On the Use of Shamir's Secret Sharing Against Side-Channel Analysis

Jean-Sébastien Coron[1], Emmanuel Prouff[2], and Thomas Roche[2]

[1] Tranef
jscoron@tranef.com
[2] ANSSI, 51, Bd de la Tour-Maubourg, 75700 Paris 07 SP, France
firstname.name@ssi.gouv.fr

Abstract. At CHES 2011 Goubin and Martinelli described a new countermeasure against side-channel analysis for AES based on Shamir's secret-sharing scheme. In the present paper, we exhibit a flaw in this scheme and we show that it is always theoretically broken by a first-order side-channel analysis. As a consequence of this attack, only a slight adaptation of the scheme proposed by Ben-Or *et al.* at STOC in 1988 can securely process multiplications on data shared with Shamir's technique. In the second part of this paper, we propose an improvement of this scheme that leads to a complexity $\tilde{\mathcal{O}}(d^2)$ instead of $\mathcal{O}(d^3)$, where d is the number of shares per data.

1 Introduction

The observation of a device during its execution (*e.g.* through power consumption measurements) can give information on the internal values actually manipulated by the device. Based on this idea, a powerful attack targeting symmetric cipher implementations called Differential Power Analysis (DPA for short) has been proposed by Kocher et al. in 1998 [14]. The main idea is to observe the device during the manipulation of key-dependent data (called *sensitive data* in the sequel), and to retrieve information about the key (and eventually the whole key) from this observation.

Since the introduction of DPA, and more generally of *Side Channel Analysis* (SCA for short), many works have focused either on the enhancement of such attacks or on the search of sound countermeasures. In the latter area of research, masking techniques are currently the most promising type of countermeasure. The idea is to split any sensitive variable manipulated by the device into several shares such that the knowledge of a subpart of the shares does not give information on the sensitive value itself. When the number of shares is $d+1$, the countermeasure is usually called a *d^{th}-order masking scheme*. In this case the attacker has to retrieve

information about the $d+1$ shares — *i.e.* to observe at least $d+1$ leakage points on the device — in order to gain knowledge about the targeted sensitive variable. Such an attack is called a $(d+1)^{th}$*-order SCA attack* and it has been shown that its complexity increases exponentially with the order d [4]. While some 1^{st}-order masking techniques have been proved to be secure against 1^{st}-order SCA attacks (see for instance [2,16]), the practicality of 2^{nd}-order attacks has been also demonstrated [15,17,25]. The construction of an efficient d^{th}-order masking scheme thus became of great interest. The main difficulty resides in the handling of $d+1$ shares of a unique intermediate variable through a non-linear function (*i.e.* the cipher s-boxes and more precisely the internal multiplications that are not squarings). We call this issue the *higher-order masking problem.*

State of the Art. The first scheme successfully dealing with the masking problem for any order d has been specified by Ishai, Sahai and Wagner in [12] for hardware implementations (where every internal operation is done over $\mathbb{F}_2$). In [19], this seminal result has been generalized from $\mathbb{F}_2$ to any finite field in the case of AES. It has subsequently been generalized to any block cipher in [3]. In parallel, Kim, Hong and Lim presented in [13] an improvement of [19]'s scheme which reduces, in the case of AES, the constant terms of the complexity for small orders d; it is based on the tower-field approach from [21]. The complexity of all those methods is $\mathcal{O}(d^2)$ where d is the masking order.

A common property of those previously cited works is that a Boolean masking is used to split the sensitive data. Namely, every sensitive variable A is assumed to be represented under the form of a $(d+1)$-tuple $(A_0, A_1, \cdots, A_d)$ such that $A = A_0 \oplus A_1 \oplus \cdots \oplus A_d$. However, other masking techniques have recently been investigated. On the one hand, Genelle, Prouff and Quisquater proposed in [9] a higher-order scheme based on the alternate use of Boolean masking and a *multiplicative masking* where the shares satisfy $A_0 \cdot A_1 \cdot \cdots \cdot A_d = A$. Its complexity is still $\mathcal{O}(d^2)$ but the constant terms are significantly reduced in the case study of AES. On the other hand, Goubin and Martinelli [11] and Prouff and Roche [18,20] have proposed to use Shamir's secret-sharing scheme to split the sensitive data. Starting from the same core observation as previous works [6,8], the authors' goal was to use a sharing technique with complex algebraic structure, in order to reduce the amount of sensitive information provided by the observation of the shares when involved *e.g.* in a correlation SCA. Additionally, the authors of [18,20] have shown that this way of sharing data enables the construction of masking schemes which thwart

higher-order side-channel attacks *in the presence of hardware glitches*. The security argumentation is essentially based on a link which is established between the problematic of securing s-box processings against SCA in the presence of glitches and the Multi-Party Computation problematic. In both [11] and [18,20] the practical security gain is achieved at the cost of a complexity overhead which is $\mathcal{O}(d^3)$ instead of $\mathcal{O}(d^2)$ (for Boolean masking).

Our Contribution. In this paper, we first show that the secure multiplication scheme published in [11] is flawed and that a first-order SCA can always be successfully performed against it. Then, we show that the single remaining scheme to process secure multiplications between variables shared with Shamir's technique (namely the adaptation of [1] in the SCA context), can be improved to have complexity $\tilde{\mathcal{O}}(d^2)$ instead of $\mathcal{O}(d^3)$. This is essentially done by computing polynomial evaluations with a DFT instead of a naive evaluation.

2 Goubin and Martinelli's Schemes

2.1 Preliminaries

In this paper, random variables will be denoted by capital letters (*e.g.* A) and they take their values (realisations) in $GF(2^\ell)$. Realisations of a random variable will be denoted in small-case letters (*e.g.* a). Throughout this paper, we will make the (common) assumption that the side-channel leakage emanating from the manipulation of a variable can be rightfully modelled by a deterministic function of this variable and the addition of an independent Gaussian noise. Under this assumption, an implementation is said to be *secure against d^{th}-order SCA attacks* if it satisfies the following property [5,11,18,19,22].

Definition 1 (d^{th}-order SCA security). *The implementation of an algorithm achieves d^{th}-order SCA security if no family of at most d intermediate variables is dependent on a sensitive variable.*

If a family of $j \leq d$ intermediate variables depends on a sensitive variable, then the implementation is said to have a *j^{th}-order flaw*.

Sharing/Masking. To achieve d^{th}-order security, a common countermeasure is to specify the implementation such that every sensitive variable is manipulated in a *$(d+1)^{th}$-order sharing* form. A classical choice is the Boolean masking, but other alternatives exist [9,11,18].

Masking Schemes. When the concept of $(d+1)^{\text{th}}$-order sharing is involved to protect an algorithm implementation, so-called *d^{th}-order masking schemes* are specified for each elementary operation (*e.g.* affine transformations or field multiplications). They aim at specifying how to build the sharing of the operation output from the sharing of the input(s), without introducing any j^{th}-order flaw with $j \leq d$.

In this paper we focus on a particular higher-order sharing based on Shamir's secret sharing [23]. For this technique, two families of masking schemes have been proposed in [11] and [18,20] respectively. We recall some of them along with the outlines of the sharing process itself in the next section.

2.2 Shamir's Secret Sharing Scheme

In a seminal paper [23], Shamir has introduced a simple and elegant way to split a secret $A \in GF(2^\ell)$ into n shares such that no tuple of shares with cardinality lower than a so-called *threshold* $d < n$ depends on A. Shamir's protocol consists in generating a degree-d polynomial with coefficients randomly generated in $GF(2^\ell)$, except the constant term which is always fixed to A. In other terms, Shamir proposes to associate A with a polynomial $P_A(X)$ defined such that $P_A(X) = A + \sum_{i=1}^{d} u_i X^i$, where the u_i denote random coefficients. Then, n distinct non-zero elements $\alpha_0, \ldots, \alpha_{n-1}$ are publicly chosen in $GF(2^\ell)$ and the polynomial $P_A(X)$ is evaluated in the α_i to construct a so-called (n,d)-*sharing* $(A_0, A_1, \cdots, A_{n-1})$ of A such that $A_i = P_A(\alpha_i)$ for every $i \in [0; n-1]$.

To re-construct A from its sharing, polynomial interpolation is first applied to re-construct $P_A(X)$ from its n evaluations A_i. Then, the polynomial is evaluated in 0. Those two steps indeed leads to the recovery of A since, by construction, we have $A = P_A(0)$. Actually, using Lagrange's interpolation formula, the two steps can be combined in a single one thanks to the following equation:

$$A = \sum_{i=0}^{n-1} A_i \cdot \beta_i \ , \tag{1}$$

where the constants β_i are defined as follows:

$$\beta_i := \prod_{k=0, k\neq i}^{n-1} \frac{\alpha_k}{\alpha_i + \alpha_k} \ .$$

Remark 1. The β_i can be precomputed once for all and will hence be considered as public values in the following.

Notation. The value β_i will sometimes be considered as the evaluation in 0 of the polynomial:

$$\beta_i(x) := \prod_{k=0,k\neq i}^{n-1} \frac{x+\alpha_k}{\alpha_i+\alpha_k} \ .$$

2.3 Multiplication of Shares

To define a d^{th}-order masking scheme for a block cipher implementation where each intermediate result is split with Shamir's technique, one must specify a secure method for the processing of field multiplications over $GF(2^\ell)$. Recently, two papers have been published on this issue respectively by Goubin and Martinelli [11] and by Prouff and Roche [18]. Both of them start from a multiplication protocol introduced by Ben-Or *et al.* in the context of the Multy-Party Computation Theory [1]. For this protocol to work, the number of shares n per variable must be at least $2d+1$ and for $n = 2d+1$, it is proved that it satisfies a security property encompassing the d^{th}-order SCA security. Whereas [18] is a straightforward rewriting of Ben-Or *et al.* (BGW) protocol for the SCA context, the scheme in [11] may be viewed as an efficiency improvement attempt in the SCA context. It is called GM protocol in the following. We recall hereafter the two solutions.

2.4 BGW Protocol in the SCA Context

Let us assume that A and B are two variables in $GF(2^\ell)$ that have been (n,d)-shared into $(A_i)_i$ and $(B_i)_i$ respectively, by evaluating the secret polynomials $P_A(X) = A + \sum_{1\leq j\leq d} u_j X^j$ and $P_B(X) = B + \sum_{1\leq j\leq d} v_j X^j$ in the public points α_i for $1 \leq i \leq n$. We give hereafter the adaptation of [1] in the SCA context as proposed in [18,20][1].

[1] The protocol is an improved version of the protocol originally proposed by Ben-Or *et al.* [1], due to Gennaro *et al.* in [10].

Algorithm 1 BGW's Secure Multiplication

INPUT: two integers n and d such that $n \geq 2d+1$, the (n,d)-sharings $(A_i)_i = (P_A(\alpha_i))_i$ and $(B_i)_i = (P_B(\alpha_i))_i$ of A and B respectively.
OUTPUT: the (n,d)-sharing $(P_C(\alpha_i))_i$ of $C = A \cdot B$.
PUBLIC: the n distinct points α_i, the interpolation values $(\beta_0, \cdots, \beta_{n-1})$

1. **for** $i = 1$ **to** n
2. **do** $W_i \leftarrow P_A(\alpha_i) \cdot P_B(\alpha_i)$

**** Compute a sharing $(Q_i(\alpha_j))_{j \leq d}$ of W_i with $Q_i(X) = W_i + \sum_{j=1}^{d} a_j \cdot X^j$*

3. **for** $j = 1$ **to** d **do** $a_j \leftarrow \texttt{rand}(GF(2^\ell))$
4. **for** $j = 1$ **to** n **do** $Q_i(\alpha_j) \leftarrow W_i + \sum_{k=1}^{d} a_k \cdot \alpha_j^k$

**** Compute the share $C_i = P_C(\alpha_i)$ for $C = A \cdot B$*

5. **for** $i = 1$ **to** n
6. **do** $C_i \leftarrow \sum_{j=1}^{n} Q_j(\alpha_i) \cdot \beta_j$.
7. **return** $(C_i)_i$

The completeness of Algorithm 1 is discussed in [1]. Its d^{th}-order SCA security can be straightforwardly deduced from the proof given by Ben-Or *et al.* in [1] in the secure multi-party computation context. Eventually, for $n = 2d + 1$ (which is the parameter choice which optimizes the security/efficiency overhead), the complexity of Algorithm 1 in terms of additions and multiplications is $\mathcal{O}(d^3)$.

2.5 GM Multiplication Protocol

The scheme proposed in [11] has the same asymptotic complexity as BGW but with much smaller constant terms. Indeed, the functional condition $n \geq 2d + 1$ is replaced by the minimal one $n \geq d + 1$ which enables to process the multiplication on $(d + 1, d)$-sharings of A and B (instead of $(2d + 1, d)$-sharings). We recall hereafter the proposal in [11] with the notations $\beta_{j,k}(X)$ standing for the polynomials $\beta_j(X) \cdot \beta_k(X)$ truncated by removing the terms of degree strictly greater than d.

Algorithm 2 Goubin and Martinelli's Secure Multiplication

INPUT: the $(d + 1, d)$-sharings $(A_i)_i$ and $(B_i)_i$ of A and B respectively.
OUTPUT: the $(d + 1, d)$-sharing $(C_i)_i$ of $C = A \cdot B$.
PUBLIC: the public elements α_i and the public polynomials $\beta_{j,k}(X)$.

1. **for** $j = 0$ **to** d
2. **for** $k = 0$ **to** d
3. **do** $t_{j,k} \leftarrow A_j \cdot B_k$
4. **for** $i = 0$ **to** d

5. **do** $C_i \leftarrow \sum_{j=1}^{d} \sum_{k=0}^{j-1} (t_{j,k} + t_{k,j}) \cdot \beta_{j,k}(\alpha_i) + \sum_{j=0}^{d} t_{j,j} \cdot \beta_{j,j}(\alpha_i)$

6. **return** $(C_i)_i$

The completeness of Algorithm 2 is argued in [11]. Here, we only point out that the fifth step may be rewritten:

$$C_i = \sum_{j=0}^{d} \sum_{k=0}^{d} t_{j,k} \cdot \beta_{j,k}(\alpha_i) = \sum_{j=0}^{d} \sum_{k=0}^{d} A_j \cdot B_k \cdot \beta_{j,k}(\alpha_i) \ , \tag{2}$$

in which the evaluation in α_i of the degree-d part of the polynomial $P_C(X) = P_A(X) \cdot P_B(X)$ can be clearly recognized. Hence, (2) can be written:

$$C_i = (P_A(X) \cdot P_B(X))_{|d}(\alpha_i) \ , \tag{3}$$

where the notation $(\cdot)_{|d}$ stands for the polynomial truncation obtained by suppressing all the monomials of degree strictly greater than d.

In [11], the authors assume that Algorithm 2 satisfies d^{th}-order SCA security and let the proof for future work. In the next section, we invalidates this assumption by exhibiting a first-order flaw which occurs whatever the input order d of the algorithm.

3 Attack against GM Protocol

Hereafter we show that Goubin and Martinelli's Algorithm 2 always has a first-order flaw whatever the masking order d. For clarity reasons, we first exhibit the flaw for $d = 1$ and generalize it afterward. We moreover give, in Annex A, an information theoretic evaluation of this first-order leakage for $d = 1$ and $d = 2$.

Attack Description for $d = 1$. In this case, (3) becomes:

$$C_i = A \cdot B + A \cdot V \cdot \alpha_i + B \cdot U \cdot \alpha_i \ , \tag{4}$$

where we denoted by U and V the random variables associated to the coefficients of the non-constant monomials in P_A and P_B respectively. By construction, those coefficients have been randomly generated and we hence assume that both U and V have a uniform distribution.

When $A = B = 0$, it can be checked that C_i is always null. Otherwise, if $(A, B) \neq (0, 0)$, say $A \neq 0$ w.l.o.g., then the term $A \cdot V \cdot \alpha_i$ is not null

(since $\alpha_i \neq 0$ by construction) and it depends neither on $B \cdot U \cdot \alpha_i$ nor on $A \cdot B$. As a consequence, C_i always follows an uniform distribution when $(A, B) \neq (0, 0)$. We hence deduce that C_i leaks information on (A, B) (whether it is null or not) and hence that the first-order countermeasure has a flaw.

More generally, we state in the following proposition that such first-order flaw exists for any masking order d.

Proposition 1. *Algorithm 2 always has a first-order flaw for any input parameter d.*

Proof. The flaw in Algorithm 2 has already been exhibited for $d = 1$. In the rest of the proof, we hence assume $d > 1$ and we show that, even in this case, a flaw can be exhibited. By developing (3) we get:

$$C_i = A \cdot B + \sum_{j=1}^{d} A \cdot V_j \cdot \alpha_i^j + \sum_{j=1}^{d} B \cdot U_j \cdot \alpha_i^j + \sum_{j=1}^{d-1} \sum_{k=1}^{d-j} U_j \cdot V_k \cdot \alpha_i^{j+k} \ , \quad (5)$$

where we denoted by U_j and V_k the random variables associated to the coefficients of the non-constant monomials in P_A and P_B respectively. Thanks to the law of total probability, for every $(a, b) \in GF(2^\ell)^2$ the probability $\Pr[C_i | A = a, B = b]$ satisfies:

$$\Pr[C_i | A = a, B = b] = 2^{-\ell d} \sum_{\boldsymbol{u} \in GF(2^\ell)^d} \Pr[C_i(a, b, \boldsymbol{u})] \ , \quad (6)$$

where $C_i(a, b, \boldsymbol{u})$ denotes $(C_i \mid A = a, B = b, \boldsymbol{U} = \boldsymbol{u})$ and $\boldsymbol{U}$ refers to $(U_1, \cdots, U_d)$.

Let us focus on $C_i(a, b, \boldsymbol{u})$. By definition, it satisfies:

$$C_i(a, b, \boldsymbol{u}) = a \cdot b + \sum_{j=1}^{d} b \cdot u_j \cdot \alpha_i^j + a \cdot \alpha_i^d \cdot V_d + \sum_{j=1}^{d-1} V_j \cdot \alpha_i^j \cdot (a + \sum_{k=1}^{d-j} u_k \cdot \alpha_i^k) \ . \quad (7)$$

It can hence be viewed as an affine combination of random variables V_j that all have uniform distribution and are mutually independent (by construction of the polynomial P_A). This linear combination always contains the term $\alpha_i^d \cdot a \cdot V_d$ which is independent of the other ones and has an uniform distribution as long as a is non-zero (since α_i^d itself is non-zero). Based on this observation, we can split our analysis into two cases related to the condition $a = 0$.

If $a \neq 0$, then $C_i(a, b, \boldsymbol{u})$ has uniform distribution for every b and every $\boldsymbol{u}$. This implies that $\Pr[C_i|A = a, B = b]$ is an uniform distribution.

If $a = 0$, then a sufficient condition for $C_i(a, b, \boldsymbol{u})$ to be uniform is that at least one of the terms $\sum_{k=1}^{d-j} u_k \cdot \alpha_i^k$ is non-zero when j ranges from 1 to $d-1$ which is equivalent with $(u_1, \cdots, u_{d-1}) \neq (0, \cdots, 0)$ since the α_i^j are all non-zero (by construction). When this sufficient condition is not satisfied, *i.e.* when $(u_1, \cdots, u_{d-1}) = (0, \cdots, 0)$, then we have:

$$\Pr[C_i|A = 0, B = b, u_1 = 0, \cdots, u_{d-1} = 0] = \Pr[b \cdot \alpha_i^d \cdot U_d] \ .$$

We deduce that, if $b \neq 0$, then $\Pr[C_i|A = 0, B = b, (u_1, \cdots, u_{d-1}) \neq (0, \cdots, 0)]$ and $\Pr[C_i|A = 0, B = b, (u_1, \cdots, u_{d-1}) = (0, \cdots, 0)]$ are both uniform, which implies (due to the law of total probability[2]) that $\Pr[C_i|A = 0, B = b]$ is uniform. On the other hand, if $b = 0$, then the variable $(C_i|A = 0, B = b, u_1 = 0, \cdots, u_{d-1} = 0)$ is constant and its distribution is the function which is zero everywhere except in 0 where it takes the value 1. Eventually for $(a, b) = (0, 0)$ we get:

$$\Pr[C_i = c|A = a, B = b] = \begin{cases} \frac{1}{2^\ell} - \frac{1}{2^{\ell d}} & \text{if } c \neq 0 \\ \frac{1}{2^\ell} + \frac{2^\ell - 1}{2^{\ell d}} & \text{if } c = 0 \end{cases} ,$$

which implies that the distribution $\Pr[C_i|A = 0, B = 0]$ is non-uniform. This concludes the proof since it shows that the distribution of C_i depends on the value of the pair of sensitive variables A and B. □

Remark 2. It can be observed that the distance between the two distributions that can take C_i decreases as the order increases and they actually merge when the order tends to infinity.

4 Improvement Proposal

In this section we describe a simple improvement of Algorithm 1 so that the complexity of the secure multiplication algorithm becomes $\tilde{\mathcal{O}}(d^2)$ instead of $\mathcal{O}(d^3)$. In Algorithm 1, the $\mathcal{O}(d^3)$ complexity comes from Step 4; namely this corresponds to the evaluation of a polynomial of degree d at n points α_i, which takes $\mathcal{O}(n \cdot d)$ times; since Step 4 is performed n times, the full complexity is then $\mathcal{O}(n^2 \cdot d) = \mathcal{O}(d^3)$.

Thanks to the Discrete Fourier Transform (DFT), the evaluation of a polynomial of degree $d < n$ at n points can actually be computed in time

[2] Th law of total probability states that for any r.v. X and any tuple of n r.v. $(Y_i)_i$, we have $\Pr[X] = \sum_i \Pr[X \mid Y_i] \Pr[Y_i]$.

$\tilde{\mathcal{O}}(n)$ instead of $\mathcal{O}(n^2)$. Therefore the full complexity of the algorithm becomes $\tilde{\mathcal{O}}(n^2) = \tilde{\mathcal{O}}(d^2)$ instead of $\mathcal{O}(d^3)$.

In the following we describe, in the context of our SCA secure multiplication problematic, the fast evaluation algorithm based on the DFT. In the similar context of Multi-Party Computation, such improvement was already known (see *e.g.* [7]).

4.1 Fast Polynomial Evaluation Based on DFT

Let ω a primitive n^{th} root of unity in $GF(2^\ell)$ with $n = 2^\ell - 1$. The n points α_i are defined as $\alpha_i = \omega^i$ for $0 \leq i < n$. For simplicity we restrict ourselves to a finite field of characteristic 2; however the algorithm can be generalized to any characteristic.

Given as input a polynomial

$$a(x) = \sum_{j=0}^{n-1} a_j \cdot x^j \ , \tag{8}$$

the algorithm described hereafter aims at efficiently process the values $a(\omega^i)$ for all $0 \leq i < n$. Noting that: $a(\omega^i) = a(x) \bmod (x - \omega^i)$, the values $a(\omega^i)$ can be computed using a remainder tree. As illustrated in Figure 1, the polynomial is progressively reduced modulo the polynomials $u_{i,j}(x)$, starting from the root polynomial:

$$u_{\ell,0}(x) = (x - 0) \cdot \prod_{i=0}^{n-1} (x - \omega^i) = x^{2^\ell} - x$$

downto the leaf polynomials $(x - \omega^i)$.

The DFT polynomial evaluation can actually be still improved by optimizing the ordering of the leafs ω^i so that the intermediate remainder polynomials $u_{i,j}(x)$ have a special form that enables fast modular reduction. It is shown in [26, page 573] that there exists an ordering $(\beta_0, \beta_1, \ldots, \beta_{2^\ell - 1})$ of all the elements of $GF(2^\ell)$, such that if $u_{0j}(x) := x - \beta_j$, and for $1 \leq i \leq \ell$,

$$u_{ij}(x) = u_{i-1,\,2j}(x) \cdot u_{i-1,\,2j+1}(x), \qquad 0 \leq j < 2^{\ell - i},$$

then each polynomial $u_{i0}(x)$ is an i^{th}-order linearized polynomial:

$$u_{i0}(x) = \sum_{k=0}^{i} v_{ik} \cdot x^{2^k},$$

and each polynomial $u_{ij}(x)$, $j \neq 0$, is an affine polynomial and is related to $u_{i0}(x)$ by $u_{ij}(x) = u_{i0}(x) + c_{ij}$, for some constants $c_{ij} \in GF(2^\ell)$. Since each polynomial $u_{ij}(x)$ has at most $i+2$ non-zero terms (instead of at most 2^i+1 for a polynomial of degree 2^i), modular reduction can be done in time quasi-linear in the degree of $u_{ij}(x)$, instead of quadratic time (see below).

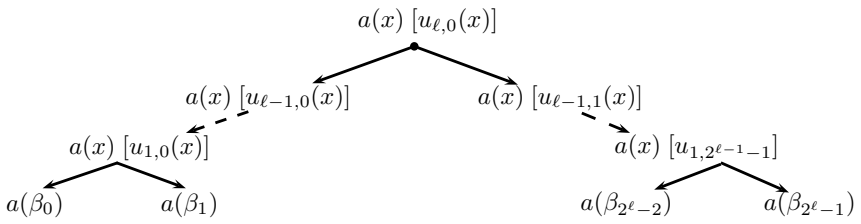

Fig. 1. Remainder tree for the computation of $a(\beta_i)$, for $0 \leq i < 2^\ell$.

Formally, the DFT computation is defined as follows:

Algorithm 3 DFT Computation

INPUT: a polynomial $a(x) = \sum_{i=0}^{n-1} a_i \cdot x^i$ over $GF(2^\ell)$, where $n = 2^\ell - 1$.
OUTPUT: the field elements $a(\beta_i)$, for $0 \leq i < 2^\ell$
PUBLIC: the public polynomials $u_{ij}(x)$

1. Let $a_{\ell 0}(x) \leftarrow a(x)$
2. **for** λ **from** $\ell - 1$ **to** 0
3. **for** $j = 0$ **to** $2^{\ell-\lambda} - 1$
4. **for** $k = 0$ **to** 1
5. **do** $a_{\lambda,\,2j+k}(x) \leftarrow a_{\lambda+1,\,j}(x) \bmod u_{\lambda,\,2j+k}(x)$
6. Return $a_{0,\,j}$ for $0 \leq j \leq 2^\ell$

When applied to process the fourth step of Algorithm 1, the polynomial $a(x)$ in (8) corresponds to $Q_i(x)$ (associating each coefficient of $a(x)$ to the corresponding coefficients in $Q_i(x)$ and setting $a_j = 0$ for all the indices $j \in [d+1; n-1]$) and the n public points α_j are assumed to be chosen equal to ω^j. In such a setting, it can then be checked that the leaf polynomials computed by the fast evaluation method described here correspond to $a(\omega^j) = Q_i(\alpha_j)$ as expected (with the special case $a(0)$ which gives the already known value W_i at Step 4 of Algorithm 1).

4.2 Complexity Analysis and Parameters' Choice

We note that the degree of the polynomials $u_{\lambda,j}$ is 2^λ, which implies that the degree of the polynomials $a_{\lambda,j}$ is at most $2^\lambda - 1$. Since the polynomials $u_{\lambda,j}$ contain at most $\lambda + 2$ non-zero terms, every modular polynomial reduction at Step 5 has complexity $\mathcal{O}(\lambda 2^\lambda) = \mathcal{O}(\ell 2^\lambda)$. For a fixed level λ there are $2 \cdot 2^{\ell-\lambda}$ such reductions, which gives a total complexity $\mathcal{O}(\ell 2^\ell)$ for a given level λ. Since there are ℓ levels the full complexity of Algorithm 3 is $\mathcal{O}(\ell^2 \cdot 2^\ell)$. With $n = 2^\ell - 1$, we obtain a complexity $\mathcal{O}(n \cdot \log^2 n) = \tilde{\mathcal{O}}(n)$ as required.[3] Note that smaller values of n are possible; namely it suffices that $n | 2^\ell - 1$. For example for AES with $GF(2^8)$, since $2^8 - 1 = 3 \cdot 5 \cdot 17$, we can take $n = 3, 5, 15, 17, 51, 85, 255$.

To select a larger value of n for a fixed field size $GF(2^\ell)$, it suffices to work in an extension field $GF(2^{\ell \cdot s})$ of $GF(2^\ell)$ for $s > 1$; then one can take $n = 2^{\ell \cdot s} - 1$. The complexity of Algorithm 3 is still $\mathcal{O}(n \cdot \log^2 n)$ operations in the extension field $GF(2^{\ell \cdot s})$. Each operation in $GF(2^{\ell \cdot s})$ can be computed with $\mathcal{O}(s^2) = \mathcal{O}(\log^2 n)$ operations in $GF(2^\ell)$. Therefore the complexity of Algorithm 3 becomes $\mathcal{O}(n \cdot \log^4 n)$, which is still $\tilde{\mathcal{O}}(n)$.

4.3 Security of the Improved Multiplication Algorithm

Since in Algorithm 1 this polynomial evaluation step is performed n times, the full complexity becomes $\mathcal{O}(n^2 \cdot \log^4 n) = \tilde{\mathcal{O}}(n^2)$ instead of $\mathcal{O}(n^3)$. In the multi-party computation setting, the new algorithm is still secure against a coalition of up to $t < n/2$ players; namely the polynomial evaluation step at Step 4 in Algorithm 1 is performed *locally* by each player; therefore changing the polynomial evaluation algorithm does not modify the security property of the algorithm. In the context of side-channel analysis, with $n = 2d + 1$, the algorithm is therefore still secure against a d-th order attack.

5 Conclusion

Several works argued on the importance of identifying new sharing techniques that minimize the amount of sensitive information extractable from the family of shares in a SCA context. This is indeed of particular importance since such a sharing, combined with noise, would be able to resist to any higher-order side-channel attack in practice, even when parametrized with small sharing orders (*e.g.* 2 or 3). The polynomial sharing introduced by Shamir is a promising candidate. However, it remains to define

[3] We write $f(\lambda) = \tilde{\mathcal{O}}(g(\lambda))$ if $f(\lambda) = \mathcal{O}(g(\lambda) \log^k g(\lambda))$ for some $k \in \mathbb{N}$.

efficient algorithms able to operate on data shared with this technique without introducing key-dependent leakages of order lower than the sharing order. Until this work, there were essentially two algorithm proposals to securely perform a multiplication between two shared data: one proposed by Goubin and Martinelli at CHES 2011 and one adapted from an algorithm by Ben-Or *et al.* at STOC in 1988. In the present paper, we showed that the first proposal is flawed (more precisely is always broken by a first-order SCA) and we improved the complexity of the second proposal from $\mathcal{O}(d^3)$ to $\tilde{\mathcal{O}}(d^2)$, where d is the number of shares per data. We think that those results are a first promising step toward efficient methods to process on data shared with Shamir's secret sharing in embedded systems.

References

1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In J. Simon, editor, *STOC*, pages 1–10. ACM, 1988.
2. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
3. C. Carlet, L. Goubin, E. Prouff, M. Quisquater, and M. Rivain. Efficient higher-order masking schemes for s-boxes. In *FSE*, pages 366–384, 2012.
4. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In *Second AES Candidate Conference – AES 2*, Mar. 1999.
5. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.
6. N. Courtois and L. Goubin. An Algebraic Masking Method to Protect AES against Power Attacks. In D. Won and S. Kim, editors, *Information Security and Cryptology – ICISC 2005*, volume 3935 of *Lecture Notes in Computer Science*, pages 199–209. Springer, 2006.
7. I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.
8. G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain. Affine masking against higher-order side channel analysis. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280. Springer, 2010.
9. L. Genelle, E. Prouff, and M. Quisquater. Thwarting Higher-Order Side Channel Analysis with Additive and Multiplicative Maskings. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International*

Workshop – CHES 2011, volume 6917 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2011.
10. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.
11. L. Goubin and A. Martinelli. Protecting aes with shamir's secret sharing scheme. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2011.
12. Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
13. H. Kim, S. Hong, and J. Lim. A Fast and Provably Secure Higher-Order Masking of AES S-Box. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems, 13th International Workshop – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2011.
14. P. Kocher, J. Jaffe, and B. Jun. Introduction to Differential Power Analysis and Related Attacks. Technical report, Cryptography Research Inc., 1998.
15. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
16. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In H. Handschuh and H. Gilbert, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
17. E. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater. Improved Higher-order Side-Channel Attacks with FPGA Experiments. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2005.
18. E. Prouff and T. Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.
19. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
20. T. Roche and E. Prouff. Higher-order glitch free implementation of the aes using secure multi-party computation protocols. *Journal of Cryptographic Engineering*, pages 1–17, 2012.
21. A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In E. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2001.
22. K. Schramm and C. Paar. Higher Order Masking of the AES. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.
23. A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.
24. F.-X. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard. The World Is Not Enough: Another Look on Second-

Order DPA. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

25. J. Waddle and D. Wagner. Toward Efficient Second-order Power Analysis. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.
26. Y. Wang and X. Zhu. A fast algorithm for the fourier transform over finite fields and its vlsi implementation. *IEEE Journal on Selected Areas in Communications*, 6(3):572–577, 1988.

A Mutual Information Study of Goubin and Martinelli's scheme 1^{st}-order flaw

We have seen in Section 3 that Goubin and Martinelli's proposal possesses a first-order flaw whatever the masking order d of their scheme. For the study of this flaw to be complete, we propose here an information theoretic evaluation of the information leakage with respect to the noise standard deviation and d. We moreover compare the quantity of sensitive information in the flaw with that contained in the observation of the $d+1$ shares build thanks to Shamir's sharing for $d = 1$ and $d = 2$.

To quantify the amount of leaking information, we modelled the relationship between the physical leakage and the value of the variable processed at the time of the leakage. For such a purpose, we associated each $(d+1)$-tuple of shares $(A_0, \cdots, A_d)$ with a $(d+1)$-tuple of leakages $\mathbf{L} = (L_0, \cdots, L_d)$ s.t. $L_j = \mathrm{HW}(A_j) + \mathcal{N}_j$, with $\mathcal{N}_j$ an independent Gaussian noise with mean 0 and standard deviation σ. We use the notation $\mathbf{L} \hookleftarrow (A_0, \cdots, A_d)$ to refer to this association. In the case of the first-order flaw exhibited in Section 3, the leakage $\mathbf{L}$ is univariate and satisfies $\mathbf{L} \hookleftarrow \mathrm{HW}(C_i) + \mathcal{N}$ with C_i being defined as in (3). In that case, the sensitive information in the product $C = (AB)$.

To evaluate the information revealed by each tuple of shares for the polynomial masking technique, we computed the mutual information[4] $I(A, \mathbf{L})$ between the sensitive variable A and $\mathbf{L}$. Similarly, in the case of Goubin and Martinelli's scheme, we computed the mutual information $I(AB, \mathbf{L})$ between the sensitive variable (AB) and $\mathbf{L}$. We list hereafter

[4] As shown in [24], the number of measurements required to achieve a given success-rate in a maximum likelihood attack can be expressed as a function of the mutual information evaluation and equals $c \times I(A, \mathbf{L})^{-1}$, where c is a constant related to the chosen success-rate.

the leakages we considered and the underlying leaking variables:

$$\text{(2, 1)-sharing leakage:} \quad \mathbf{L} \hookleftarrow (P_A(\alpha_1), P_A(\alpha_2)) \ . \tag{9}$$

$$\text{(3, 2)-sharing leakage:} \quad \mathbf{L} \hookleftarrow (P_A(\alpha_1), P_A(\alpha_2), P_A(\alpha_3)) \ . \tag{10}$$

$$\text{Flaw in Alg. 2 for } d = 1: \quad \mathbf{L} \hookleftarrow (C_i = (P_A \cdot P_B)_{|1}(\alpha_i)) \ . \tag{11}$$

$$\text{Flaw in Alg. 2 for } d = 2: \quad \mathbf{L} \hookleftarrow (C_i = (P_A \cdot P_B)_{|2}(\alpha_i)) \ . \tag{12}$$

Figure A summarizes the information theoretic evaluation for each leakage (9) to (12). For d equal to 1 or 2, it can be observed that the amount of information revealed by the $d + 1$ sharing elements is greater than that revealed by the 1^{st}-order flaw up to a certain amount of noise. As a matter of fact, the first-order flaw is less impacted by the noise than the 2^{nd}-order and 3^{rd}-order leakages. Hence, for any choice of input parameter d in Algorithm 2 and for any Shamir's sharing order $d' > d$, there exists a noise standard deviation σ s.t. the first-order flaw leaks more sensitive information than the d'-tuple of Shamir's shares. For example, for $d = 1$, then the first-order flaw in Algorithm 2 leaks more information than any d'^{th}-order sharing with $d' > 1$ as long as $\sigma > 3.7$. We also emphasize that the traces' resynchronization issue and the computational complexity of the processings make higher-order SCA attacks much more difficult to mount in practice than first-order ones. As a consequence, the first-order flaw is even more important from a practical point of view than suggested in Fig. A.

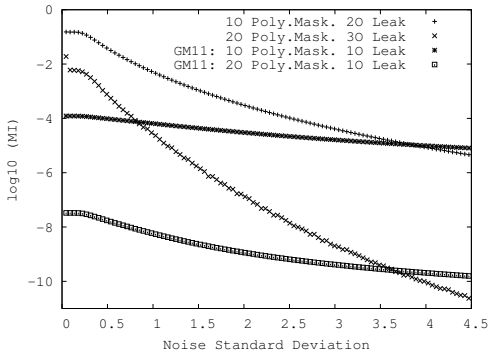

Fig. 2. Mutual information ($\log_{10}$) between the leakage and the sensitive variable over an increasing noise standard deviation (x-axis).