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 and/or software/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 [25], 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 [10] 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.* [13] 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.* [27] 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,

K. Sako and P. Sarkar (Eds.) ASIACRYPT 2013 Part I, LNCS 8269, pp. 526–544, 2013.

a SCARE attack on stream ciphers was proposed by Guilley *et al.* [19]. 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 iterated cipher composed of substitution boxes (or s-boxes), linear layers and key additions. As in [25,10], 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 [32,31,3,4,2,6,5,24,11,17]). 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 [14]. 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). In the full version of the paper [29], we further 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, so that we 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 [12], 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 6).

Paper Organization. In the first section we describe the design of target SPN block ciphers. Then we present our generic SCARE attack in Section 3. The practical SCARE attack dealing with noisy leakages is described in Section 4, and experimental results are presented in Section 5. Finally, we give some discussions and perspectives in Section 6.

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, called *round function* ρ. 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 [14]). But when they are not, 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 [15]). In the following, we only focus on the classical SPN structures. Extension of our work to Feistel schemes is provided in the full version of the paper.

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.

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 6.

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. In Section 4, 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

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

for every (i, 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 $[y_i \leftarrow S(p_i \oplus k_{1,i})] \sim [y_j \leftarrow S(p_j \oplus k_{1,j})]$ for some $i \in \mathcal{I}$, 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_j be the jth s-box input before key addition in the second round (*i.e.* w_j is the jth 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 and the $k_{2,i}$), 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 $\boldsymbol{k_2} = A \cdot \boldsymbol{k'_2}$.

As explained in Section 3.2, we must fix $a_{1,1} = 1$ in order to fix a representation among the equivalence class of the cipher. For the above system, this amounts to fixing $a'_{1,1} = 1$. 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 if a key-addition is performed after the nonlinear layer in the last round then its recovery absorbs this offset as for the other rounds. 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

$$(i \cdot n + 0, i \cdot n + 1, i \cdot n + 2, \ldots, (i+1)n - 1) \oplus k_1 \ ,$$

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 $(0, 1, 2, \ldots, n-1)$, $(n, n+1, \ldots, 2n-1)$, *etc.* Every possible s-box value thus appears in the system after $\lceil 2^m/n \rceil$ encryptions. It just remains to ask for 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 $[y_i \leftarrow S(p_{j,i} \oplus k_{j,i})] \sim [\mu_{j,i} \leftarrow S(\beta_{j,i})]$ giving $k_{j,i} = p_{j,i} \oplus \beta_{j,i}$ for every round $j \in \{3, 4, \ldots, r\}$ and every s-box index $i \in \{1, 2, \ldots, n\}$.

4 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 do 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.

4.1 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 [3,4,24,17].

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 [3], 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 [17], Gérard and Standaert further show that solving such an equation system can be written as a LDPC[1] 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.

4.2 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 $\mathcal{B} = \{(\widehat{m}_\beta, \widehat{\Sigma}_\beta)_\beta \mid \beta \in \mathbb{F}_{2^m}\}$ 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 [21]). 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:

[1] Low Density Parity Check

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

The rationale of this definition is detailed in the full version of the paper.

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 inter-class covariance matrix $\widehat{\mathbf{S}}$. Then if we have $\max_\beta R_\alpha(\widehat{\sigma}_\beta^2/\det(\widehat{\mathbf{S}}))^{1/d} \leq q$ 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 $\beta_1^*, \beta_2^*, \ldots, \beta_n^*$ 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 [9]. 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

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

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 $\text{argmax}_{\boldsymbol{\beta}}\ \mathsf{L}(\boldsymbol{\beta} \mid \boldsymbol{\ell})$ is also the one whose coordinates are the most likely *i.e.* equal to $\text{argmax}_{\beta_i}\ \mathsf{L}(\beta_i \mid \ell_i)$ 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 [28] 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}, $\ldots$, 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

$$\beta_i^* = a_{i,j}\, x_s \oplus k_{2,i} \oplus \bigoplus_{j \neq j_0} a_{i,j}\, x_{t_j} \ . \tag{3}$$

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 (3). 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 (2) 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}.

4.3 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.[2]

5 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). We consider two different settings for the cipher dimensions:

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

- **the (128,8)-setting:** 128-bit message block and 8-bit s-boxes, as in the AES block cipher [14] (*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 [20] and PRESENT [7] 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.[3] 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 the full version of the paper. In particular we get a multivariate SNR[4] of 0.033 and univariate SNRs[5] 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. 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 [17]) as it requires a very small amount of leakage traces compared to the next stage.

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

[4] 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).

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

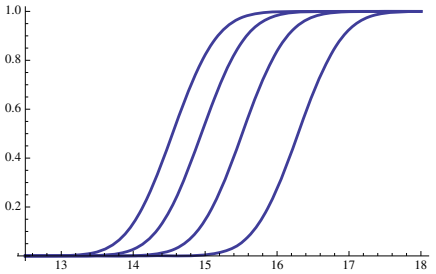

Fig. 1. Stage 2.1 for the (128,8)-setting: success rate over an increasing number of leakage traces (in $\log_2$-scale) for a computing power of 2^k with $k \in \{0, 1, 8, 32\}$

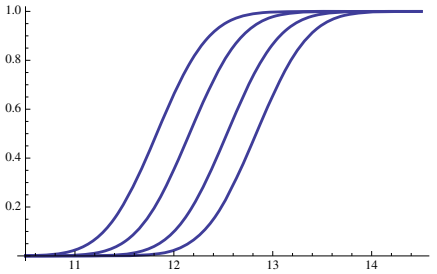

Fig. 2. Stage 2.1 for the (64,4)-setting: success rate of stage 2.1 over an increasing number of leakage traces (in $\log_2$-scale) for a computing power of 2^k with $k \in \{0, 1, 8, 32\}$

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.

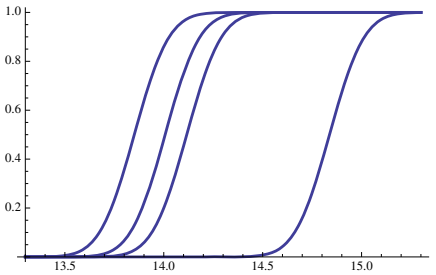

Fig. 3. Stage 2.1 for the (128,8)-setting: success rate over an increasing number of leakage measurements (in $\log_2$-scale) for a estimation quality $q = 0.1$

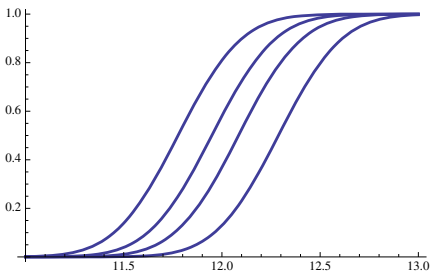

Fig. 4. Stage 2.1 for the (64,4)-setting: success rate over an increasing number of leakage measurements (in $\log_2$-scale) for a estimation quality $q = 0.1$

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.

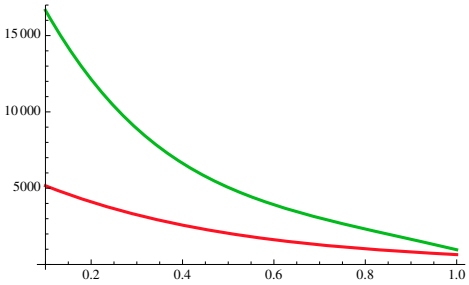

Fig. 5. Number of leakage traces to get a 90% success rate over an increasing SNR in $[0.1; 1]$ for the (128,8)-setting (green curve) and the (64,4)-setting (red curve)

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.

6 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 [17,30] 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 the full version of the paper (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 [23,1]). 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 [12]. 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 [26,8]), so that intra-execution collisions would not be detectable anymore. Another common software countermeasure is operation shuffling (see for instance [22]). 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.* [12], 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 4.2). One could for instance follow the approach of [18,16] 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 [12]). 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. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)

2. Biryukov, A., Khovratovich, D.: Two New Techniques of Side-Channel Cryptanalysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 195–208. Springer, Heidelberg (2007)
3. Bogdanov, A.: Improved Side-Channel Collision Attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 84–95. Springer, Heidelberg (2007)
4. Bogdanov, A.: Multiple-Differential Side-Channel Collision Attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008)
5. Bogdanov, A., Kizhvatov, I.: Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. IEEE Trans. Computers 61(8), 1153–1164 (2012)
6. Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)
7. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
8. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-Order Masking Schemes for S-Boxes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 366–384. Springer, Heidelberg (2012)
9. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
10. Clavier, C.: An Improved SCARE Cryptanalysis Against a Secret A3/A8 GSM Algorithm. In: McDaniel, P., Gupta, S.K. (eds.) ICISS 2007. LNCS, vol. 4812, pp. 143–155. Springer, Heidelberg (2007)
11. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved Collision-Correlation Power Analysis on First Order Protected AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 49–62. Springer, Heidelberg (2011)
12. Clavier, C., Isorez, Q., Wurcker, A.: Complete SCARE of AES-like Block Ciphers by Chosen Plaintext Collision Power Analysis. In: INDOCRYPT 2013 (to Appear, 2013)
13. Daudigny, R., Ledig, H., Muller, F., Valette, F.: SCARE of the DES. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 393–406. Springer, Heidelberg (2005)
14. FIPS PUB 197. Advanced Encryption Standard. National Bureau of Standards (November 2001)
15. FIPS PUB 46. The Data Encryption Standard. National Bureau of Standards (January 1977)
16. Fourquet, R., Loidreau, P., Tavernier, C.: 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)
17. Gérard, B., Standaert, F.-X.: Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 175–192. Springer, Heidelberg (2012)

18. Gérard, B., Standaert, F.-X.: Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version. J. Cryptographic Engineering 3(1), 45–58 (2013)
19. Guilley, S., Sauvage, L., Micolod, J., Réal, D., Valette, F.: Defeating Any Secret Cryptography with SCARE Attacks. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 273–293. Springer, Heidelberg (2010)
20. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
21. Härdle, W., Simar, L.: Applied Multivariate Statistical Analysis. Springer (2003)
22. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
23. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
24. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
25. Novak, R.: Side-Channel Attack on Substitution Blocks. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 307–318. Springer, Heidelberg (2003)
26. Prouff, E., Rivain, M.: A Generic Method for Secure SBox Implementation. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2008)
27. Réal, D., Dubois, V., Guilloux, A.-M., Valette, F., Drissi, M.: SCARE of an Unknown Hardware Feistel Implementation. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 218–227. Springer, Heidelberg (2008)
28. Rivain, M.: On the Exact Success Rate of Side Channel Analysis in the Gaussian Model. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 165–183. Springer, Heidelberg (2009)
29. Rivain, M., Roche, T.: SCARE of Secret Ciphers with SPN Structures. Cryptology ePrint Archive (2013), `http://eprint.iacr.org/`
30. Roche, T., Lomné, V.: Collision-correlation attack against some 1st-order boolean masking schemes in the context of secure devices. In: Prouff, E. (ed.) COSADE 2013. LNCS, vol. 7864, pp. 114–136. Springer, Heidelberg (2013)
31. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES (Combining Side Channel and Differential-Attack). In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
32. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)