A First-Order Leak-Free Masking Countermeasure

Houssem Maghrebi[1], Emmanuel Prouff[2], Sylvain Guilley[1,3], and Jean-Luc Danger[1,3]

[1] TELECOM-ParisTech, Crypto Group,
37/39 rue Dareau, 75 634 PARIS Cedex 13, France
{maghrebi,guilley,danger}@telecom-paristech.fr
[2] Oberthur Technologies,
71-73, rue des Hautes Pâtures 92726 Nanterre Cedex, France
e.prouff@oberthur.com
[3] Secure-IC S.A.S.,
2 rue de la Châtaigneraie, 35 576 CESSON SEVIGNÉ, France

Abstract. One protection of cryptographic implementations against side-channel attacks is the masking of the sensitive variables. In this article, we present a first-order masking that does not leak information when the registers change values according to some specific (and realistic) rules. This countermeasure applies to all devices that leak a function of the distance between consecutive values of internal variables. In particular, we illustrate its practicality on both hardware and software implementations.

Moreover, we introduce a framework to evaluate the soundness of the new first-order masking when the leakage slightly deviates from the rules involved to design the countermeasure. It reveals that the countermeasure remains more efficient than the state-of-the-art first-order masking if the deviation from the ideal model is equal to a few tens of percents, and that it is as good as a first-order Boolean masking even if the deviation is 50%.

Keywords: First-order masking, leakage in distance, leakage-free countermeasure.

1 Introduction

During the last ten years, a lot of efforts have been dedicated towards the research about side-channel attacks [9, 1] and the development of corresponding countermeasures. In particular, there have been many endeavors to develop effective countermeasures against differential power analysis (DPA) [10] attacks. DPA, and more generally side channel analysis (SCA for short), take advantage of the fact that the power consumption of a cryptographic device depends on the internally used secret key. Since this property can be exploited with relatively cheap equipments, DPA attacks pose a serious practical threat to cryptographic devices, like smart cards (ASICs) or embedded systems (DSPs, CPUs and FPGAs).

O. Dunkelman (Ed.): CT-RSA 2012, LNCS 7178, pp. 156–170, 2012.

A very common countermeasure to protect implementations of block ciphers against SCA is to randomize the sensitive variables by masking techniques. The idea of masking the intermediate values inside a cryptographic algorithm has been suggested in several papers [8, 5] as a possible countermeasure to power analysis attacks. The technique is generally applicable if all the fundamental operations used in a given algorithm can be rewritten in the masked domain. This is easily seen to be the case in classical algorithms such as the DES or AES. Masking ensures that the sensitive data manipulated by the device is masked with at least one random value so that the knowledge on a subpart of the shares (the masked data or the mask) does not give information on the sensitive data itself.

The masking can be characterized by the number of random masks used per sensitive variable. So, it is possible to give a formal definition for a high-order masking scheme: a d^{th}-order masking scheme involves $d+1$ shares. The security is reached at order d provided that any combination of d intermediate variables during the entire computation conveys no information about the sensitive variable.

We must concur that computing with $d+1$ shares without revealing information from any set of size d of intermediate values can be challenging. Some first-order masking techniques have been successfully proved to be secure against first-order SCA (1O-SCA) attacks. Nonetheless, masked implementations can always be attacked, since all shares [7] or a judicious combination [16] of them unambiguously leaks information about the sensitive variable. The construction of an efficient d^{th}-order masking scheme thus became of great interest. One sound solution has been put forward recently in [17].

In this paper, we are not concerned with higher-order masking, but devise optimised masking schemes when the leakage function is known. Typically, we show that with $d = 1$, and assuming a Hamming distance leakage function (or even some small variations of it), it is possible to zero the sensible information leaked during the registers update.

The rest of the paper is structured as follows. In Sec. 2, the concept of Boolean masking and 1O-SCA are formally defined. We also introduce some useful notations. The most critical part when securing implementations is to protect their non-linear operations (*i.e.* the sbox calls). In Sec. 3, we recall the methods which have been proposed in the literature. Then, we introduce a new and a simple countermeasure which counteracts 1O-SCA when the device leaks the Hamming distance. The security analysis is conducted for both idealized model in Sec. 4 and imperfect model in Sec. 5. The conclusion and some perspectives are in Sec. 6. Simulation results in the imperfect model are in appendix A.

2 Preliminaries

In this paper we focus on the Boolean masking countermeasure [5, 8]. Its idea is to mask the sensitive data (which depends on both plaintext and the secret key) by a *XOR* operation (denoted $\oplus$) with a random word, in order to avoid

the correlation between the cryptographic device's power consumption and the data being processed. The main difficulty of masking resides in the handling of shares of a unique intermediate variable through a non-linear function (*i.e.* the cipher sboxes). An $n \times m$ sbox (*i.e.* with n input bits and m output bits) can be described as a vectorial output mapping $F : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, or a collection of Boolean functions $F = (f_1, \cdots, f_m)$, where each f_i is defined from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. The vectorial function F is also called an (n, m)-function. A (n, m)-function F such that every element $Y \in \mathbb{F}_2^m$ has exactly 2^{n-m} pre-images by F is said to be *balanced*. Its outputs are uniformly distributed over $\mathbb{F}_2^m$ when its inputs are uniformly distributed over $\mathbb{F}_2^n$. As recalled in introduction, the manipulation of sensitive data through this function may be protected using d^{th}-order masking scheme. When such a scheme is applied, it is expected that no HO-SCA of order less than or equal to d is successful. The order relates to the number of different instantaneous leakages considered by the attack. In this paper, we focus on first-order masking scheme secure against 1O-SCA. For the rest of the paper, we adopt the following notations. Random variables are printed in capital letters (*e.g.* X), whereas their realization is noted with small letters (*e.g.* x), and their support by calligraphic letters (*e.g.* $\mathcal{X}$). The mutual information between X and Y is denoted $\mathsf{I}[X; Y]$; it measures the mutual dependency of the two variables. The Hamming weight of x, written as $\mathsf{HW}(x)$, is the number of ones in the binary word x.

3 Secure Computation against 1O-DPA Using ROMs

Let X and K denote two random variables respectively associated with some plaintext subpart values x and a secret sub-key k manipulated by a cryptographic algorithm. Let us moreover denote by Z the sensitive variable $X \oplus K$. When a *first-order* Boolean masking is involved to secure the manipulation of Z, the latter variable is randomly split into two shares M_0, M_1 such that:

$$Z = M_0 \oplus M_1 \ . \tag{1}$$

The share M_1 is usually called *the mask* and is a random variable uniformly distributed over $\mathbb{F}_2^n$. The share M_0, called *the masked variable*, plays a particular role and is built such that $M_0 = Z \oplus M_1$. Variables Z and M_1 are assumed to be mutually independent. To enable the application of a transformation S on a variable Z split in two shares, as in (1), a so-called first-order masking scheme must be designed. It leads to the processing of two new shares M_0' and M_1' such that:

$$S(Z) = M_0' \oplus M_1' \ . \tag{2}$$

Once again, the share M_1' is usually generated at random and the share M_0' is defined such that $M_0' = S(Z) \oplus M_1'$. The critical point is to deduce M_0' from M_0, M_1 and M_1' without compromising the security of the scheme (*w.r.t.* 1O-SCA). When S is linear for the law $\oplus$, then deducing M_0' is an easy task. Actually, since the relation $S(Z) = S(M_0 \oplus M_1) = S(M_0) \oplus S(M_1)$ holds, then M_0' can be simply chosen such that $M_0' = S(M_0) \oplus S(M_1) \oplus M_1'$.

When S is non-linear for the law $\oplus$ (which occurs when S is a sbox), achieving first-order security is much more difficult. The latter security indeed implies that no instantaneous leakage during the processing leaks information on Z and hence, particular attention must be paid on each elementary calculus or memory manipulation. Several solutions have been proposed to deal with this issue. Commonly, there are three strategies [15]:

1. *The re-computation method* [2,11]: this technique involves the computation of a precomputed table corresponding to the masked sbox and the generation of one or several random value(s). In its most elementary version, two random values M_1 and M'_1 are generated and the table $T^\star$ representing the function $S' : Y \mapsto S(Y \oplus M_1) \oplus M'_1$ is computed from S and stored in RAM. Then, each time the masked variable M'_0 has to be computed from the masked input $Z \oplus M_1$, the table $T^\star$ is accessed.
2. *Global Look-up Table* [15, 24]: this method also involves the computation of a precomputed look-up table, denoted $T^\star$, associated to the function $(X, Y, Y') \mapsto S(X \oplus Y) \oplus Y'$. To compute the masked variable M'_0, the global look-up table method (GLUT for short) performs a single operation: the table look-up $T^\star[Z \oplus M_1, M_1, M'_1]$. The main and important difference with the first method is that the value $S(X \oplus Y) \oplus Y'$ has been precomputed for every possible 3-tuple of values. Consequently, there is no need to re-compute before each algorithm processing and it can be stored in ROM[1]. In a simplified version (sufficient to thwart only 1O-SCA), the output mask and the input mask are chosen equal. In this case, the dimension of the table is $2n$ instead of $3n$ and the table look-up becomes $T^\star[Z \oplus M_1, M_1]$. We consider this latter version of the GLUT method in the following.
3. *The sbox secure calculation* [17,5,14,18,26]: the sbox outputs are computed *on-the-fly* by using a mathematical (*e.g.* polynomial) representation of the sbox. Then, each time the masked value M'_0 has to be computed, an algorithm performing S and parametrized by the 3-tuple (M_0, M_1, M'_1) is executed. The computation of S is split into elementary operations (bitwise addition, bitwise multiplication, ...) performed by accessing one or several look-up table(s).

Moreover, depending on the number of masks generated to protect the sbox calculations, we can distinguish two modes of protections:

1. *The single mask protection mode*: in this mode, every computation $S(Z)$ performed during the execution is protected with a single pair of input/output masks (M_1, M'_1).
2. *The multi-mask protection mode*: in this mode, the pair of masks (M_1, M'_1) is re-generated each time a computation $S(Z)$ must be protected and thus many times per algorithm execution.

In [15], the authors have shown that the choice between the three methods depends on the protection mode in which the algorithm is implemented. In fact,

[1] Recall that in embedded systems, ROM is a much less costly resource than RAM.

when the algorithm is protected in the single-mask protection mode, the re-computation method is more appropriate and induces a smaller timing/memory overhead. In the multi-mask protection mode, the re-computation method is often much more costly since the recomputation must be done before every sbox processing. Moreover, in both contexts it requires 2^n bytes of RAM to be free, which can be impossible in some very constrained environments. Concerning the sbox secure computation, it is secure against first-order SCA and does not need particular RAM allocation. However, it is often more time consuming than the first two methods and can only be used to secure sboxes with a simple algebraic structure (as *e.g.* the AES or the SEED sboxes). Regarding the GLUT method, it seems at a first glance to be the most appropriate method. Its timing performances are ideal since it requires only one memory transfer. Moreover, it can be applied in both protection modes described above. From a security point of view, the GLUT method has however a flaw since it manipulates the masked data $Z \oplus M_1$ and the mask M_1 at the same time. Actually, $Z \oplus M_1$ and M_1 are concatenated to address the look-up table $T^\star$ and thus, the value $Z \oplus M_1 \,||\, M_1$ is transferred through the bus. Since the latter variable is statistically dependent on Z, any leakage on it is potentially exploitable by a first-order DPA involving the higher-order moments of the concatenated random variable. It must however be noted that such a leakage on the address does not necessarily occurs during the bus transfers or the registers updates. Indeed, when for instance the latter ones leak the Hamming weight between an independent and random initial state and the address $Z \oplus M_1||M_1$, then the leakage is independent on Z and no first-order DPA is hence applicable. This example shows the importance of the device architecture when assessing on a countermeasure soundness. In this paper, we focus on the GLUT method. Our proposal is to benefit from all the seminal assets of the method and to additionally achieve first-order security for some realistic architectures (including the Von-Neumann ones).

3.1 Detailed Description of GLUT Method

In hardware, GLUT method can be implemented as shown in Fig. 1. This figure encompasses the masking scheme already presented in [25]. For the sake of simplicity, the linear parts, like the expansion (in DES), MixColumns (in AES), *etc.* are not represented. So, without loss of generality, we assume that the sbox S in an (n, n)-function. For instance, using AES, n can be chosen equal to 8 (straightforward tabulation of SubBytes), 4 (with the decomposition of SubBytes in $\mathrm{GF}((2^4)^2)$), or even 2 (using the $\mathrm{GF}(((2^2)^2)^2)$ tower field [19]). The registers R and M contain respectively the masked variable and the mask.

For any (n, n)-function S that must be processed in a secure way, the core principle is to define from S the lookup table representation of a new $(3n, n)$-function S' which is indexed by both the masked data and the masking material. Thanks to this new function, a masked representation $S(Z) \oplus M'_1$ of $S(Z)$ is securely derived from $Z \oplus M_1$, M_1 and the output mask M'_1 by accessing the look-up table representing S'. The size of the table can be reduced by defining the output mask as a deterministic function of the input mask. In such a case, the

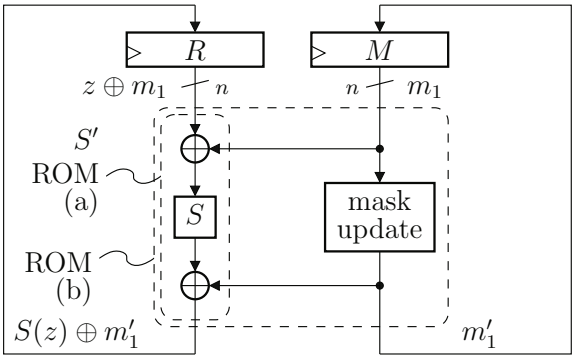

Fig. 1. First-order hardware masking implementation

ROM lookup-table represents a $(2n, n)$-function S' such that $S'(Z \oplus M, M) = S(Z) \oplus M'$, where M' is a deterministic function of M (*e.g.* $M' = M \oplus \alpha$ for some constant α).

In the first case, the ROM look-up table has $(3n)$-bit input words: the two shares and the new mask for the remasking, and one n-bit output (*e.g.* option (a) of Fig. 1). In the second case, the new masks are derived deterministically from the old ones, and thus the ROM look-up table can have only the two input shares as inputs (*e.g.* option (b) of Fig. 1). The ROM look-up table thus represents a $(2n, 2n)$-function. This is the scenario we consider in the rest of this article.

3.2 Leakage of the ROM-Based 1O-DPA Protection Implementation

During the processing of the scheme depicted in Fig. 1, we assume that only the updating of the registers R and M leak information. Indeed, since the leakage at the register level is perfectly synchronized with the system clock, it has a relatively high density of energy which is easily detectable. On the other hand, the leakage from the combinational logic is very dependent on the implementation and spreads over the time. It can be seriously reduced by taking advantage of the ROM tables [22]. In the following, we denote by L_R and L_M the leakage variables corresponding to the updating of the registers R and M respectively. We have:

$$\begin{aligned} L_R &= A(Z \oplus M_1, Z' \oplus M'_1) + N_R \\ L_M &= A(M_1, M'_1) + N_M \ , \end{aligned} \tag{3}$$

where A is a deterministic function representing the power consumption during the register updating and where N_R and N_L are two independent noises. The power consumption related to the simultaneous updating of the registers R and M equals $L_R + L_M$ and is denoted by O. In a first time, we assume that A in (3) has the following property that will be relaxed in the second part of this paper.

Property 1. For any pair (X, Y)*, we have* $A(X, Y) = A(X \oplus Y)$.

Remark 1. Many security analyses in the literature have been conducted in the so-called Hamming Distance model [12, 3]. In this model, the function A is assumed to be the Hamming distance between X and Y and thus clearly satisfies Property 1.

When A satisfies Property 1, the variable O satisfies:

$$O = A(\Delta(Z) \oplus \Delta(M)) + A(\Delta(M)) + N_R + N_M \ , \tag{4}$$

where $\Delta(Z)$ and $\Delta(M)$ respectively denote $Z \oplus Z'$ and $M_1 \oplus M_1'$.

The distribution of O (and in particular its variance) depends on the sensitive variable $\Delta(Z)$. This dependency has already been exploited in several attacks (*e.g.* [28]). In this paper, we study whether it can be broken by replacing the bitwise data masking $Z \oplus M_1$ by a new one denoted by $Z@M_1$ and by adding conditions on M_1 and M_1'.

3.3 Towards a New Masking Function

A simple solution, deeply analyzed in this paper, is to choose a function @ such that $Z \ @ \ M_1 = Z \ \oplus \ F(M_1)$ for some well chosen function F. For such a new masking function, @ is not commutative and M_1 and Z do no longer need to have the same dimension n. Only the output size of the function F must be n. In the following, we denote by p the dimension of M_1 and we assume that F is a (p, n)-function. We will see in Sec. 4.1 that p and n must satisfy some conditions for the masking to be sound. In this case, the deterministic part in (4) can be rewritten:

$$\begin{aligned} & A(Z@M_1, Z'@M_1') + A(M_1, M_1') \\ \doteq\ & A(Z \oplus Z' \oplus F(M_1) \oplus F(M_1')) + A(M_1 \oplus M_1') \\ =\ & A(\Delta(Z) \oplus F(M_1) \oplus F(M_1')) + A(\Delta(M_1)) \, . \end{aligned} \tag{5}$$

In view of (5), we deduce the two following sufficient conditions for O to be independent of $\Delta(Z)$:

1. [**Constant Masks Difference**]: $M_1 \oplus M_1'$ is constant and
2. [**Difference Uniformity**]: $F(M_1) \oplus F(M_1')$ is uniform.

To the two security conditions above, a third one must also be introduced to enable the bitwise introduction of the key on the internal state X:

3. [**Operations Commutativity**]: For every (X, M_1, K), we have:

$$X@M_1 \oplus K = (X \oplus K)@M_1 \ .$$

In the following section, we propose a way to specify M_1, M_1' and F to satisfy the three sufficient conditions. We structure our study of this new technique in two steps: the first one (*cf.* Sec. 4) is performed by assuming that A satisfies Property 1 (*i.e.* $A(X,Y) = A(X \oplus Y)$) and the second one (*cf.* Sec. 5) is conducted in an imperfect model where A satisfies $A(X,Y) = P(X,Y)$, with $P(X,Y)$ being a polynomial function in $\mathbb{R}[X_1, \cdots, X_n, Y_1, \cdots, Y_n]$ where X_i and Y_i denote the i^{th} Boolean coordinate of X and Y respectively.

4 Study in the Idealized Model

4.1 Our Proposal

Under Property 1 and as argued in the previous section, we can render the variable O independent of $\Delta(Z)$. It indeed amounts to fix the condition $M_1' = M_1 \oplus \alpha$ for some nonzero constant term α and to design a function F *s.t.* the function $Y \mapsto F(Y) \oplus F(Y \oplus \alpha)$ is uniform for this α. The latter function is usually called derivative of F with respect to α. The construction of functions F having such uniform derivatives has been highly investigated in the literature [4, Chp. 4].

We give hereafter two examples of construction of such functions F.

First Construction Proposal: we choose $p = n + 1$ and we split $\mathbb{F}_2^{n+1}$ into the direct sum $E \oplus (E \oplus \alpha)$, where E is a n-dimensional vector space and $\alpha \in \mathbb{F}_2^p$. One bijective function G from E into $\mathbb{F}_2^n$ is arbitrarily chosen and F is defined such that for every $Y \in \mathbb{F}_2^{n+1}$, we have $F(Y) = G(Y)$ if $Y \in E$ and $F(Y) = 0$ otherwise.

Second Construction Proposal: we choose $p = n + n'$ with $n' < n$ and we select one injective function G from $\mathbb{F}_2^{n'}$ into $\mathbb{F}_2^n - \{0\}$. Then, for every $(X,Y) \in \mathbb{F}_{2^{n'}} \times \mathbb{F}_{2^n} = \mathbb{F}_{2^p}$ we define $F(X,Y) = G(X) \cdot Y$ with $\cdot$ the field product over $\mathbb{F}_{2^n}$. The outputs of the (p,n)-function F are uniformly distributed over $\mathbb{F}_2^n$ (since the functions $Y \mapsto G(X) \cdot Y$ are linear and non-zero for every X). Moreover, for every non-zero element α' in $\mathbb{F}_{2^{n'}}$, the function $D_\alpha F$ defined with respect to $\alpha = (\alpha', 0) \in \mathbb{F}_{2^{n'}} \times \mathbb{F}_{2^n}$ is also balanced. Indeed, we have $D_\alpha F = (G(X) \oplus G(X + \alpha')) \cdot Y$ and, since the injectivity of G implies that $G(X) \oplus G(X \oplus \alpha')$ is never zero, the functions $Y \mapsto (G(X) \oplus G(X \oplus \alpha')) \cdot Y$ are linear and non-constant for every X.

The two constructions of F satisfy the *difference uniformity* condition defined in Sec. 3.3. The mask dimension p for the first construction is only slightly greater than the dimension n of the data to be masked. This makes it more efficient than the second construction. However, the second construction ensures that not only $D_\alpha F$ but also F is balanced. This is not mandatory to ensure the security of the countermeasure in our context where the targeted leakage is assumed to satisfy Property 1, but it can be of interest if one wishes that the data Z and Z' be masked with a uniform mask $F(M_1)$ and $F(M_1')$ respectively.

Figure 2 shows a hardware implementation of our countermeasure. The registers R and M contain respectively the masked variable $Z \oplus F(M_1)$ and the mask M_1. The mask update operation is constrained to be a $\oplus$ operation with

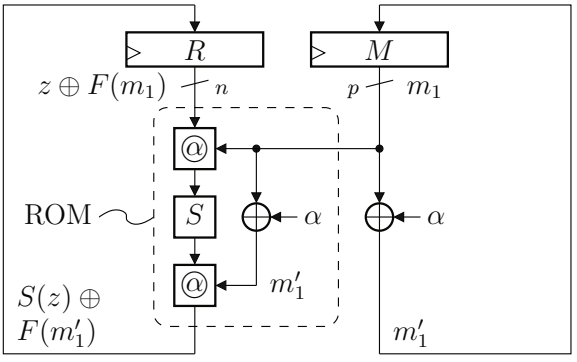

Fig. 2. Leak-free masking hardware implementation

a constant value α in order to satisfy the first condition. Consequently, every computation in the algorithm is protected with the single pair of masks (M_1, $M'_1 = M_1 \oplus \alpha$). Nonetheless, the value of M_1 changes at every computation; thus, the injected entropy in one computation is p bits. The table $T^\star$ representing the function $S' : (X, Y) \mapsto S(X@Y)@(Y \oplus \alpha) = S(X \oplus F(Y)) \oplus F(Y \oplus \alpha)$ has been pre-computed and stored in ROM. The new masked variable $S(Z) \oplus F(M'_1)$ is got by accessing the ROM table $T^\star$ as described in Fig. 2. We assume that this address is not leaking sensitive information but the leakage comes from the updating of the registers R and M following Equations (4) and (5).

4.2 Security Evaluation

In our security analysis, we assume that the attacker can query the targeted cryptographic primitive with an arbitrary number of plaintexts and obtain the corresponding physical observations, but cannot choose its queries in function of the previously obtained observations (such a model is called *non-adaptive known plaintext model* in [23]). We also assume that the attacker has access to the power consumption and electromagnetic emanations of the device and applies a first-order DPA attack but is not able to perform HO-DPA.

Regarding the leakage model, we assume that the device leaks a function A of the distance between the processed data and its initial state handled in the register (*i.e.* A satisfies Property 1). This situation is more general than the Hamming distance model, and notably encompasses the imperfect model studied in [27, Sec. 4]. The mutual information $\mathsf{I}[A(\Delta(Z) \oplus F(M_1) \oplus F(M'_1)) + A(\Delta(M)); \Delta(Z)] = 0$ since $\Delta(M)$ is constant and since $F(M_1) \oplus F(M'_1)$ is uniformly distributed over $\mathbb{F}_2^n$ and independent of $\Delta(Z)$. Hence, our construction is *leak-free* and immune against first-order attacks. Furthermore, as $A(\Delta(M))$ is constant, the mutual information

$$\mathsf{I}[A(\Delta(Z) \oplus F(M_1) \oplus F(M'_1)), A(\Delta(M)); \Delta(Z)]$$

is also null, which means that the masking countermeasure is secure against an adversary who observes the leakage in the transition from one state during

the registers update and can repeat this as many times as he wants. The adversary recovers the observations of the variable $(L_R + L_M)$ and can make all the treatments he wants (*e.g.* computation of mutual information univariate or multivariate, raise to any power the variable $L_R + L_M, \ldots$).

4.3 Application to the Software Implementation Case

Our proposal can be applied also in some particular software implementations. In some access memory schemes, the address and the value read are transferred through the same bus (*e.g.* Von-Neumann architecture). Thus, when accessing a table, the value overwrites the address and a leakage as in (4) occurs. Such access is obtained with a code such as:

```
mov  dptr, #tab
mov  acc, y
movc acc, @acc+dptr
```

In the code above, `dptr` refers to a data memory pointer and `#tab` to the address of a table stored in data. The variable `y` is assumed to contain the index of the value that must be read in table `tab`. After the third step, the accumulator register `acc` contains the value `tab[y]`. During this processing, the accumulator goes from state `y` to state `tab[y]`. Let us now assume that `tab` refers to the table T' defined in Sec. 4.1 and that `y` refers to the variable $(Z@M_1, M_1)$. If we associate the most significant bits of the accumulator `acc` to a (sub-)register R and its least significant bits to a (sub-)register M then we are in the same context as the analysis conducted in Secs. 4.1 and 4.2. A first-order DPA attack can be conducted on this register to reveal information about the sensitive data. Taking advantage from our proposal, the memory access is made completely secure under the assumption of Property 1.

5 Study in the Imperfect Model

It must first be remarked that the countermeasure proposed in the previous sections stays valid if $A(X, Y)$ can be rewritten under the form $P(X_1 \oplus Y_1, \cdots, X_n \oplus Y_n)$ with P being any polynomial defined over $\mathbb{F}_2^n$ with real coefficients.

In this section we assume that the hardware has been protected under the assumption that A satisfies Property 1, while the assumption is wrong. Namely, A was assumed to be s.t. $A(X, Y) = A(X \oplus Y)$ whereas in reality, it is a polynomial $P(X_1, \cdots, X_n, Y_1, \ldots, Y_n)$ that does not satisfy this property. In the following, we study experimentally the amount of information that the pair (L_R, L_M) defined in (3) leaks on (Z, Z') in this context where P is of (multivariate) degree d.

We recall that a polynomial of degree d in $\mathbb{R}[X_1, \cdots, X_n, Y_1, \cdots, Y_n]$ takes the following form:

$$P(X_1, \cdots, X_n, Y_1, \cdots, Y_n) = \sum_{\substack{(u,v) \in \mathbb{F}_2^n \times \mathbb{F}_2^n, \\ \mathsf{HW}(u) + \mathsf{HW}(v) \leq d}} a_{(u,v)} X_1^{u_1} \cdots X_n^{u_n} Y_1^{v_1} \cdots Y_n^{v_n} \ ,$$

where the $a_{(u,v)}$ are real coefficients. This leakage formulation is similar to that of the high-order stochastic model [21]. For example, it is shown in [16, Eqn. (3)] that $P(X_1, \cdots, X_n, Y_1, \cdots, Y_n)$ is equal to $\mathsf{HW}(X \oplus Y)$ when the coefficients $a_{(u,v)}$ satisfy:

$$a^{\mathrm{HD}}_{(u,v)} = \begin{cases} +1 & \text{if } \mathsf{HW}(u) = 1 \text{ and } v = 0, \\ +1 & \text{if } u = 0 \text{ and } \mathsf{HW}(v) = 1, \\ -2 & \text{if } \mathsf{HW}(u) = 1 \text{ and } v = u, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

In the following experiment, we compute the mutual information between (L_R, L_M) and (Z, Z') when $d \leq 2$ or 3 and when the coefficients $a_{(u,v)}$ deviate randomly from those of (6)[2]. More precisely, the coefficients $a_{(u,v)}$ are drawn at random from this law:

$$\begin{array}{l} a_{(u,v)} \sim a^{\mathrm{HD}}_{(u,v)} + \mathcal{U}(\left[-\frac{\text{deviation}}{2}, +\frac{\text{deviation}}{2}\right]) \ , \\ a_{(u,v)} = 0 \quad \text{if} \quad \mathsf{HW}(u, v) > d \ . \end{array} \tag{7}$$

The randomness lays in the uniform law $\mathcal{U}(\left[-\frac{\text{deviation}}{2}, +\frac{\text{deviation}}{2}\right])$, that we parametrize by deviation $\in \{0.1, 0.2, 0.5, 1.0\}$. The low deviation values (such as 0.1 or 0.2) are realistic in hardware, as attested by [13]; in this paper, the leakage captured by a tiny coil has been shown to differ from the Hamming distance model by 17%. We thus consider that a a deviation of $\approx [10, 20]\%$ is representative of the hardware imperfections or on the model bias by integrated probes. A deviation of 1 has the same order of magnitude as the actual coefficients in (6); it indicates that the Hamming distance model is an incorrect hypothesis. Nonetheless, this case is very unlikely: indeed, the designer of the countermeasure can be expected to know (or to have checked) that the circuit leaks approximately in Hamming distance. Eventually, the deviation 0.5 represents an intermediate case: the leakage model is in-between an approximate Hamming distance model and a full random leakage model. The computed mutual information is $\mathsf{I}[O; Z, Z']$, where $O = P(Z \oplus F(M), Z' \oplus F(M \oplus \alpha)) + N_R + P(M, M \oplus \alpha) + N_M$. Therefore O is a RV, sum of a function of Z, Z', M and of $N_R + N_M \sim \mathcal{N}(\mu_R + \mu_M, \sigma_R^2 + \sigma_M^2) = \mathcal{N}(\mu, \sigma^2)$, a normal law. The simulation parameters and the results are shown in Appendix A.

It appears that the degree d has minor influence on the leakage. The major factor is the deviation from the Hamming distance model. As expected, for low deviations (much smaller than 1, *e.g.* 10% or 20%), the one-mask countermeasure (abridged CM) of Fig. 2 definitely outperforms the CM of Fig. 1. However, in the presence of deviations close to the unity, the state-of-the-art CM remains the best. In this case, the proposed countermeasures still leaks less that an

[2] This approach clearly differs from that put forward in [6, §5.2] for comparing univariate side channel attacks (treated in the special $d = 1$ case, *i.e.* the linear case). In the later paper, the coefficients are drawn at random in the $[-1, +1]$ interval, irrespective of the sensitive data (*i.e.* the model is randomized, of expectation a "null model"), whereas in our paper, the coefficients are considered as deviations from a known non-trivial model.

unprotected design. Nonetheless, we insist that this situation is unlikely, as the deviations from the assumed Hamming distance model is of the order of one bit flip. This means that the designer has a very poor knowledge of the technology as he applies the countermeasure without checking the assumption (Property 1).

Eventually, it is noteworthy that state-of-the-art CM is even slightly improved by the imperfection of the leakage function A. This reflects the fact that the random variable $\mathsf{HW}(Z \oplus M_1) + \mathsf{HW}(M_1)$ do carry a lot of information on Z, and the noise help reduce the dependency (and thus favors the defender).

Also, both CM are equivalent for an intermediate deviation of 0.5. As this value is already quite large, we can conclude that our countermeasure is relevant even if the assumptions on the hardware leakage are extremely approximate.

6 Conclusion and Perspectives

We have presented a new masking scheme for hardware sbox implementations. We have argued that our proposal is a leak-free countermeasure under some realistic assumptions about the device architecture. The solution has been evaluated within an information-theoretic study, proving its security against 1O-SCA under the Hamming distance assumption. When the leakage function deviates slightly from this assumption (by a few tens of percent), our solution still achieves excellent results. However, if the model is very noisy (the model deviates from the Hamming distance by $\approx 50\%$), then our countermeasure remains all the same as good as state-of-the-art countermeasures.

It has been underlined (in the second construction) that some functions F have a balanced derivative in more than one direction $\alpha \neq 0$. As a perspective, we mention that this feature can be taken advantage of to increase the security of the countermeasure. Indeed, in the perfect model, the leakage remains null. However, using many αs certainly help counter model imperfections, thus reducing the leakage in this case.

Also, we underline that the proposed countermeasure can be adapted to the hypothetical case where the perfect model is not the Hamming distance $A(X, Y) = \mathsf{HW}(X \oplus Y)$, but is asymmetrical in rising and falling edges (*e.g.* $A(X, Y) = \mathsf{HW}(X \cdot \neg Y)$). Such leakages can be found in near-field electromagnetic measurements (refer to: [13] or [20, Fig. 4, left]).

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-Channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. 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)
3. Brier, É., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

4. Carlet, C.: Vectorial Boolean Functions for Cryptography (June 1 2008); Crama, Y., Hammer, P. (eds.): To appear as a chapter of the volume Boolean Methods and Models. Published by Cambridge University Press
5. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–540. Springer, Heidelberg (1999)
6. Doget, J., Prouff, E., Rivain, M., Standaert, F.X.: Univariate side channel attacks and leakage modeling. J. Cryptographic Engineering 1(2), 123–144 (2011)
7. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 221–234. Springer, Heidelberg (2010)
8. Goubin, L., Patarin, J.: DES and Differential Power Analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
9. Kocher, P.C., Jaffe, J., Jun, B.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996), `http://www.cryptography.com/timingattack/paper.html`
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
11. 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)
12. Peeters, É., Standaert, F.X., Donckers, N., Quisquater, J.J.: Improved Higher-Order Side-Channel Attacks With FPGA Experiments. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 309–323. Springer, Heidelberg (2005)
13. Peeters, É., Standaert, F.X., Quisquater, J.J.: Power and electromagnetic analysis: Improved model, consequences and comparisons. Integration, The VLSI Journal, special issue on Embedded Cryptographic Hardware 40, 52–60 (2007), doi:10.1016/j.vlsi.2005.12.0 13
14. Prouff, E., Giraud, C., Aumônier, S.: Provably Secure S-Box Implementation Based on Fourier Transform. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 216–230. Springer, Heidelberg (2006)
15. 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)
16. Prouff, E., Rivain, M., Bevan, R.: Statistical Analysis of Second Order Differential Power Analysis. IEEE Trans. Computers 58(6), 799–811 (2009)
17. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
18. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 171–184. Springer, Heidelberg (2001)
19. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
20. Sauvage, L., Guilley, S., Danger, J.L., Mathieu, Y., Nassar, M.: Successful Attack on an FPGA-based WDDL DES Cryptoprocessor Without Place and Route Constraints. In: DATE, pp. 640–645. IEEE Computer Society, Nice (2009)

21. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
22. Shah, S., Velegalati, R., Kaps, J.P., Hwang, D.: Investigation of DPA Resistance of Block RAMs in Cryptographic Implementations on FPGAs. In: Prasanna, V.K., Becker, J., Cumplido, R. (eds.) ReConFig, pp. 274–279. IEEE Computer Society (2010)
23. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
24. Standaert, F.X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World Is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010), http://www.dice.ucl.ac.be/~fstandae/PUBLIS/88.pdf
25. Standaert, F.X., Rouvroy, G., Quisquater, J.J.: FPGA Implementations of the DES and Triple-DES Masked Against Power Analysis Attacks. In: Proceedings of FPL 2006. IEEE, Madrid (2006)
26. Trichina, E.: Combinational logic design for aes subbytes transformation on masked data (2003), http://eprint.iacr.org/2003/236, not published elsewhere. e.v.trichina@samsung.com 12368 (received November 11, 2003)
27. Veyrat-Charvillon, N., Standaert, F.X.: Mutual Information Analysis: How, When and Why? In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 429–443. Springer, Heidelberg (2009)
28. Waddle, J., Wagner, D.: Towards Efficient Second-Order Power Analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)

A Simulation Results in the Imperfect Model

We assume that F has been designed thanks to the first construction presented in Sec. 4.1. Hence it is a function from $\mathbb{F}_2^{n+1}$ into $\mathbb{F}_2^n$. The mask M and the constant α are of dimension $n+1$, whereas Z is n-bit long.

The mutual information $\mathsf{I}[O+N; Z, Z']$ is represented in Tab. 1 for:

- a Gaussian noise N of standard deviation σ varying in $]0, 5]$,
- $n = 3$ bit (to speed up the computations),
- $E = \{0\} \times \mathbb{F}_2^n \subset \mathbb{F}_2^{n+1}$ and the constant α is equal to 1000 in binary, and
- $F(x_3x_2x_1x_0) = 0$ if $x_3 = 1$ or $x_2x_1x_0$ otherwise.

For each experiment just described, we also compute the mutual information for the straightforward CM of the state-of-the-art (implementation of [25] represented in Fig. 1). We also give the mutual information of this CM if the model is exactly the Hamming distance, and indicate the corresponding leakage without any countermeasure. We recall that, still with a perfect model, the mutual information for our countermeasure with (Z, Z') is null, whatever sigma.

For every $d \in \{2, 3\}$ and deviation $\in \{0.1, 0.2, 0.5, 1.0\}$, the random number generator is seeded the same. The noisy Hamming distance model is plotted for ten sets of random coefficients $a_{(u,v)}$ defined in (7), and the average is superimposed using a thick line.

Table 1. Leakage comparison of one state-of-the-art CM and our proposed CM in the imperfect model