S(w_i \oplus k_{2,i})$ collides with $\mu_i \leftarrow S(\beta_i)$. The obtained system for the n equations can be written in matrix form as

$$A \cdot \boldsymbol{x} \oplus \boldsymbol{k_2} = \boldsymbol{\beta} \ ,$$

where $A = (a_{i,j})_{i,j}$, $\boldsymbol{x} = (x_{j_1}, x_{j_2}, \ldots, x_{j_n})^T$, $\boldsymbol{k_2} = (k_{2,1}, k_{2,2}, \ldots, k_{2,n})^T$ and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)^T$. Since λ is invertible, we have

$$\boldsymbol{x} \oplus A^{-1} \cdot \boldsymbol{k_2} = A^{-1} \cdot \boldsymbol{\beta} \ .$$

Let $\boldsymbol{k'_2} = (k'_{2,1}, k'_{2,2}, \ldots, k'_{2,n})$ denote the vector resulting from the product $A^{-1} \cdot \boldsymbol{k_2}$ and let $a'_{i,j}$ denote the coefficients of A^{-1}. We obtained the n following equations:

$$\begin{aligned}
x_{j_1} \oplus k'_{2,1} &= a'_{1,1}\, \beta_1 \oplus a'_{1,2}\, \beta_2 \oplus \cdots \oplus a'_{1,n}\, \beta_n \ , \\
x_{j_2} \oplus k'_{2,2} &= a'_{2,1}\, \beta_1 \oplus a'_{2,2}\, \beta_2 \oplus \cdots \oplus a'_{2,n}\, \beta_n \ , \\
&\vdots \\
x_{j_n} \oplus k'_{2,n} &= a'_{n,1}\, \beta_1 \oplus a'_{n,2}\, \beta_2 \oplus \cdots \oplus a'_{n,n}\, \beta_n \ .
\end{aligned}$$

After collecting several such equations, we obtained a linear system with $n^2 + n + 2^m$ unknowns: the x_i's, the $a'_{i,j}$'s and the $k'_{2,i}$'s. This system can hence be solved based on $n^2 + n + 2^m$ independent equations. Since every encryption provides n new equations, the required number of encryptions is at least $n + 1 + 2^m/n$. Once all the $a'_{i,j}$'s and the $k'_{2,i}$'s have been recovered, we can inverse the matrix A^{-1} to get λ and then compute $\boldsymbol{k_2} = A \cdot \boldsymbol{k'_2}$.

Degrees of freedom. As explained in Section 3.2, we must fix the value of $a_{1,1}$ in order to fix a representation among the equivalence class of the cipher. For the above system, this amounts to fixing the value of $a'_{1,1}$. We hence add the equation $a'_{1,1} = 1$ to the system. Here again, $a'_{1,1}$ may equal 0 in which case the solving fails and the attacker must try again by fixing $a'_{1,2}$ and so on. Another degree of freedom exists that is not recovered by solving the above system: one can add a fixed offset δ to every s-box output and to every coordinate of $\boldsymbol{k'_2}$ (which amounts to add $A \cdot (\delta, \delta, \ldots, \delta)$ to $\boldsymbol{k_2}$). Clearly, such a modification would not change the collected equations. In order to set this degree of freedom, we can fix one of the s-box output, say x_0 to 0. To summarize, additionally to the collected n-equation groups from each encryption, we add the equations $a'_{1,1} = 1$ and $x_0 = 0$ in order to obtain a full rank system.

Note that fixing $x_0 = 0$ may induce a non-equivalent representation of the cipher. Indeed, the recovered cipher is equivalent to the real cipher but a fixed offset δ is xor-ed to each s-box outputs in the last round. As a consequence the resulting ciphertexts are xor-ed with the constant value $A \cdot (\delta, \delta, \ldots, \delta)$. Note that in the case where a key-addition is performed after the nonlinear layer in the last round (see Remark 1) then its recovery absorbs this offset as for the other rounds (and hence δ is just an additional degree of freedom in the equivalence class of the cipher). Otherwise, one must recover the offset δ in order to correct the ciphertext values and get an equivalent representation of the cipher. This can be easily done by comparing a real ciphertext with the one obtained from the recovered cipher.

Chosen plaintexts attack. To optimize the attack, one shall select the plaintexts in order to make every unknown of the system appear with the least possible number of requested encryptions. The $a'_{i,j}$'s and the $k'_{2,i}$'s all appear in each group of n equations resulting from a single encryption. On the other hand such a group of equations only involves n out of 2^m unknowns x_i's. The best approach is hence to make n different x_i's appear for each encryption request. To do so, one can simply ask for the encryption of the plaintext

$$(i \cdot n + 0,\ i \cdot n + 1,\ \ldots,\ i \cdot n + n - 1) \oplus (k_{1,1},\ k_{1,2},\ \ldots,\ k_{1,n})\ ,$$

for $i = 0, 1, \ldots, \lceil 2^m/n \rceil - 1$. The s-box inputs in the first round of the corresponding encryptions then equal $(0, 1, 2, \ldots, n-1)$, $(n, n+1, \ldots, 2n-1)$, and so on. Every possible s-box value thus appears in the system after $\lceil 2^m/n \rceil$ encryptions. Afterwards, one just needs the encryption of $n + 1$ additional plaintexts to get a full rank linear system in the $n^2 + n + 2^m$ unknowns.

3.5 Stage 3: Recovering k_3, k_4, ..., k_r

Once the two first stages have been completed, it only remains to recover the last round keys k_3, k_4, ..., k_r. This is simply done by detecting a collision between two s-box computations $y_i \leftarrow S(p_{j,i} \oplus k_{j,i})$ and $\mu_{j,i} \leftarrow S(\beta_{j,i})$, giving $k_{j,i} = p_{j,i} \oplus \beta_{j,i}$, for every s-box index $i \in \{1, 2, \ldots, n\}$ and every round index $j \in \{1, 2, \ldots, r\}$.

4 Extensions to More General SPN Structures

In this section, we extend the previous attack to take into account the possible generalizations described in Section 2.1. We shall only detail the generalization of the two first stages since the third one keeps pretty similar.

4.1 Multiple S-Boxes Setting

In the multiple s-boxes setting, the nonlinear layer γ is defined as:

$$\gamma(p) = (S_1(p_1), S_2(p_2), \ldots, S_n(p_n)) \ ,$$

where $S_1, S_2, \cdots, S_n$ are n different s-boxes.

In this setting, our basic assumption is that the attacker is able to detect if two s-box computations $y_1 \leftarrow S_i(x_1)$ and $y_2 \leftarrow S_i(x_2)$ collide, for a given s-box S_i. Since two s-boxes S_i and S_j are different for different indices $i \neq j$, we consider that one cannot detect if two s-box computations $y_1 \leftarrow S_i(x_1)$ and $y_2 \leftarrow S_j(x_2)$ are such that $x_1 = x_2$ or not. Indeed the underlying side-channel leakages are likely to be different even if we have $x_1 = x_2$.

Equivalent representations. As for the single s-box setting, the cipher has several equivalent representations. In particular, for every $(\delta_1, \delta_2, \ldots, \delta_n) \in (\mathbb{F}_{2^m})^n$, replacing γ by the layer γ' defined as

$$\gamma'(p) = (S_1(p_1 \oplus \delta_1), S_2(p_2 \oplus \delta_2), \ldots, S_n(p_n \oplus \delta_n)) \ , \tag{3}$$

and replacing every round key $k_i = (k_{i,1}, k_{i,2}, \ldots, k_{i,n})$ by

$$k'_i = (k_{i,1} \oplus \delta_1, k_{i,2} \oplus \delta_2, \ldots, k_{i,n} \oplus \delta_n) \tag{4}$$

does not change the cipher in a functional sense. And here again, the ability of detecting collisions in s-box computations does not make it possible to distinguish between two different equivalent representations.

The other way to obtain equivalent representations is by changing the nonlinear layer γ for the nonlinear layer γ' defined as

$$\gamma'(p) = (\alpha_1 \cdot S_1(p_1), \alpha_2 \cdot S_2(p_2), \ldots, \alpha_n \cdot S_n(p_n))$$

for some $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in (\mathbb{F}_{2^m}^*)^n$, and by replacing the linear layer λ defined in (1) by the linear layer λ' obtained from the matrix $(a'_{i,j})_{i,j}$ whose coefficients satisfy

$$a'_{i,j} = \frac{a_{i,j}}{\alpha_j}$$

for every i and j.

Generalized attack. To deal with equivalent representations in the multiple s-box setting, one can set the first round key k_1 to $(0, 0, \ldots, 0)$, and hence skip the first stage of the attack. For the second stage (recovery of γ, λ and k_2), the procedure is quite similar to that in the single s-box setting. However, each s-box S_i leads to its own set of 2^m unknowns $x_{i,0} = S_i(0)$, $x_{i,1} = S_i(1)$, $\ldots$, $x_{i,2^m-1} = S_i(2^m - 1)$. We must hence solve a system with $n \times 2^m + n^2 + n$ unknowns. To construct this system, the attacker build a leakage basis $\mathcal{B}_i$ for each s-box S_i. Namely, he collects the leakage $\ell_{i,\beta}$ from the s-box computation $\mu \leftarrow S_i(\beta)$ for every $i \in \{1, 2, \ldots, n\}$ and for every $\beta \in \mathbb{F}_{2^m}$ to get the n leakage basis $\mathcal{B}_i = \{\ell_{i,\beta} \mid \beta \in \mathbb{F}_{2^m}\}$.

When the attacker detects a collision between the ith s-box computation of the second round and the element $\ell_{i,\beta}$ from $\mathcal{B}_i$ for every i, he obtains the following quadratic equation:

$$a_{i,1}\, x_{1,p_1} \oplus a_{i,2}\, x_{2,p_2} \oplus \cdots \oplus a_{i,n}\, x_{n,p_n} \oplus k_{2,i} = \beta \ ,$$

where $(p_1, p_2, \ldots, p_n)$ is the input plaintext.

Here again, the obtained quadratic system can be linearized. Doing so, the number of unknowns increases up to $2^m n^2 + n$. The linearized system can hence be solved based on $2^m n^2 + n$ independent equations. Since every encryption provides n new equations, the expected number of required encryptions is $2^m n + 1$. Note that there is no increase of the number of unknowns compared to the linearized system in the single s-box setting (see Section 3.4) since the outputs of the s-box S_j are multiplied by the coefficients of a unique column of the matrix, namely $(a_{1,j}, a_{2,j}, \ldots, a_{n,j})^T$.

Here again, the system can be directly rewritten as a linear system in order to avoid the increase of unknowns. If the ith s-box computation in the second round collides with $\mu_i \leftarrow S_i(\beta_i)$ from the leakage basis $\mathcal{B}_i$, we get the n following equations:

$$\begin{aligned}
x_{1,p_1} \oplus k'_{2,1} &= a'_{1,1}\, \beta_1 \oplus a'_{1,2}\, \beta_2 \oplus \cdots \oplus a'_{1,n}\, \beta_n \ , \\
x_{2,p_2} \oplus k'_{2,2} &= a'_{2,1}\, \beta_1 \oplus a'_{2,2}\, \beta_2 \oplus \cdots \oplus a'_{2,n}\, \beta_n \ , \\
&\vdots \\
x_{n,p_n} \oplus k'_{2,n} &= a'_{n,1}\, \beta_1 \oplus a'_{n,2}\, \beta_2 \oplus \cdots \oplus a'_{n,n}\, \beta_n \ ,
\end{aligned}$$

where the $a'_{i,j}$'s are the coefficients of the inverse matrix A^{-1} and $(k'_{2,1}, k'_{2,2}, \cdots, k'_{2,n}) = A^{-1} \cdot (k_{2,1}, k_{2,2}, \cdots, k_{2,n})^T$.

From these equations, we obtain n independent linear systems: one per row of A^{-1} or equivalently one per s-box S_i. Each system has $2^m + n + 1$ unknowns: the $x_{i,j}$'s for $0 \leq j \leq 2^m - 1$, the $a'_{i,j}$'s for $1 \leq j \leq n$ and $k'_{2,i}$ (for a given row index i).

Degrees of freedom. As for the basic attack, each subsystem has additional degrees of freedom. The first one corresponds to the second equivalence relation described above. Namely one can multiply S_i by a non-zero value α_i and the ith column of A by α_i^{-1} (which amounts multiplying the ith row of A^{-1} by α_i). This degree of freedom is fixed by setting $a'_{1,i}$ to 1. Once again, if $a'_{1,i}$ equals 0, we get an inconsistent system, then we try again by fixing $a'_{1,i} = 1$ and so on. Then, each of the n s-boxes has its own additional degree of freedom. One can indeed xor an offset δ_i to each output of S_i and to $k'_{2,i}$ for every i. These degrees

of freedom are fixed by setting $k'_{2,i} = 0$ for every i. Such statement results in a xor-ing of the constant $A \cdot (\delta_1, \delta_2, \ldots, \delta_n)$ to the ciphertexts, which is easily corrected once the overall cipher has been recovered.

Chosen-plaintext attack. For a chosen plaintext attack, we shall request the encryption of the 2^m plaintexts $(j, j, \ldots, j)$ for $0 \leq j \leq 2^m - 1$ in order to make appear all the unknowns $(x_{i,j})_j$ in every subsystem. Then the equations arising from the encryption of $n+1$ additional (random) plaintexts yield a full rank system for every i.

4.2 Binary Linear Layer Setting

In the binary linear layer setting, the linear layer λ is defined over $\mathbb{F}_2$, namely the matrix A is a full rank $\ell \times \ell$ binary matrix.

Equivalent representations. Let $A_{i,j}$ be the $m \times m$ binary sub-matrices of A such that

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \cdots & A_{n,n} \end{pmatrix} . \tag{5}$$

An equivalent representation of the cipher can be obtained by replacing S with $\psi \circ S$ and every sub-matrix $A_{i,j}$ with $A_{i,j} \cdot M_\psi^{-1}$ where ψ is a bijective linear transformation over $\mathbb{F}_2^m$ and M_ψ is the corresponding $m \times m$ full-rank binary matrix.

Generalized attack. The first step of the attack keeps unchanged and the second stage also starts by constructing a leakage basis $\{\ell_\beta\}$ for every possible s-box computation $\mu \leftarrow S(\beta)$ with $\beta \in \mathbb{F}_{2^m}$.

Each unknown x_i is then split in m binary unknowns $x_{i,1}$, $x_{i,2}$, …, $x_{i,m}$ corresponding to the m unknown bits of $S(i)$. A collision detection between the ith s-box computation in the second round and the element ℓ_β from the leakage basis $\mathcal{B}$, leads to the following equation:

$$A_{i,1} \cdot x_{j_1} \oplus A_{i,2} \cdot x_{j_2} \oplus \cdots \oplus A_{i,n} \cdot x_{j_n} \oplus k_{2,i} = \beta \ ,$$

where $j_t = p_t \oplus k_{1,t}$ is a known index. Each equation as above yields m binary linear equations in the $x_{i,j}$'s, the binary coefficients of the $A_{i,j}$'s and the bits of k_2. We hence get n groups of m binary equations per encryption, that is a total of ℓ binary linear equations per encryption.

Once several such equations have been collected, one can solve the system and recover all the $m2^m + \ell^2 + \ell$ binary unknowns (*i.e.* the $x_{i,j}$'s, the binary coefficients of the $A_{i,j}$'s and the bits of k_2). In the exact same way as for the basic attack, this equation system may be solved by linearization of the quadratic monomials or by rewriting the equation system as a linear one and recovering the inverse of A. For the first solution, the number of binary unknowns is of $2^m\ell^2 + \ell$. The equation system can hence be solved based on $2^m\ell^2 + \ell$ independent equations.

Since every encryption provides ℓ new equations, the required number of encryptions is $2^m\ell+1$. In the second solution, the system can be solved with only $m2^m+\ell^2+\ell$ independent equations, which requires $2^m/n+\ell+1$ encryptions.

Degrees of freedom. Here again, additional degrees of freedom must be taken into account. On the one hand, due to the equivalent representation presented above, composing the s-box with a linear transformation ψ over $\mathbb{F}_2^m$ is canceled out by multiplying M_ψ^{-1} to each sub-matrix $A_{i,j}$. To remove this degree of freedom, one just need to fix the value of say $A'_{1,1}$ (the top left $m \times m$ sub-matrix of $A' = A^{-1}$). We arbitrary choose to set $A'_{1,1}$ to the identity matrix. If the matrix $A'_{1,1}$ was actually singular, this would lead to an inconsistent system. One would then try fixing $A'_{1,2}$ to the identity matrix, and so on until the system can be solved. As in the basic attack, one more degree of freedom must be considered. Indeed, one can xor an offset δ to the output of the s-box and to $k_{2,0}$ without affecting the system of collected equations. This degree of freedom can be removed by adding the equation $k_{2,0} = 0$ to the system. This shall lead to the recovery of the unknown cipher up to the xor of a constant to the ciphertexts (specifically $A \cdot (\delta, \delta, \ldots, \delta)$), which can be easily retrieved at the end of the attack.

4.3 Multiple S-Boxes and Binary Linear Layer Setting

We now address the combination of the two previous generalized settings *i.e.* when the cipher has both multiple s-boxes and a binary linear layer. The following generalized attack is a natural combination of the two previous generalized attacks.

Equivalent representations. As for the multiple s-box setting, we can obtain an equivalent representation of the cipher by replacing γ with γ' as defined in (3) and replacing every round key k_i with k'_i as defined in (4), for some $(\delta_1, \delta_2, \ldots, \delta_n) \in (\mathbb{F}_{2^m})^n$. On the other hand, and as in the binary linear layer setting, we can also obtain an equivalent representation of the cipher by replacing each s-box S_i with $\psi_i \circ S_i$ and every sub-matrix $A_{i,j}$ as defined in (5) by $A_{i,j} \cdot M_{\psi_i}^{-1}$, for some bijective linear transformations $\psi_i : \mathbb{F}_2^m \to \mathbb{F}_2^m$ and corresponding $m \times m$ full-rank binary matrices M_{ψ_i}.

Generalized attack. In the combined generalized setting, each s-box S_i leads to its own set of 2^m unknowns $x_{i,0} = S_i(0)$, $x_{i,1} = S_i(1)$, ..., $x_{i,2^m1} = S_i(2^m1)$, each of them split into m binary unknowns $x_{i,j,1}$, $x_{i,j,2}$, ..., $x_{i,j,m}$ representing the m bits of $x_{i,j}$.

As in the multiple s-box setting, the attack starts by fixing k_1 to $(0, 0, \ldots, 0)$ in order to fix the first degree of freedom in the class of equivalence of the cipher, and by constructing a basis $\mathcal{B}_i = \{\ell_{i,\beta} \mid \beta \in \mathbb{F}_{2^m}\}$ for every s-box S_i. When the attacker detects a collision between the ith s-box computation of the second round and the element $\ell_{i,\beta}$ from $\mathcal{B}_i$ for every i, he obtains the following quadratic equation:

$$A_{i,1} \cdot x_{1,p_1} \oplus A_{i,2} \cdot x_{2,p_2} \oplus \cdots \oplus A_{i,n} \cdot x_{n,p_n} \oplus k_{2,i} = \beta \ ,$$

where $(p_1, p_2, \ldots, p_n)$ is the input plaintext. Each equation as above yields m binary linear equations in the $x_{i,j,t}$'s, the binary coefficients of the $A_{i,j}$'s and the bits of k_2. We hence get n groups of m binary equations per encryption, that is a total of ℓ binary linear equations per encryption.

Once several such equations have been collected, one can solve the system and recover all the $\ell 2^m + \ell^2 + \ell$ binary unknowns (*i.e.* the $x_{i,j,t}$'s, the binary coefficients of the $A_{i,j}$'s and the bits of k_2). As previously, this can be done either by linearization of the system of by rewriting the system as a linear one (and recovering the inverse of A.). In the latter case, one get n independent linear systems (one per s-box S_i) of $m2^m + m\ell + m$ binary unknowns.

Degrees of freedom. To fix a representation in the equivalence class of the cipher, one sub-matrix $A'_{i,j}$ per subsystem, say $A'_{i,1}$, must be set to the identity matrix. As previously, if $A'_{i,1}$ is actually singular, one gets an inconsistent subsystem, and one can then try again with $A'_{i,2}$, and so on. Then we have n additional degrees of freedom (one per s-box) as in the multiple s-box setting, which can be fixed by setting k_2 to $(0, 0, \ldots, 0)$. As previously detailed, fixing these degrees enable the recovery of the cipher up to some constant xor-ed to the ciphertext and which can be easily retrieved at the end of the attack.

5 Extension to Feistel Schemes

In this section, we extend our SCARE attack to the generic structure of Feistel schemes described in Section 2.2. Due to the potentially non-invertible Feistel function, the attack is slightly more tricky but it follows the same steps as for the classical SPN structure.

Here again the attacker is assumed to know the block size $\ell = 2nm$ and, by identifying the s-box computations from leakage trace, he can also retrieve the number s of s-box computations per round. But due to the unknown extension/compression linear layers, the attacker is not able to directly recover the dimension m of the s-box. This is not a big issue in practice as one could just guess the value of m and mount the attack until it succeeds. Note that the possible values are limited to divisors of ℓ in a reasonable range (most often between 4 and 8).

5.1 Equivalent Representations

As for the classical SPN structure, several equivalent representations are possible for a Feistel scheme. For instance one can replace the s-box S for the s-box S' defined as

$$S'(x) = S(\alpha_1 \cdot (x \oplus \delta))$$

for some $\delta \in \mathbb{F}_{2^m}$ and $\alpha_1 \in \mathbb{F}^*_{2^m}$, replace the linear layer λ_1 defined in (2) by the linear layer λ'_1 obtained from the matrix $(a'_{i,j})_{i,j}$ whose coefficients satisfy

$$a'_{i,j} = \frac{a_{i,j}}{\alpha_1}$$

for every i and j, and replace every round key $k_i = (k_{i,1}, k_{i,2}, \ldots, k_{i,n})$ by

$$k'_i = \Big(\frac{k_{i,1} \oplus \delta}{\alpha_1}, \frac{k_{i,2} \oplus \delta}{\alpha_1}, \ldots, \frac{k_{i,n} \oplus \delta}{\alpha_1}\Big) \ .$$

The two representations are clearly equivalent in a functional sense. Moreover, the ability to detect collisions in s-box computations does not make it possible to distinguish between these two different equivalent representations.

Another way to obtain equivalent representations is by changing the s-box S for the s-box S' defined as

$$S'(x) = \alpha_2 \cdot S(x)$$

for some $\alpha_2 \in \mathbb{F}_{2^m}^*$, and by replacing the linear layer λ_2 defined in (2) by the linear layer λ'_2 obtained from the matrix $(b'_{i,j})_{i,j}$ whose coefficients satisfy

$$b'_{i,j} = \frac{b_{i,j}}{\alpha_2}$$

for every i and j.

As for the attack against classical SPN structure, we fix the first round key coordinate $k_{1,1}$ to 0 and we fix the coefficients $a_{1,1}$ and $b_{1,1}$ to 1, which is equivalent to fixing the variables δ, α_1 and α_2. Note that $a_{1,1}$ (resp. $b_{1,1}$) may equal 0 (which is revealed by the attack), in which case we try fixing $a_{1,2}$ (resp. $b_{1,2}$), then $a_{1,3}$ (resp. $b_{1,3}$), and so on. We describe hereafter the successive steps of the attack.

5.2 Stage 1: Recovering λ_1 and k_1

Let $\lambda_1(q) = (w_1, w_2, \cdots, w_s)$ denote the output of the linear extension layer in the processing of the first round function f_{k_1}. For every $i \leq s$, w_i satisfies

$$w_i = a_{i,1}q_1 \oplus a_{i,2}q_2 \oplus \cdots \oplus a_{i,n}q_n \ ,$$

where $q = (q_1, \cdots, q_n)$ denotes the right part of the plaintext. Then for any collision between two s-box computations $y_i \leftarrow S(w_i \oplus k_{1,i})$ and $y'_j \leftarrow S(w'_j \oplus k_{1,j})$, where w_i and w'_j may come from different executions, we have $w_i \oplus k_{1,i} = w'_j \oplus k_{1,j}$, that is

$$a_{i,1}\, q_1 \oplus a_{i,2}\, q_2 \oplus \cdots \oplus a_{i,n}\, q_n \oplus k_{1,i} = a_{j,1}\, q'_1 \oplus a_{j,2}\, q'_2 \oplus \cdots \oplus a_{j,n}\, q'_n \oplus k_{1,j} \ .$$

Once several such equations have been collected, we get a linear system with $ns + s - 2$ unknowns (*i.e.* the $a_{i,j}$'s and the $k_{1,i}$'s, except $a_{1,1}$ and $k_{1,1}$). This system can hence be solved based on $ns + s - 2$ independent equations. Since every encryption is expected to provide s new equations, the expected required number of encryptions is $n + 1$.

5.3 Stage 2: Recovering λ_2, S and k_2

Once λ_1 and k_1 have been recovered, one knows the inputs of the s-box in the first round. Recovering the s-box means recovering the 2^m unknowns x_0, x_1, ..., x_{2^m-1}, where $x_i = S(i)$ for every i. The attack consists in constructing a set of equations in the x_i's, the $b_{i,j}$'s and the $k_{2,i}$'s. Solving the obtained system hence amounts to recover λ_2, S and k_2.

As for the classical SPN structure, the first step of this stage consists in collecting the leakage ℓ_β from the s-box computation $\mu \leftarrow S(\beta)$ for every $\beta \in \mathbb{F}_{2^m}$. We shall denote by $\mathcal{B}$ the obtained leakage basis $\{\ell_\beta \mid \beta \in \mathbb{F}_{2^m}\}$. The basis is then used to detect collisions between s-box computations in the second round and s-box computations $\mu \leftarrow S(\beta)$. Let v_j be the jth m-bit coordinate in output of the first Feistel function in the encryption of some plaintext (p, q). Then v_j satisfies

$$v_j = b_{j,1}\, x_{t_1} \oplus b_{j,2}\, x_{t_2} \oplus \cdots \oplus b_{j,s}\, x_{t_s} \ ,$$

where $t_i = a_{i,1}q_1 \oplus a_{i,2}q_2 \oplus \cdots \oplus a_{i,n}q_n \oplus k_{1,i}$ is a known index. Then let w_i be the ith output of the linear extension layer in the processing of the second round function f_{k_2} in the encryption of (p, q). We have

$$w_i = a_{i,1}(v_1 \oplus p_1) \oplus a_{i,2}(v_2 \oplus p_2) \oplus \cdots \oplus a_{i,n}(v_n \oplus p_n) \ .$$

If the corresponding s-box computation $y_i \leftarrow S(w_i \oplus k_{2,i})$ collides with some s-box computation $\mu \leftarrow S(\beta)$ from $\mathcal{B}$, then we get the following equation

$$a_{i,1}(v_1 \oplus p_1) \oplus a_{i,2}(v_2 \oplus p_2) \oplus \cdots \oplus a_{i,n}(v_n \oplus p_n) = \beta \ ,$$

that is

$$\bigoplus_{j=1}^{n} a_{i,j} \Big(\bigoplus_{u=1}^{s} b_{j,u}\, x_{t_u} \Big) = \omega_i \oplus k_{2,i} \ , \tag{6}$$

where $\omega_i = \beta \oplus \bigoplus_{j=1}^{n} a_{i,j}\, p_j$. Note that the $a_{i,j}$'s are also known, hence the above equation is quadratic. Once several such equations have been collected, one can solve the system and recover all the unknowns (*i.e.* the x_i's, the $b_{i,j}$'s and the $k_{2,i}$'s).

Solving the system. A straightforward approach to solve this quadratic system is to use linearization. The monomial $b_{i,j}\, x_u$ is replaced by a new unknown y_t for every triplet $t \equiv (i, j, u) \in [1; n] \times [1; s] \times [0; 2^m - 1]$. We get a linear system with $2^m ns + s$ unknowns (the y_t and the $k_{2,i}$), which can be solved based on $2^m ns + s$ independent equations. Since every encryption provides s new equations, the required number of encryptions is $2^m n + 1$.

Like in the classical SPN case, we can get a simpler system by inverting the linear part. However, the linear part of the Feistel scheme may not be bijective in which case we can only partly invert it. We describe hereafter a chosen-plaintext approach to do so. Our attack further assumes that the extended dimension s is less that twice the original dimension n (*i.e.* $s < 2n$).

Chosen-plaintext attack. Let $\beta_1, \beta_2, \ldots, \beta_s$ be the s values such that $y_i \leftarrow S(w_i \oplus k_{2,i})$ collides with $\mu_i \leftarrow S(\beta_i)$, and let $\omega_1, \omega_2, \ldots, \omega_s$ be the s values such that $\omega_i = \beta_i \oplus \bigoplus_{j=1}^{n} a_{i,j} p_j$. The obtained system for the s equations can be written in matrix form as

$$A \cdot B \cdot \boldsymbol{x} \oplus \boldsymbol{k_2} = \boldsymbol{\omega} \ , \tag{7}$$

where $A = (a_{i,j})_{i,j}$, $B = (b_{i,j})_{i,j}$, $\boldsymbol{x} = (x_{t_1}, x_{t_2}, \ldots, x_{t_s})^T$, $\boldsymbol{k_2} = (k_{2,1}, k_{2,2}, \ldots, k_{2,s})^T$ and $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_s)^T$.

Since A is a full rank $s \times n$ matrix with $s \geq n$, it admits a $n \times s$ left inverse A^{-1} (which is known at this stage of the attack), we thus have

$$B \cdot \boldsymbol{x} \oplus \tilde{\boldsymbol{k_2}} = \tilde{\boldsymbol{\omega}} \ ,$$

where $\tilde{\boldsymbol{k_2}} = A^{-1} \cdot \boldsymbol{k_2}$ and $\tilde{\boldsymbol{\omega}} = A^{-1} \cdot \boldsymbol{\omega}$ are vectors of size n.

In order to get rid of B in the left part of the formula, the attacker chooses plaintexts so that $s-n$ s-box outputs, say $x_{t_{n+1}}, x_{t_{n+2}}, \cdots, x_{t_s}$, take constant (unknown) values. To do so, the attacker fixes the corresponding s-box inputs to arbitrary constant values by selecting plaintexts with right part q lying in a $(2n-s)$-dimensional vectorial subspace $(\mathbb{F}_{2^m})^n$. Let $\tilde{B}$ denote the truncated matrix composed of the n first columns of B. Likewise, let $\tilde{\boldsymbol{x}}$ be the truncated vector composed of the n first coordinates of $\boldsymbol{x}$. Eventually, let $\boldsymbol{z}$ the fixed but unknown vector of size n defined such that $z_i = b_{i,n+1}\, x_{t_{n+1}} \oplus \cdots \oplus b_{i,s} x_{t_s}$. With these notations we now have

$$\tilde{B} \cdot \tilde{\boldsymbol{x}} \oplus \boldsymbol{z} \oplus \tilde{\boldsymbol{k_2}} = \tilde{\boldsymbol{\omega}} \ .$$

Since B is a full rank matrix, the truncated square matrix $\tilde{B}$ is also of full rank and hence invertible. The above equation can then be rewritten as

$$\tilde{\boldsymbol{x}} \oplus \tilde{\boldsymbol{k_2}}' = \tilde{B}^{-1} \cdot \tilde{\boldsymbol{\omega}} \ ,$$

where $\tilde{\boldsymbol{k_2}}' = \tilde{B}^{-1} \cdot (\boldsymbol{m} \oplus \tilde{\boldsymbol{k_2}})$ is a constant unknown vector.

We hence obtain a linear system with $2^m + n + n^2$ unknowns. Since each encryption provides n new equations, the expected number of encryption to get a full-rank system is $2^m/n + n + 1$. In order to fix a representation among the equivalence class of the cipher (see Section 5.1) and remove the corresponding degree of freedom, one can set one coefficient of $\tilde{B}^{-1}$ to 1. Once again, if this coefficient is actually 0, one get an unsolvable system, and one must try again with another coefficient.

Once the above system has been solved and all the x_i's have been recovered, the original quadratic system (see (7)) becomes a linear system with $n\,(s-n) + n$ unknowns (namely the $b_{i,j}$'s with $j > n$ and the $k_{2,i}$'s). The expected number of encryptions to get a solvable system is then of $s-n+1$. Eventually, the recovery of the next round keys follows the exact same approach as in the classical SPN case.

5.4 Generalized Feistel Structures

Note that the Feistel structure described in Section 2.2 can be generalized as the classical SPN structure by considering either multiple s-boxes, or binary linear layers (or both). The extensions of our attack described in Section 4 to deal with these settings can be simply transposed to the attack against the Feistel structure, so we do not give further details here.

6 SCARE in the Presence of Noisy Leakage

So far, we have considered an idealized model in which the attacker is able to detect a collision between two s-box computations from their respective leakages with a 100% confidence. As a matter of facts, the proposed SCARE attack does not tolerate any false-positive error in the collision detections. In this section, we relax this assumption and describe a practical SCARE attack in the presence of noise in the side-channel leakage. As for the basic attack, the principle is to exploit equations arising from collisions in s-box computations. We explain hereafter how to collect sound equations with high confidence in the presence of noisy leakage.

We only focus on secret ciphers with classical SPN structures (as described in Section 2.1), but the proposed techniques naturally extend to the attack generalizations proposed in the previous sections.

6.1 Stage 1: Recovering k_1

In our SCARE attack, the first stage exactly corresponds to the usual scenario of *linear collision attacks* that aim at recovering key bytes differences $k_{1,i} \oplus k_{1,j}$ by detecting collisions between s-box computations in the first round from the side-channel leakage [4, 5, 18, 25].

In a linear collision attack, the attacker is assumed to possess the leakage traces corresponding to the encryption of N random plaintexts $((p_t)_{t \leq N})$. Let $\ell_{t,i}$ denote the leakage associated to ith s-box computation in the encryption of p_t. The principle is to compute the mean leakage $\bar{\ell}_{i,x}$ of the set $\{\ell_{t,i} \ ; \ p_{t,i} = x\}$ for every i and x, in order to average the leakage noise and detect collisions more easily. As explained in Section 3.3, detecting a collision between $\bar{\ell}_{i,x}$ and $\bar{\ell}_{j,y}$ implies the equality of the two s-box inputs $x \oplus k_{1,i}$ and $y \oplus k_{1,j}$ and provides the linear equation $k_{1,i} \oplus k_{1,j} = x \oplus y$. In [4], Bogdanov points out that the equation system arising from the key byte differences is overdetermined and that the redundant information could be used to tolerate some erroneous equations. In [18], Gérard and Standaert further show that solving such an equation system can be written as a LDPC[3] code decoding problem for which an efficient algorithm is known. We suggest to use their method for the first stage of our practical SCARE attack, whose principle is recalled in Appendix A.

6.2 Stage 2: Recovering λ, S and k_2

As for the attack without collision errors, the second stage is the main task. To deal with the leakage noise, we make the well admitted *Gaussian noise assumption*. Namely, we assume that the leakage corresponding to an s-box computation $\mu \leftarrow S(\beta)$ follows a multivariate Gaussian distribution with mean m_β and covariance matrix Σ_β, denoted $\mathcal{N}(m_\beta, \Sigma_\beta)$.

Building leakage templates. The first step of the second stage consists in estimating the leakage parameters. Namely, for each $\beta \in \mathbb{F}_{2^m}$ we estimate the mean m_β and the covariance matrix Σ_β of the leakage from the s-box computation $\mu \leftarrow S(\beta)$. The leakage basis of the noise-free attack is then replaced by a leakage template basis $\mathcal{B} = \{(\widehat{m}_\beta, \widehat{\Sigma}_\beta)_\beta \mid \beta \in \mathbb{F}_{2^m}\}$

[3] Low Density Parity Check

where $\widehat{m}_\beta$ and $\widehat{\Sigma}_\beta$ denote the estimated values for the leakage parameters. The estimation is obtained from the leakages used in the first stage, and possibly more, until the estimated means converge.

Our convergence criterion is based on the Hotelling T^2-test which is the natural extension of the Student T-test for multinormal distributions (see for instance [22]). Let d denote the dimension of the distribution $\mathcal{N}(m_\beta, \Sigma_\beta)$ *i.e.* the number of points in an s-box leakage trace, and let $F^{-1}_{(d_1,d_2)}$ denote the quantile function of the Fisher's F-distribution with parameters (d_1, d_2) (*i.e.* $F_{(d_1,d_2)}$ is the distribution CDF). For some confidence parameter $\alpha \in [0; 1]$ and some estimation quality parameter $q \in [0; 1]$, our convergence criterion is satisfied when we have:

$$R_\alpha\Big(\frac{\widehat{\sigma}^2_\beta}{\det(\widehat{\mathbf{S}})}\Big)^{1/d} \leq q \qquad \text{where} \quad R_\alpha := \frac{d}{N-d}\, F^{-1}_{(d,N-d)}(\alpha) \quad \text{and} \quad \widehat{\sigma}^2_\beta := \det(\widehat{\Sigma}_\beta)\ .$$

The rationale of this definition is detailed in Appendix B.

Based on this criterion, the template basis is built iteratively: we first collect N leakage samples for every s-box input value β. Based on these samples, we estimate the distribution parameters $(\widehat{m}_\beta, \widehat{\Sigma}_\beta)$ for every β, as well as the interclass covariance matrix $\widehat{\mathbf{S}}$. Then if we have $\max_\beta R_\alpha(\widehat{\sigma}^2_\beta/\det(\widehat{\mathbf{S}}))^{1/d} \leq q$ for some chosen confidence α and estimation quality parameter q we stop. Otherwise we continue with twice more samples (namely we collect N more leakage samples and set N to $2N$), and so on until we get a satisfying estimation quality. In practice, we shall use $\alpha = 99\%$ and $q = 0.5$.

Remark 3. A possible variant for building the template basis is to make the identical noise assumption which considers that Σ_β is equal to some constant matrix Σ for every β. This enables a better estimation $\widehat{\Sigma}$ based on all leakage samples.

Collecting equations. Once the template basis has been built, we collect several groups of n equations of the form $\boldsymbol{x} \oplus \boldsymbol{k'_2} = A^{-1} \cdot \boldsymbol{\beta}$, as in the basic attack (see Section 3.4). Due to the noise, we cannot determine the value of $\boldsymbol{\beta}$ with a 100% confidence. To deal with this issue we use averaging. Namely, the encryption of the same plaintext p is requested several – say N – times and we compute the average leakage for each s-box computation in the second round. Let ℓ_i denote the average leakage for the ith s-box, and let β^*_i denote the corresponding (unknown) s-box input. The average leakage ℓ_i follows a distribution $\mathcal{N}(m_{\beta^*_i}, \frac{1}{N}\Sigma_{\beta^*_i})$. Then we must recover the n corresponding values β^*_1, β^*_2, ..., β^*_n in order to get a group of equations. The problem is hence to determine to which distribution $\mathcal{N}(m_\beta, \frac{1}{N}\Sigma_\beta)$ belongs each leakage ℓ_i based on the template basis. For such a purpose, we use a maximum likelihood approach, namely we follow the classical approach of template attacks [10]. Given the leakage observation ℓ_i, the probability that the ith s-box input value β^*_i equals some value β satisfies

$$\Pr[\beta^*_i = \beta \mid \ell_i] = \frac{\phi_\beta(\ell_i)}{\sum_{\beta' \in \mathbb{F}_{2^m}} \phi_{\beta'}(\ell_i)}\ ,$$

where ϕ_β denotes the pdf of $\mathcal{N}(m_\beta, \frac{1}{N}\Sigma_\beta)$ satisfying

$$\phi_\beta(\ell) \propto \exp\big(-\frac{N}{2}(\ell - m_\beta)^T \cdot \Sigma_\beta^{-1} \cdot (\ell - m_\beta)\big) \ .$$

The likelihood of the candidate β for β_i^* based on the estimations $(\widehat{m}_\beta)_\beta$ and $(\widehat{\Sigma}_\beta)_\beta$ is hence defined as

$$\mathsf{L}(\beta \mid \ell_i) := \frac{\exp\big(-\frac{N}{2}(\ell_i - \widehat{m}_\beta)^T \cdot \widehat{\Sigma}_\beta^{-1} \cdot (\ell_i - \widehat{m}_\beta)\big)}{\sum_{\beta' \in \mathbb{F}_{2^m}} \exp\big(-\frac{N}{2}(\ell_i - \widehat{m}_{\beta'})^T \cdot \widehat{\Sigma}_{\beta'}^{-1} \cdot (\ell_i - \widehat{m}_{\beta'})\big)} \ . \tag{8}$$

The corresponding likelihood for a vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$ given the average leakage vector $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_n)$ can then be defined as $\mathsf{L}(\boldsymbol{\beta} \mid \boldsymbol{\ell}) := \prod_i \mathsf{L}(\beta_i \mid \ell_i)$. Note that the most likely candidate $\mathrm{argmax}_{\boldsymbol{\beta}}\ \mathsf{L}(\boldsymbol{\beta} \mid \boldsymbol{\ell})$ is also the one whose coordinates are the most likely *i.e.* equal to $\mathrm{argmax}_{\beta_i}\ \mathsf{L}(\beta_i \mid \ell_i)$ for every i.

In practice, we shall select the most likely value of $\boldsymbol{\beta}$ as the good one with a confidence $\mathsf{L}_{\boldsymbol{\beta}}$. However we not only want to select the best candidate, we further want its likelihood to be high (*i.e.* close to 1) in order to have a high confidence in the selected candidate. Getting a vector $\boldsymbol{\beta}$ with high likelihood may however be far more difficult than getting a single coordinate β_i with high likelihood since for the vector one needs all coordinates to have high likelihood. Indeed, the probability of having a high likelihood for the vector $\boldsymbol{\beta}$ is the probability of having a high likelihood for all coordinates β_i which is exponentially smaller in n.

To deal with this issue, our approach is to restrict the number of equations of the form $\boldsymbol{x} \oplus \boldsymbol{k'_2} = A^{-1} \cdot \boldsymbol{\beta}$ that are needed to succeed the attack. For such a purpose, we first solve a subsytem (*i.e.* with less unknowns) for which we require less equations than in the original attack, and then we recover the remaining unknowns based on simpler forms of equations.

Solving a subsystem. We first solve a subsystem involving the $a'_{i,j}$'s, the $k_{2,i}$'s and a restricted number of x_i's. To do so we select a set of s values β, say $\mathcal{S} = \{0, 1, \ldots, s-1\}$, and we only request the device for the encryption of plaintexts from the set

$$\mathcal{P}_s := \{(p_1, p_2, \ldots, p_n) \mid \forall i : 0 \leq p_i \oplus k_{1,i} \leq s-1\} \ .$$

These plaintexts are such that all s-box inputs in the first encryption rounds are in $\mathcal{S}$. We hence obtained a linear system as described in Section 3.4 but with $n^2 + n + s - 2$ unknowns: the $a'_{i,j}$'s (but $a'_{1,1}$ which is set to 1), the $k'_{2,i}$'s, and the x_i's for $i \in \mathcal{S}$ (but x_0 which is set 0). Such a system can be solved based on $t = n + 1 + \lceil (s-2)/n \rceil$ good groups of equations. The value of s must hence be selected to ensure that the plaintext subspace $\mathcal{P}_s$ is large enough to get t good groups with high confidence, while making t the smallest possible.

In order to increase our chances to actually come up with t groups of correct equation, one direction would be to select a larger set of say q groups of equations (instead of only taking the t best) and test all combinations of t groups among them. The complexity of the resulting attack will however increase dramatically with q.

So, let us assume that we have a computing power of 2^k, meaning that we can try to solve 2^k linear systems, and that we can get the leakage measurement from T encryptions. Then our approach is to request N times for the encryption of T/N different plaintexts in $\mathcal{P}_s$. For each of the T/N plaintexts, we compute the more likely candidate $\boldsymbol{\beta}$ for the s-box inputs in the second round, based on the N-averaged leakages. We thus obtained T/N groups of n equations with a corresponding confidence (*i.e.* the likelihood of the best candidate $\boldsymbol{\beta}$). Then we select the q groups for which we get the highest confidence in the best candidate $\boldsymbol{\beta}$, where q is such that $\binom{q}{t} \approx 2^k$ (that is $q = c_0 t\, 2^{k/t}$ for some $c_0 \in [\mathrm{e}^{-1}; 1]$), and we try to solve each system arising from t of these q groups. In order to make sure that a found solution is the good one, we make the system over determined. This can be done without increasing the number t of needed equation groups. Namely, we take $s \leq n+2$ in order to get $t = n + 1 + \lceil (s-2)/n \rceil = n + 2$. We thus obtain systems of $n^2 + 2n$ equations with $n^2 + n + s - 2$ unknowns. Obtaining a bad system that has a solution roughly occurs with probability $p_e \approx \left(\frac{1}{2^m}\right)^{n-s+2}$. So we take s to make this probability small, typically $s = n + 2 - 32/m$ giving $p_e \approx 2^{-32}$. For instance, for $n = 16$ and taking $s = 14$, we then have to select $t = 18$ good groups of equations from $|\mathcal{P}_s| \approx 2^{61}$ possible encryptions (which is quite enough). Another direction in increasing our chances of success would be to select the optimal averaging level.

Selecting the averaging level. We now explain how to select the averaging level N in order to optimize the success probability of the attack. Increasing the averaging level is good on the one hand to lower the noise and get better confidence in the recovered s-box inputs. On the other hand, the lower N, the greater the number T/N of different equation groups among which we can select the q best ones. To select a good tradeoff, we adopt the approach of [29] which estimates the success probability of an attack based on estimated leakage parameters. Namely we assume that the estimated paremeters $(m_\beta)_\beta$, and $(\Sigma_\beta)_\beta$ are the real leakage parameters and we simulate the attack accordingly. To simulate an attack, we fill two lists Succ and Fail by repeating the following steps:

1: $\boldsymbol{\beta}^* \leftarrow^{\$} (\mathbb{F}_{2^m})^n$
2: **for** $i = 1$ **to** n **do** $\ell_i \leftarrow^{\$} \mathcal{N}(\widehat{m}_{\beta_i^*}, \frac{1}{N}\widehat{\Sigma}_{\beta_i^*})$
3: $\mathsf{L}^{\max} \leftarrow \prod_i \max_\beta \mathsf{L}(\beta \mid \ell_i)$
4: **if** $\mathrm{argmax}_\beta \mathsf{L}(\beta \mid \ell_i) = \beta_i^*$ for every i
5: **then** add $\mathsf{L}^{\max}$ to Succ
6: **else** add $\mathsf{L}^{\max}$ to Fail

After iterating the above steps T/N times, one checks whether the q maximum values of $\mathsf{Succ} \cup \mathsf{Fail}$ include at least t value from Succ or not. In the affirmative, the simulated attack succeeded, otherwise it failed. Once the attack simulation has been performed several times, we obtain an estimation for the success probability of the attack.

Compared to a real attack experiment, the obtained success probability is affected by two differences: the actual leakage distributions $\mathcal{N}(m_{\beta_i^*}, \frac{1}{N}\Sigma_{\beta_i^*})$ are replaced by the estimated distributions $\mathcal{N}(\widehat{m}_{\beta_i^*}, \frac{1}{N}\widehat{\Sigma}_{\beta_i^*})$ and the distribution of the vector $\boldsymbol{\beta}^*$ of s-box inputs in the second round is replaced by the uniform distribution although it is not the case in practice since

the plaintexts are randomly drawn from $\mathcal{P}_s$ instead of $\{0,1\}^\ell$. However for good estimations of the leakage parameters, we expect to get a good estimation of the trade-off of choice for averaging.

Recovering remaining unknowns. For the remaining unknowns x_s, x_{s+1}, ..., x_{2^m-1} we will here again use an iterative approach that recovers them one by one. For the sake of clarity, we assume that the linear layer is such that the matrix A has a column j_0 with no zero coefficients. Then our approach is to take a random plaintext p in $\mathcal{P}_s$ and to set its j_0th coordinates to $s \oplus k_{1,j_0}$ so that the j_0th s-box inputs equals s. By definition, the ith s-box input in the second round satisfies

$$\beta_i^* = a_{i,1}\, x_{t_1} \oplus a_{i,2}\, x_{t_2} \oplus \cdots \oplus a_{i,n}\, x_{t_n} \oplus k_{2,i} \ ,$$

where $t_j \leq s-1$ for every $j \neq j_0$ and $t_{j_0} = s$. This can be rewritten

$$\beta_i^* = a_{i,j}\, x_s \oplus k_{2,i} \oplus \bigoplus_{j \neq j_0} a_{i,j}\, x_{t_j} \ . \tag{9}$$

Since we know the values of the $a_{i,j}$'s, the $k_{2,i}$'s and the x_{t_j}'s for $t_j \leq s-1$, recovering x_s amounts to recovering β_i^*. And as we cannot recover β_i^* with a 100% success probability, we use a maximum likelihood approach.

Specifically, the likelihood of each candidate value $\omega \in \mathbb{F}_{2^m}$ for x_s is initialized to 0 if $\omega \in \{x_0, x_1, \ldots, x_{s-1}\}$ (indeed $x_s \notin \{x_0, x_1, \ldots, x_{s-1}\}$ as the s-box is bijective) and to $(2^m - s)^{-1}$ otherwise. Then the leakage ℓ_i resulting from each s-box computation is used to update the likelihood of each candidate for x_s. Namely, the likelihood $\mathsf{L}(\omega)$ of the candidate ω is multiplied by the likelihood of the candidate β_i^ω for the ith s-box input, where $\beta_i^\omega = a_{i,j}\, \omega \oplus k_{2,i} \oplus \bigoplus_{j \neq j_0} a_{i,j}\, x_{t_j}$ according to (9). Doing so for every s-box, $\mathsf{L}(\omega)$ is updated by

$$\mathsf{L}(\omega) \leftarrow \mathsf{L}(\omega) \times \prod_{i=1}^{n} \mathsf{L}(\beta_i^\omega \mid \ell_i) \ ,$$

where $\mathsf{L}(\cdot \mid \ell_i)$ is computed as in (8) with $N = 1$ (since we do not use averaging here). Eventually, the likelihood vector is normalized, that is all the coordinates are divided by $\sum_\omega \mathsf{L}(\omega)$. We iterate this process for several encryptions until one likelihood value $\mathsf{L}(\omega)$ get close enough to 1. Then we deduce $x_s = \omega$, and start again with x_{s+1}, and so on until x_{2^m-1}. Note that we can stop once x_{2^m-2} since a single value remains for x_{2^m-1}.

6.3 Stage 3: Recovering k_3, k_4, ..., k_r

Eventually, the last round keys can be recovered one by one by performing any classical side-channel key recovery attack (since we now know the design of the cipher). We suggest to use a maximum likelihood approach based on the template basis.[4]

[4] Such technique is well known and pretty similar to that used in the previous section so we do not detail it here.

7 Experiments

We report hereafter the results of various simulations of the practical SCARE attack described in the previous section. Each simulated attack aims at recovering a secret cipher with classical SPN structure (such as described in Section 2.1). We consider two different settings for the cipher dimensions:

- **the (128,8)-setting:** 128-bit message block and 8-bit s-boxes, as in the AES block cipher [15] (*i.e.* $\ell = 128$, $n = 16$, $m = 8$),
- **the (64,4)-setting:** 64-bit message block and 4-bit s-boxes, as in the LED [21] and PRESENT [8] lightweight block ciphers (*i.e.* $\ell = 64$, $n = 16$, $m = 4$).

For each attack experiment, a random secret cipher is picked up. Namely, we randomly generate a full-rank $n \times n$ matrix over $\mathbb{F}_{2^m}$, a bijective m-bit s-box, and several ℓ-bit round keys. The attack succeed if it recovers an equivalent representation of the generated cipher.

In order to evaluate our attack under a realistic leakage model, we have profiled the leakage of an 8-bit s-box computation on an AVR chip.[5] The side-channel leakage was captured by the means of an electromagnetic probe and a digital oscilloscope with a sampling rate of 1G sample per second. To infer a leakage model from the measurements we made the Gaussian and independent noise assumptions. We therefore estimated the mean leakage for every s-box input value and the mean leakage for every s-box output value based on 100000 leakage traces. We then selected three leakage points for the input and three leakage points for the output. We thus obtained 256 means $(m_{1,\beta}, m_{2,\beta}, m_{3,\beta})_\beta$ for the 256 possible input values $\beta \in \{0, 1, \ldots, 255\}$ and the 256 means $(m_{4,\mu}, m_{5,\mu}, m_{6,\mu})_\mu$ for the 256 possible output values $\mu \in \{0, 1, \ldots, 255\}$. Afterwards we estimated the noise covariance matrix Σ for the selected points (*i.e.* the matrix of covariances between the 6 points after subtracting the means). A preview of the obtained parameters can be found in Appendix C. In particular we get a multivariate SNR[6] of 0.033 and univariate SNRs[7] of 0.13, 0.033, 0.099, 0.058, 0.047, and 0.051, for the different leakage points. These inferred parameters provide us with a leakage model for our attack simulations. Namely, for a given cipher with s-box S, the leakage associated to the s-box computation with input β is randomly drawn from the multivariate Gaussian distribution $\mathcal{N}(m_\beta, \Sigma)$ with mean satisfying

$$m_\beta = (m_{1,\beta}, m_{2,\beta}, m_{3,\beta}, m_{4,S(\beta)}, m_{5,S(\beta)}, m_{6,S(\beta)}) \ .$$

Stage 1. For the recovering of k_1, we implemented the Gérard and Standaert method based on the normalized Euclidean distance (see Appendix A for details). For the (128,8)-setting, we obtained a 100% success rate using a few thousands of leakage traces while for the (64,4)-setting a few hundreds were sufficient. We did not try to optimize this stage of the attack (in particular we did not use the Bayesian extension proposed in [18]) as it requires a very small amount of leakage traces compared to the next stage.

[5] ATMega 32A, 8-bit architecture, 8Mz.

[6] The multivariate SNR is defined as the ratio of the interclass generalized variance (*i.e.* the determinant of the leakage means covariance matrix) over the intraclass generalized variance (*i.e.* the determinant of the noise covariance matrix) to the power $1/d$ (where d is the dimension equal to 6 in our case).

[7] The univariate SNR is defined as the variance of the means over the variance of the noise.

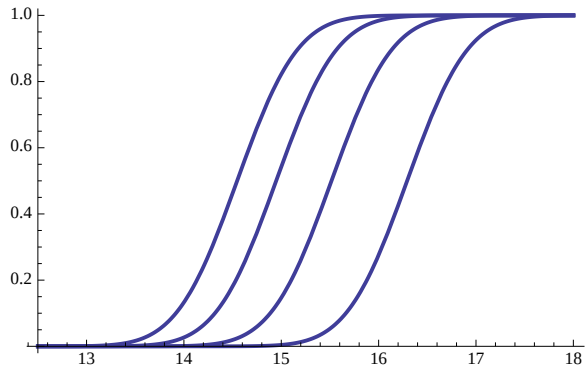

Fig. 1. Stage 2.1 for the (128,8)-setting: success rate over an increasing number of leakage traces (in $\log_2$-scale) for a computing power of 2^k with $k \in \{0, 1, 8, 32\}$.

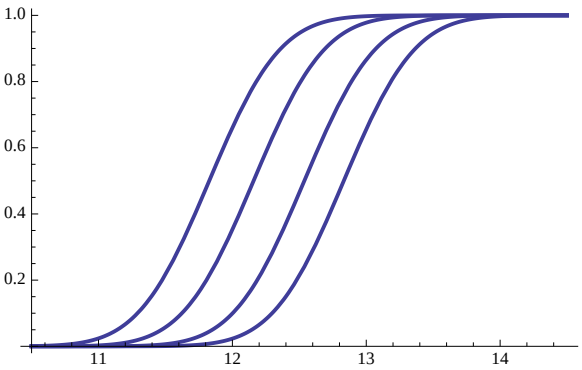

Fig. 2. Stage 2.1 for the (64,4)-setting: success rate of stage 2.1 over an increasing number of leakage traces (in $\log_2$-scale) for a computing power of 2^k with $k \in \{0, 1, 8, 32\}$.

Stage 2.1. For this stage (recovery of λ, k_2, $S(0)$, $S(1)$, ..., $S(s-1)$) we fixed the number s of s-box outputs in the system to 14 for the (128,8)-setting and to 10 for the (64,4)-setting (according to the suggested formula $s = n+2-32/m$). For both settings, we chose a precision quality parameter $q = 0.5$ for the building of the template basis and we simulated the attack for a computing power of 2^k with $k \in \{0, 8, 16, 32\}$ (*i.e.* 2^k systems among the likeliest ones are tested). The obtained success rates are plotted in Figure 1 for the (128,8)-setting and in Figure 2 for the (64,4)-setting. Each curve represents a different computing power. Naturally the leftmost curves (*i.e.* the most successful) correspond to the 2^{32} computing power and the rightmost ones to the 2^0 computing power. As one can see, with a reasonable computing power, a 100% success rate is reached with less than 2^{16} leakage traces for the (128,8)-setting, and with less than 2^{13} leakage traces for the (64,4)-setting.

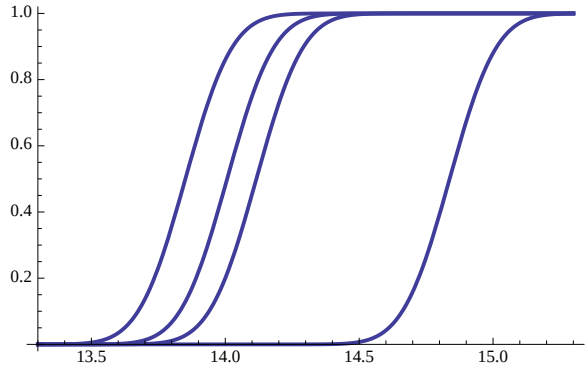

Fig. 3. Stage 2.1 for the (128,8)-setting: success rate over an increasing number of leakage measurements (in $\log_2$-scale) for a estimation quality $q = 0.1$.

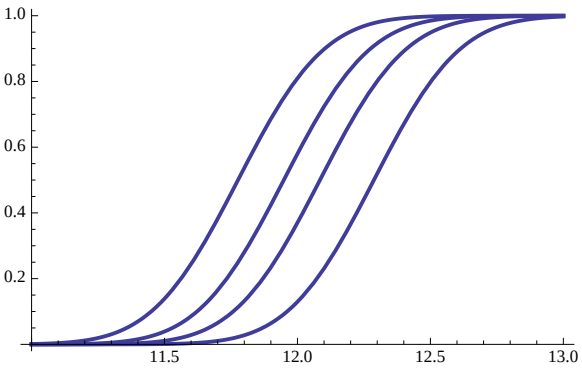

Fig. 4. Stage 2.1 for the (64,4)-setting: success rate over an increasing number of leakage measurements (in $\log_2$-scale) for a estimation quality $q = 0.1$.

For the (128,8)-setting the precision quality $q = 0.5$ makes our means estimations to converge after 1024 leakage samples per value $\beta \in \mathbb{F}_{256}$. Since 16 samples are provided per leakage trace (one for each s-box in the first round), this makes a data complexity of 2^{14} leakage traces for building the template basis. As we need around 2^{16} leakage traces to get a 100% success rate in stage 2.1 we might get a better overall attack complexity by improving the estimation precision a little bit. In order to see the kind of improvement we could get from a better estimation, we also performed attack simulations for a precision quality $q = 0.1$, implying an increase of the data complexity to 2^{17} leakage traces for the template basis. The obtained success rates are given in Figure 3. We get a 100% success rate with between 2^{14} and $2^{14.5}$ leakage traces for all computing powers except for $k = 0$ which requires 2^{15} traces.

For the (64,4)-setting, the estimated means converge after 2048 samples per value $\beta \in \mathbb{F}_{16}$, making a data complexity of 2048 for template basis. Here again we also performed attack simulations for a precision quality of $q = 0.1$ (see results in Figure 4). We get a data complexity of 2^{13} leakage traces for the template basis and around $2^{12.5}$ leakage traces for the system solving. This precision therefore seems to give the best tradeoff for the (64,4)-setting.

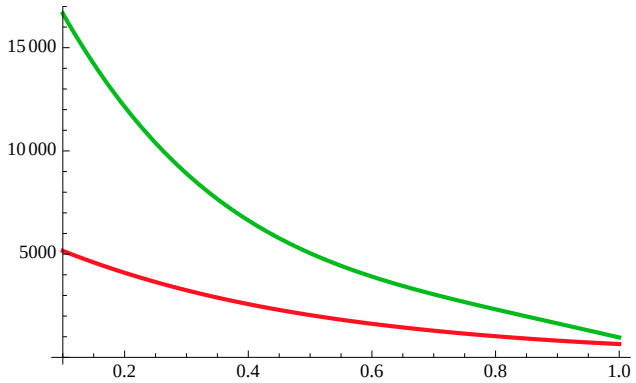

Fig. 5. Number of leakage traces to get a 90% success rate over an increasing SNR in $[0.1; 1]$ for the (128,8)-setting (green curve) and the (64,4)-setting (red curve).

In order to observe the impact of the SNR on the data complexity we performed attack simulation for which we weighted the noise covariance matrix in order to get some desired multivariate SNR between 0.1 and 1. For both settings, we fixed the estimation quality to $q = 0.5$ and the computed power to 2^{16}. Figure 5 plot the required number of leakage traces to obtain a 90% success rate with respect to the multivariate SNR. We observe a strong impact of the SNR on the attack efficiency. In particular for an SNR close to 1 our attack only requires a few thousands of traces.

Stage 2.2 and 3. The recovery of the remaining s-box outputs based on the maximum likelihood approach is very efficient. Taking a lower bound of 0.999 on the likelihood to decide that a candidate is the good one, the attack stops after 640 leakage traces on average

and reaches a 97% success rate for the (128,8)-setting (a tighter likelihood bound would yield a 100% success rate). For the (64,4)-setting, it stops after 10 leakage traces on average and reaches a 100% success rate. The high efficiency of the attack for the (64,4)-setting comes from the fact that it only has to recover 6 remaining s-box outputs. Therefore the likelihoods quickly converge.

We did not implement attack simulation for the third step but we would clearly get comparable figures than for stage 2.2, *i.e.* negligible data requirements compared to stage 2.1 which is clearly the bottleneck of our attack.

8 Discussions and Perspectives

In this paper we have described a generic SCARE attack against a wide class of SPN block ciphers. The attacker model defined in Section 3.1 assumes that colliding s-box computations can be detected from the side-channel leakage. We have first investigated the case of perfect collision detection and then we have extended our attack to deal with noisy leakages.

About the attacker model. As mentioned in Section 3.1 (Remark 2), our attacker model implicitly means that the cipher implementation processes the s-box computations in a sequential way, which is therefore more suited for software implementations. This makes sense for secret ciphers which are rarely implemented at the hardware level. Note that it is also common to use a sequential approach for the s-box computations in light-weight hardware implementations of block ciphers, and our attack naturally applies to this context. Our model further implicitly assumes that two s-box computations with the same input at two different points in the execution produce identical side-channel leakages (or identically distributed in the noisy context). Although this assumption seems fair in practice, it might not always be satisfied. It was for instance observed in [18, 30] that for some software implementations the side-channel leakage of an s-box computation may vary according to the s-box index and the target register. For such implementations, it might not be possible to detect collisions between two s-box computations at different indices. This issue can be addressed by considering each s-box index independently, which amounts to deal with the multiple s-boxes setting studied in Section 4.1 (except that we need to recover a single s-box). In this context, one only detects collisions between s-box computations at the same index. Note that our attack still assumes that s-box computations at a given index leak identically in the successive rounds.

Countermeasures to our attack. Our work shows that under a practically relevant assumption, it is possible to retrieve the complete secret design of a block cipher with a common SPN structure. This clearly emphasizes that the secrecy of the design is not sufficient to prevent side-channel attacks, and that one should include countermeasures to the implementation of secret ciphers as well. A typical choice for block cipher implementations in software is to use masking with table recomputation for the s-box (see for instance [1,24]). As studied by Roche and Lomné in [30], such a countermeasure only prevents collision detections between different cipher executions but it still allows the detection of intra-execution

collisions. In a variant of their attack against AES-like secret ciphers, Clavier *et al.* take this constraint into account in order to bypass the masking countermeasure with table recomputation [13]. Our attack in the idealized leakage model (perfect collision detection) could also be extended to work with this constraint. It would be more tricky in the presence of noise as averaging would not be an option anymore, but our attack could still be generalized using a similar approach as [30]. In order to thwart our attack, one should therefore favor masking schemes enabling the use of different masks for the different s-box computations (see for instance [9, 27]), so that intra-execution collisions would not be detectable anymore. Another common software countermeasure is operation shuffling (see for instance [23]). This countermeasure has a direct impact on our attack as it randomizes the indices of the s-box computations from one execution to another. As shown by Clavier *et al.* [13], such a countermeasure can be simply bypassed in the idealized leakage model. However, it seems more complicated to deal with in a noisy leakage model especially if combined with masking. We therefore suggest to use such a combination of countermeasure against our attack.

Perspectives. Our work opens several interesting issues for further research. First, our attack could probably be improved by using better/optimal approaches to solve the set of noisy equations arising in Stage 2.1 (see Section 6.2). One could for instance follow the approach of [17, 19] by rewriting the system as a decoding problem. Our attack could also be improved by considering a known ciphertext scenario (as *e.g.* done in [13]). On the other hand, our attack was only validated by simulations (although from a practically inferred leakage model). It would be interesting to mount the attack against a real implementation of a secret SPN cipher *e.g.* on a smart card, to check how the different steps work in practice. Another interesting direction would be to investigate extensions of our attack against protected implementations in order to determine to what extent an implementation should be protected in practice.

Acknowledgements

This work has been financially supported by the French national FUI12 project MARSHAL+ (Mechanisms Against Reverse-Engineering for Secure Hardware and Algorithms). We would like to thank Victor Lomné for providing the microcontroller side-channel traces and the anonymous reviewers for their useful comments.

References

1. Mehdi-Laurent Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. Amir Bennatan and David Burshtein. Design and Analysis of Nonbinary LDPC Codes for Arbitrary Discrete-Memoryless Channels. *IEEE Transactions on Information Theory*, 52(2):549–583, 2006.
3. Alex Biryukov and Dmitry Khovratovich. Two New Techniques of Side-Channel Cryptanalysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems, 9th International Workshop – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 195–208. Springer, 2007.

4. Andrey Bogdanov. Improved Side-Channel Collision Attacks on AES. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop – SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2007.
5. Andrey Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems, 10th International Workshop – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2008.
6. Andrey Bogdanov and Ilya Kizhvatov. Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. *IEEE Trans. Computers*, 61(8):1153–1164, 2012.
7. Andrey Bogdanov, Ilya Kizhvatov, and Andrei Pyshkin. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology, 9th International Conference on Cryptology in India – INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2008.
8. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems, 9th International Workshop – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
9. Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-Order Masking Schemes for S-Boxes. In Anne Canteaut, editor, *Fast Software Encryption, 19th International Workshop – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.
10. S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, 4th International Workshop – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–29. Springer, 2002.
11. Christophe Clavier. An Improved SCARE Cryptanalysis Against a Secret A3/A8 GSM Algorithm. In Patrick Drew McDaniel and Shyam K. Gupta, editors, *Information Systems Security, Third International Conference – ICISS 2007*, volume 4812 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2007.
12. Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Improved Collision-Correlation Power Analysis on First Order Protected AES. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2011.
13. Christophe Clavier, Quentin Isorez, and Antoine Wurcker. Complete SCARE of AES-like Block Ciphers by Chosen Plaintext Collision Power Analysis. *To Appear in INDOCRYPT 2013*, 2013.
14. Rémy Daudigny, Hervé Ledig, Frédéric Muller, and Frédéric Valette. SCARE of the DES. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, Third International Conference – ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 393–406, 2005.
15. FIPS PUB 197. *Advanced Encryption Standard*. National Bureau of Standards, November 2001.
16. FIPS PUB 46. *The Data Encryption Standard*. National Bureau of Standards, January 1977.
17. R. Fourquet, Pierre Loidreau, and Cédric Tavernier. Finding good linear approximations of block ciphers and its application to cryptanalysis of reduced round DES. In *the 6th international workshop on Coding and Cryptography (WCC 2009)*, Ullensvang, Norvège, May 2009.
18. Benoît Gérard and François-Xavier Standaert. Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems, 14th International Workshop – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2012.
19. Benoît Gérard and François-Xavier Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version. *J. Cryptographic Engineering*, 3(1):45–58, 2013.
20. Sylvain Guilley, Laurent Sauvage, Julien Micolod, Denis Réal, and Frédéric Valette. Defeating Any Secret Cryptography with SCARE Attacks. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *Progress in Cryptology, First International Conference on Cryptology and Information Security in Latin America – LATINCRYPT 2010*, volume 6212 of *Lecture Notes in Computer Science*, pages 273–293. Springer, 2010.
21. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
22. W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer Verlag, 2003.
23. P. Herbst, E. Oswald, and S. Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security – ANCS 2006*, volume 3989, pages 239–252, 2006.

24. T.S. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
25. Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, 12th International Workshop – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.
26. Roman Novak. Side-Channel Attack on Substitution Blocks. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *ACNS*, volume 2846 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2003.
27. Emmanuel Prouff and Matthieu Rivain. A Generic Method for Secure SBox Implementation. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2008.
28. Denis Réal, Vivien Dubois, Anne-Marie Guilloux, Frédéric Valette, and M'hamed Drissi. SCARE of an Unknown Hardware Feistel Implementation. In Gilles Grimaud and François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications, 8th International Conference – CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 218–227. Springer, 2008.
29. Matthieu Rivain. On the Exact Success Rate of Side Channel Analysis in the Gaussian Model. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop – SAC 2008*, volume 5381 of *Lecture Notes in Computer Science*, pages 165–183. Springer, 2009.
30. Thomas Roche and Victor Lomné. Collision-correlation attack against some 1st-order boolean masking schemes in the context of secure devices. In Emmanuel Prouff, editor, *COSADE*, volume 7864 of *Lecture Notes in Computer Science*, pages 114–136. Springer, 2013.
31. Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A Collision-Attack on AES (Combining Side Channel and Differential-Attack). In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156, pages 163–175, 2004.
32. Kai Schramm, Thomas Wollinger, and Christof Paar. A New Class of Collision Attacks and its Application to DES. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887, pages 206–222, 2003.

A Linear Collision Attacks with LDPC Decoding

We recall hereafter the basic principle of linear collision attacks using LDPC decoding [18]. Let $\{s_{i,j,\delta}\}_{i,j,\delta}$ denote a set of confidence scores to the presence of collision between $x \oplus k_{1,i}$ and $x \oplus \delta \oplus k_{1,j}$ for each pair of s-box indices (i, j) and every possible inputs differences δ. In other words, $s_{i,j,\delta}$ is the score of confidence in the equation $k_{1,i} \oplus k_{1,j} = \delta$. The authors of [18] then argue that finding the likeliest equation system from the set of scores $\{s_{i,j,\delta}\}_{i,j,\delta}$ amounts to soft-decode in a LDPC code of length $\frac{n(n-1)}{2}$ and dimension $n-1$, *i.e.* to find the closest *codeword* $(k_{1,i} \oplus k_{1,j})_{i<j}$. To that aim they use the following efficient decoding algorithm from [2] (Alg. 2 in [18]).

Algorithm 1 `LDPCSoftDecoding` procedure

Input: The set of normalized scores $\{s_{i,j,\delta}\}_{i,j,\delta}$
Output: The likeliest consistent system S

for $1 \leq i < j \leq n, \delta \in \mathbb{F}_{2^m}$
  do $P_{i,j}(\delta) \leftarrow s_{i,j,\delta}$
while$(\text{argmax}_\delta P_{1,2}(\delta), \cdots, \text{argmax}_\delta P_{n-1,n}(\delta))$ is not a codeword
  for $1 \leq i < j \leq n, \delta \in \mathbb{F}_{2^m}$
    do $P_{i,j}(\delta) \leftarrow P_{i,j}(\delta) \cdot \prod_{u \notin \{i,j\}} \sum_{\alpha \in \mathbb{F}_{2^n}} P_{i,u}(\alpha) \times P_{j,u}(\alpha \oplus \delta)$
return $(\text{argmax}_\delta P_{1,2}(\delta), \cdots, \text{argmax}_\delta P_{n-1,n}(\delta))$

In order to associate a sound confidence score to each collision, two approaches are investigated in [18], namely the use of the Euclidean distance (as proposed in [4, 5]) and

the Pearson correlation coefficient (as proposed in [25]). For each techniques, a Bayesian extension is proposed to get scores with proper meaning for the LDPC decoding algorithm (*i.e.* scores corresponding to estimated probabilities).

For our simulation we use the normalised Euclidean distance [18]. Namely, the srcore $s_{i,j,\delta}$ for the equation $k_{1,i} \oplus k_{1,j} = \delta$ is defined as

$$s_{i,j,\delta} = \frac{\max\limits_{i,j,x\neq y} \|\bar{\ell}_{i,x} - \bar{\ell}_{j,y}\| - \max\limits_{x\oplus y=\delta} \|\bar{\ell}_{i,x} - \bar{\ell}_{j,y}\|}{\sum\limits_{\delta}\left(\max\limits_{i,j,x\neq y} \|\bar{\ell}_{i,x} - \bar{\ell}_{j,y}\| - \max\limits_{x\oplus y=\delta} \|\bar{\ell}_{i,x} - \bar{\ell}_{j,y}\|\right)} \ ,$$

where $\|\cdot\|$ denotes the Euclidean norm.

B Convergence Criterion for Leakage Templates

Our convergence criterion is based on the Hotelling T^2-test which is the natural extension of the Student T-test for multinormal distributions (see for instance [22]). Let d denote the dimension of the distribution $\mathcal{N}(m_\beta, \Sigma_\beta)$ *i.e.* the number of points in an s-box leakage trace, and let $F^{-1}_{(d_1,d_2)}$ denote the quantile function of the Fisher's F-distribution with parameters (d_1, d_2) (*i.e.* $F_{(d_1,d_2)}$ is the distribution CDF). Then for any $\alpha \in [0; 1)$, the sample mean $\widehat{m}_\beta$ based on N leakage samples satisfies

$$(m_\beta - \widehat{m}_\beta)^T \widehat{\Sigma}_\beta^{-1}(m_\beta - \widehat{m}_\beta) \ \leq \ R_\alpha := \frac{d}{N-d} F^{-1}_{(d,N-d)}(\alpha) \ ,$$

with confidence α. Namely, we have a probability at least α that the estimation $\widehat{m}_\beta$ satisfies the above inequality. Note that the left term is a kind of distance between the real mean and its estimation weighted by $\widehat{\Sigma}_\beta^{-1}$. On the other hand, the upper bound R_α is independent of the noise. It results that for a given R_α, the stronger the noise, the worst the estimation. To get a more meaningful bound, we multiply both side of the inequality by the dth root of the *generalized variance* $\widehat{\sigma}^2_\beta = \det(\widehat{\Sigma}_\beta)$ to get

$$(m_\beta - \widehat{m}_\beta)^T \widehat{N}_\beta^{-1}(m_\beta - \widehat{m}_\beta) \ \leq \ (\widehat{\sigma}^2_\beta)^{1/d} R_\alpha \ , \tag{10}$$

where $\widehat{N}_\beta = \det(\widehat{\Sigma}_\beta)^{-1/d}\widehat{\Sigma}_\beta$ is the normalized form of the covariance matrix (*i.e.* it has determinant equal to 1). Here a variation of the noise level (in terms of linear increasing or decreasing of the covariance matrix coefficients) does not affect the weighted distance between the means anymore, but it directly affects the upper bound on the distance. That is why we shall define the estimation error as $\varepsilon = (\widehat{\sigma}^2_\beta)^{1/d} R_\alpha$. In practice, we want the estimation error to be substantially lower than the *interclass variance* that is the variance between the leakage means. Specifically, we want ε to be smaller than $\det(\widehat{\mathbf{S}})^{1/d}$ where $\widehat{\mathbf{S}}$ denotes the covariance matrix of the estimated means $(\widehat{m}_\beta)_\beta$. In our experiment we observe that taking a ratio $R_\alpha(\widehat{\sigma}^2_\beta / \det(\widehat{\mathbf{S}}))^{1/d}$ lower than 0.5 is a good choice.

Note that the ratio of the interclass generalized variance (*i.e.* the determinant of the leakage means covariance matrix) over the intraclass generalized variance (*i.e.* the determinant of the noise covariance matrix) to the power $1/d$ is a sound definition for the multivariate Signal-to-Noise Ratio (SNR). For this definition, our criterion is actually to have a multivariate SNR greater than R_α.

C Profiled Leakage Parameters from an AVR Chip

Figures 6–11 plot the mean values for the 6 selected leakage points from the s-box computation on the test AVR chip. We can observe from Figures 8–11 that the leakage on the output is highly correlated to its Hamming weight. For profiled noise covariance matrix, we obtained the following coefficients:

$$\Sigma = \begin{pmatrix} \mathbf{36.7} & \mathbf{-13.7} & -1.8 & 2.9 & -2.2 & -0.7 \\ \mathbf{-13.7} & \mathbf{30.7} & 0.6 & 0.7 & -0.5 & -0.1 \\ -1.8 & 0.6 & \mathbf{27.5} & -0.9 & 0.7 & 0.4 \\ 2.9 & 0.7 & -0.9 & \mathbf{38.7} & \mathbf{-27.0} & -5.4 \\ -2.2 & -0.5 & 0.7 & \mathbf{-27.0} & \mathbf{37.2} & 3.9 \\ -0.7 & -0.1 & 0.4 & -5.4 & 3.9 & \mathbf{26.2} \end{pmatrix}$$

(where the highest values are bold). In particular, it is interesting to note that there is a strong (negative) dependence between the noises in points 1 and 2 and between the noises in points 4 and 5.

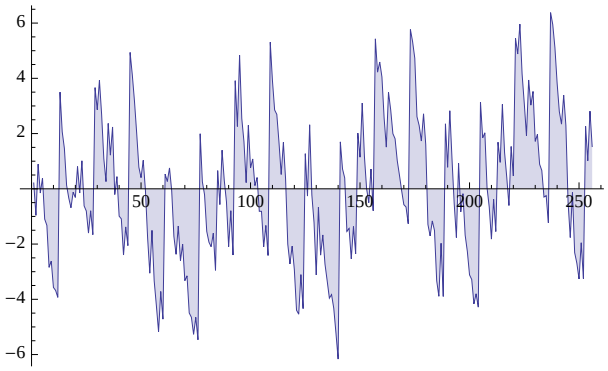

Fig. 6. Mean leakage for the first point with respect to the input s-box value from 0 to 255.

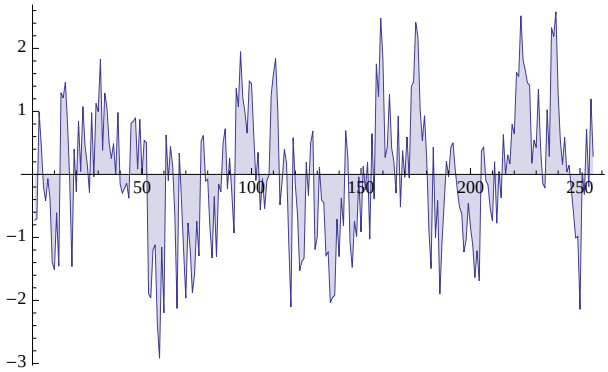

Fig. 7. Mean leakage for the second point with respect to the input s-box value from 0 to 255.

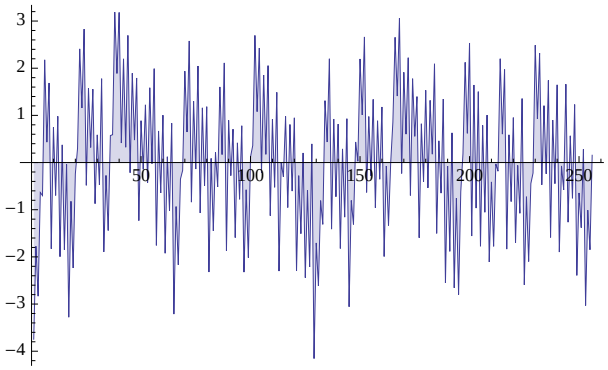

Fig. 8. Mean leakage for the third point with respect to the input s-box value from 0 to 255.

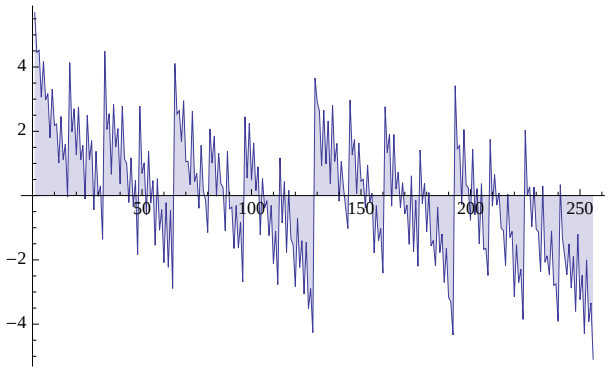

Fig. 9. Mean leakage for the fourth point with respect to the output s-box value from 0 to 255.

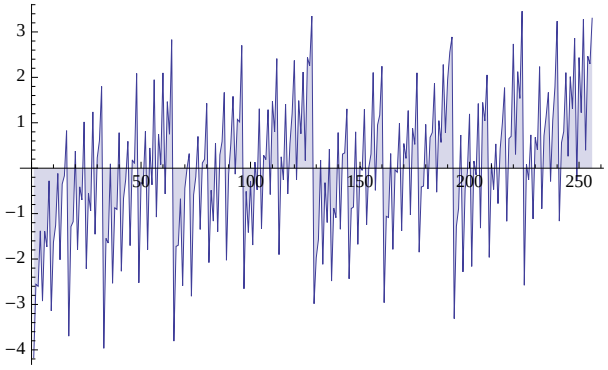

Fig. 10. Mean leakage for the fifth point with respect to the output s-box value from 0 to 255.

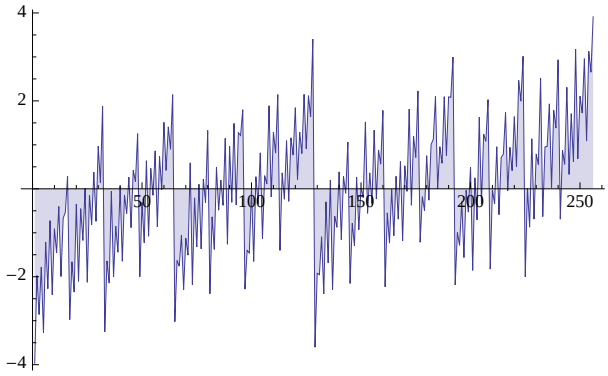

Fig. 11. Mean leakage for the sixth point with respect to the output s-box value from 0 to 255.