# Protecting AES with Shamir's Secret Sharing Scheme

Louis Goubin[1] and Ange Martinelli[1,2]

[1] Versailles Saint-Quentin-en-Yvelines University
Louis.Goubin@prism.uvsq.fr
[2] Thales Communications
jean.martinelli@fr.thalesgroup.com

**Abstract.** Cryptographic algorithms embedded on physical devices are particularly vulnerable to Side Channel Analysis (SCA). The most common countermeasure for block cipher implementations is masking, which randomizes the variables to be protected by combining them with one or several random values. In this paper, we propose an original masking scheme based on Shamir's Secret Sharing scheme [22] as an alternative to Boolean masking. We detail its implementation for the AES using the same tool than Rivain and Prouff in CHES 2010 [16]: multi-party computation. We then conduct a security analysis of our scheme in order to compare it to Boolean masking. Our results show that for a given amount of noise the proposed scheme - implemented to the first order - provides the same security level as $3^{rd}$ up to $4^{th}$ order boolean masking, together with a better efficiency.

**Keywords:** Side Channel Analysis (SCA), Masking, AES Implementation, Shamir's Secret Sharing, Multi-party computation.

## 1 Introduction

*Side Channel Analysis* is a cryptanalytic method in which an attacker analyzes the *side channel leakage* (*e.g.* the power consumption, ...) produced during the execution of a cryptographic algorithm embedded on a physical device. SCA exploits the fact that this leakage is statistically dependent on the intermediate variables that are involved in the computation. Some of these variables are called *sensitive* in that they are related to a secret data (*e.g.* the key) and a known data (*e.g.* the plain text), and recovering information on them therefore enables efficient key recovery attacks [11, 3, 8].

The most common countermeasure to protect implementations of block ciphers against SCA is to use masking techniques [4, 9] to randomize the sensitive variables. The principle is to combine one or several random values, called *masks*, with every processed sensitive variable. Masks and masked variables propagate throughout the cipher in such a way that any intermediate variable is independent of any sensitive variable. This method ensures that the leakage at an instant

$t$ is independent of any sensitive variable, thus rendering SCA difficult to perform. The masking can be improved by increasing the number of random masks that are used per sensitive variable. A masking that involves $d$ random masks is called a *$d^{th}$-order masking* and can always be theoretically broken by a $(d+1)^{th}$-*order SCA*, namely an SCA that targets $d+1$ intermediate variables at the same time [13, 21, 18]. However, the noise effects imply that the complexity of a $d^{\text{th}}$-order SCA increases exponentially with $d$ in practice [4]. The *$d^{th}$-order SCA resistance* (for a given $d$) is thus a good security criterion for implementations of block ciphers. In [17] Rivain and Prouff give a general method to implement a $d^{\text{th}}$-order masking scheme to the AES using secure Multi-Party Computation. Instead of looking for perfect theoretical security against $d^{\text{th}}$-order SCA as done in [17], an alternative approach consists in looking for practical resistance to these attacks. It may for instance be observed that the efficiency of higher-order SCA is related to the way the masks are introduced to randomize sensitive variables. The most widely studied masking schemes are based on *Boolean masking* where masks are introduced by exclusive-or (XOR). First order boolean masking enables securing implementations against first-order SCA quite efficiently[1, 16]. It is however especially vulnerable to higher-order SCA [13] due to the intrinsic physical properties of electronic devices. Other masking schemes may provide better resistance against these attacks using various operations to randomize sensitive variables. This approach will be further investigated in this paper.

*Related work.* In [25, 6], the authors propose to use an affine function instead of just XOR to mask sensitive variables, thus improving the security of the scheme for a low complexity overhead. However, this countermeasure is developed only to the $1^{\text{th}}$ order and it is not clear how it can be extended to higher orders. In [10, 16] the authors explain how to use secure Multi-Party Computation to process the cipher on shared variables. They use a sharing scheme based on XOR, implementing boolean masking to any order to secure the AES block cipher. At last, in [19], Prouff and Roche give a hardware oriented glitch free way to implement block ciphers using Shamir's Secret Sharing scheme and Ben-Or *et al.* secure multi-party computation [2] protocol operating on $2d+1$ shares to thwart $d$-th order SCA.

*Our contribution.* In this paper, we propose to combine both approaches in implementing a masking scheme based upon Shamir's Secret Sharing scheme [22], called *SSS masking* and processed using Multi-party Computation methods. Namely, we present an implementation of the block cipher such that every 8-bit intermediate result $z \in \mathrm{GF}(256)$ is manipulated under the form $(x_i, P(x_i))_{i=0..d}$, where $x_i \in \mathrm{GF}(256)^*$ is a random value generated before each new execution of the algorithm and $P(X) \in \mathrm{GF}(256)[X]$ is a polynomial of degree $d$ such that $P(0) = z$. Our scheme maintains the same compatibility as Boolean masking with the linear transformations of the algorithm. Moreover, the fact that the masks are never processed alone prevents them to be targeted by a higher-order SCA, thus greatly improves the resistance of the scheme to such attacks.

*Organization of the paper.* We fist recall Shamir's secret sharing scheme in Sect. 2. In Sect. 3, we show how SSS masking can be applied to the AES and give some implementation results. Sect. 4 analyzes the resistance of our method to high-order SCA and Sect. 5 concludes the paper.

## 2 Shamir's Secret Sharing scheme

In some cryptographic context ones may need to share a secret between (at least) $d$ users without any $k < d$ users being able to recover the secret alone. In [22] Shamir exposes the problematic and gives a secret sharing scheme using polynomial interpolation as a recovery mean. Namely, every user has a pair $(x_i, P(x_i))_{x_i \neq 0}$, where $P$ is a polynomial of degree $k$, and the secret is given by $P(0)$. In this configuration, one needs at least $k+1$ shares to recover $P$, then $P(0)$. We recall hereafter the sharing and reconstruction algorithms for a value of $k = d-1$, operating on $n$-bits words. With these parameters, in order to share a secret $a_0$ into $d$ shares, one needs to choose $d-1$ random numbers $(a_{d-1}, \ldots, a_1)$ to construct the polynomial $P(x) = a_{d-1} \cdot x^{d-1} + a_{d-2} \cdot x^{d-2} + \cdots + a_1 \cdot x + a_0$. Every share $i$ is then given by $(x_i, y_i)$ where $y_i = P(x_i)$, and the $x_i$'s are all distinct and non-zero. Formally we have Algorithm 1.

**Algorithm 1** Shamir's Secret Sharing scheme

INPUT: A secret $a_0$, random values $(x_i)_{i=0..d-1}$

OUTPUT: Shares $(x_i, y_i)_{i=0..d-1}$

```
$(a_i)_{i=1..d-1} \leftarrow$ Rand($n$)
for $i = 0$ to $d$ do
    $y_i \leftarrow a_{d-1} \cdot x_i^{d-1} + a_{d-2} \cdot x_i^{d-2} + \cdots + a_1 \cdot x_i + a_0$
return $(x_i, y_i)_{i=0..d-1}$
```

The reconstruction step is directly derived from polynomial interpolation and proceeds as follows:

$$a_0 = \sum_{0}^{d} y_i \cdot \beta_i \tag{1}$$

where each $\beta_i$ is a precomputed value such that $\beta_i = \displaystyle\prod_{j=0, j \neq i}^{d} \frac{-x_j}{x_i - x_j}$.

## 3 Higher order masking of AES

The AES block cipher iterates a round transform composed of four stages: `AddRoundKey`, `ShiftRows`, `MixColumn` and `SubByte`. In this section we will show how to securely mask the different layers at order $d$ using *SSS masking*.

### 3.1 Masking Field Operations

In order to secure the AES there are five field operations that must be protected: the addition with an unmasked constant, the addition with a masked variable, the multiplication by a scalar, the square, and the multiplication between two shared variables. Moreover, as the AES Sbox is the composition of the inversion in GF(256) and an affine function modulus the polynomial $X^8+1$, this affine transform also has to be secured.

Let $b$ be a sensitive variable shared as $(x_i, y_i)_{i=0..d}$ following Shamir's secret sharing scheme. Addition with an unmasked constant $u$ can be directly computed by XORing $u$ to the second component of the shares, such as:

$$(x'_i, y'_i) \leftarrow (x_i, y_i \oplus u).$$

Let $(x_i, u_i)_{i=0..d}$ be the shared representation of a variable $u$. The shared representation of the addition $b \oplus u$ is obtained as: $(x'_i, y'_i) \leftarrow (x_i, y_i \oplus u_i)$. Similarly the multiplication by a scalar $p$ is computed as: $(x'_i, y'_i) \leftarrow (x_i, y_i \cdot p)$.

While working in a field of characteristic 2 squaring is GF(256)-linear: $(x'_i, y'_i) \leftarrow (x_i^2, y_i^2)$. Remark that the coefficient $x'_i \neq x_i$. This matter must be taken in account in the computation as shown in Algorithm 4.

The product of two variables protected by secret sharing cannot be solved with the linear property of the transformation, as multiplying two polynomials with the same degree gives a polynomial with degree double of the original polynomial. Two different approaches can be studied. The first, developed in [19], is to use the proven secure multi-party computation scheme of [2] operating on $2d+1$ shares to process the product. However this approach has a very high complexity. The second possibility is to exploit the context of side channel countermeasure that allows us to compute values unknown in classical multi-party computation in order to improve the complexity at the loss of the security proof. We give Algorithm 2 to compute secure shared field multiplication.

**Algorithm 2** Share multiplication `SecMult`

INPUT: Shared representation of $b$, $(x_i, y_i)_{i=0..d}$ and $u$, $(x_i, w_i)_{i=0..d}$

OUTPUT: Shares $(x_i, y'_i)_{i=0..d}$ representing the product of $b$ and $u$

```
1. for $j = 0$ to $d$ do
2.     for $k = 0$ to $d$ do
3.         $z_{j,k} \leftarrow y_j \cdot w_k$
4. for $i = 0$ to $d$ do
5.     $(x_i, y'_i) \leftarrow \left( x_i, \left( \sum_{j=1}^{d} \sum_{0 \leq k < j} (z_{j,k} \oplus z_{k,j}) \cdot \beta_{j,k}(x_i) \right) + \sum_{j=0}^{d} z_{j,j} \cdot \beta_{j,j}(x_i) \right)$
6. return $(x_i, y'_i)_{i=0..d}$
```

where the $\beta_{j,k}(x_i)$ are precomputed values defined as follows.

Recall that $\beta_j(x) = \prod_{l=0, l \neq j}^{d} \frac{x - x_l}{x_j - x_l}$. We have

$$\begin{aligned} \beta_j(x) \cdot \beta_k(x) &= \prod_{l=0, l \neq j}^{d} \frac{x - x_l}{x_j - x_l} \cdot \prod_{m=0, m \neq k}^{d} \frac{x - x_m}{x_k - x_m} \\ &= \alpha_{2d} x^{2d} + \cdots + \alpha_d x^d + \cdots + \alpha_1 x + \alpha_0 \end{aligned} \tag{2}$$

We then define $\beta_{j,k}(x) = \beta_{k,j}(x) = \alpha_d x^d + \cdots + \alpha_1 x + \alpha_0$.

**Proposition 1.** *Algorithm 2 holds because the polynomial* $P(x) = \sum_{j=0}^{d} \sum_{k=0}^{d} y_j \cdot w_k \cdot \beta_{j,k}(x)$ *is such that:* $\begin{cases} degree(P) = d \\ P(0) = b \cdot u \\ \forall x \in \{x_i\}_{i=0..d}, \; P(x_i) = y'_i \end{cases}$

*Proof.* – By construction of the $\beta_{k,j}(x)$, $degree(P) = d$.

- Let $b, u$ be shared respectively in $(x_i, y_i = R(x_i))$ and $(x_i, w_i = Q(x_i))$. $R(x) = \sum_0^d y_i \cdot \beta_i(x)$ and $b = R(0)$ and $Q(x) = \sum_0^d w_i \cdot \beta_i(x)$ and $u = Q(0)$. As the truncation does not modify the constant term of the polynomial,

$$P(0) = R(0) \cdot Q(0) = b \cdot u.$$

- At last, by construction $\forall x \in \{x_i\}_{i=0..d}$,

$$y'_i = \sum_{j=0}^{d} \sum_{k=0}^{d} y_j \cdot w_k \cdot \beta_{j,k}(x_i) = P(x_i)$$

□

Intuitivley, the security of the scheme against $k$-th order SCA ($k \leq d$) is based on the following points:

- according to polynomial interpolation, one needs at least $d + 1$ shares to define a polynomial of degree $d$,
- the computation of the $\beta_{j,k}(x_i)$ is independent of any secret,
- the knowledge of $y_j \cdot w_k$ does not leak more information on $b$ (resp. $u$) than the knowledge of $y_j$ (resp. $w_k$),

However the security proof of Algorithm 2 does not seems to be an easy matter and is still an open work.

Finally the affine function $A$ involved in the AES Sbox, as if non linear with respect to the polynomial mask, can nevertheless be implemented using straightforwardly as : $(x'_i, y'_i) \leftarrow (x_i, A(y_i))$. Indeed, if $y_i = P(x_i)$, since $A$ s affine $A(P)$ is a polynomial of degree $d$ with $A(P(0)) = A(b)$ and every $A(y_i)$ is the polynomial value of $A(P)(x)$ in $x_i$.

### 3.2 Complexity of the operations

In order to evaluate the complexity overhead of SSS masking with respect to boolean masking, we compare the complexity of each operation involved in the AES computation for both kind of masking. As shown in the previous section, the multiplication between two shared variables is the most consuming operation, but this is also the case for boolean masking (see [16]). Table 1 resumes the complexities of both schemes.

**Table 1.** Complexity of masked operations

| Operation \ Scheme | Boolean Masking [16] | SSS Masking |
|---|---|---|
| XOR with a constant | 1 XOR | $d+1$ XORs |
| Shared XOR | $d+1$ XORs | $d+1$ XORs |
| Scalar Multiplication | $d+1$ multiplications | $d+1$ multiplications |
| Squaring | $d+1$ squaring | $2(d+1)$ squaring |
| Shared Field Mult. | $2d(d+1)$ XORs<br>$d(d+1)/2$ random numbers<br>$(d+1)^2$ field products | $d(d+1)(d+2)$ XORs<br>$(d+1)^2(2+\frac{d}{2})$ field products |
| Sbox Affine transform. | 1 XOR<br>$d$ ring products | $d$ XORs<br>$d$ ring products |

### 3.3 Masking the full S-box

We have defined secure squaring and multiplication in Section 3.1, we then use the exponentiation algorithm given in [16], and resumed afterward (Algo 3), to implement the power function involved in the AES S-box.

**Algorithm 3** Secure Exponentiation to the power 254 over GF($2^8$)

INPUT: Shared representation of $b$, $(x_i, y_i)_{i=0..d}$

OUTPUT: Shares $(x_i, y'_i)_{i=0..d}$ of the value $b^{254}$

```
for $i = 0$ to $d$ do$(\alpha_i, \zeta_i) \leftarrow (x_i^2, y_i^2)$
$(x_i, \zeta_i)_i \leftarrow$ RefreshMasks$((\alpha_i, \zeta_i)_i, 2)$
$(x_i, \gamma_i)_i \leftarrow$ SecMult$((x_i, \zeta_i), (x_i, y_i))$
for $i = 0$ to $d$ do$(\alpha_i, \delta_i) \leftarrow (x_i^4, \gamma_i^4)$
$(x_i, \delta_i)_i \leftarrow$ RefreshMasks$((\alpha_i, \delta_i)_i, 4)$
$(x_i, \gamma_i)_i \leftarrow$ SecMult$((x_i, \gamma_i), (x_i, \delta_i))$
for $i = 0$ to $d$ do$(\alpha_i, \gamma_i) \leftarrow (x_i^{16}, \gamma_i^{16})$
$(x_i, \gamma_i)_i \leftarrow$ RefreshMasks$((\alpha_i, \gamma_i)_i, 16)$
```

```
$(x_i, \gamma_i)_i \leftarrow$ SecMult$((x_i, \gamma_i), (x_i, \delta_i))$
$(x_i, y'_i)_i \leftarrow$ SecMult$((x_i, \gamma_i), (x_i, \zeta_i))$
return $(x_i, y'_i)_{i=0..d}$
```

Here the RefreshMasks operation is needed to ensure the conservation of the $x_i$'s during the computation and the independence of the coefficients of the polynomials before SecMult operation. Formally it follows Algorithm 4.

**Algorithm 4** RefreshMasks

INPUT: Shared representation of $b$, $(\alpha_i, y_i)_{i=0..d}$, chosen $(x_i)_{i=0..d}$, $t$ such that $\alpha_i = x_i^{2^t}$

OUTPUT: Shared representation of $b$, $(x_i, y'_i)_{i=0..d}$

```
for $i = 0$ to $d$ do
    $\beta'_i \leftarrow \beta_i^{2^t}$
    Share $y_i$ in $(x_j, z_{i_j})$ using Algo 1
for $i = 0$ to $d$ do
    $(x_i, y'_i) \leftarrow \left(x_i, \sum_{j=0}^{d} \beta'_j \cdot z_{j_i}\right)$
return $(x_i, y'_i)_{i=0..d}$
```

Algorithm 4 consists in re-sharing each shares separately using a new random polynomial, then to reconstruct the original shares to obtain $d+1$ shares corresponding to this new polynomial. Eventually the complexity of Algorithm 3 is resumed in Table 2. As a matter of comparison, we recall hereafter the complexity of Boolean masking as given in [16].

**Table 2.** Complexity of inversion algorithms

| order | XORs | multiplications | ˆ$2^j$ | Rand. bytes | RAM (bytes) |
|---|---|---|---|---|---|
| $O1$-SSS | 36 | 54 | 14 | 6 | 20 |
| $O2$-SSS | 150 | 165 | 21 | 18 | 33 |
| $Od$-SSS | $7d^3 + 18d^2 + 11d$ | $5d^3 + 18d^2 + 22d + 9$ | $7(d+1)$ | $3d^2 + 3d$ | $d^2 + 10d + 9$ |
| $O1$-Bool. | 20 | 16 | 6 | 6 | 7 |
| $O2$-Bool. | 56 | 36 | 9 | 16 | 12 |
| $O3$-Bool. | 108 | 64 | 12 | 20 | 18 |
| $O4$-Bool. | 176 | 100 | 15 | 48 | 25 |
| $Od$-Bool. | $7d^2 + 12d$ | $4d^2 + 8d + 4$ | $3(d+1)$ | $2d^2 + 4d$ | $\frac{1}{2}d^2 + \frac{7}{2}d + 3$ |

As a matter of fact, the number of operations involved in SSS masking is larger than that of boolean masking for a given order $d$, as the number of field multiplications and XOR operations are cubic in the order instead of quadratic for boolean masking. We can ask ourselves if this observation remains true for a given security level. This question will be studied in section 4.

### 3.4 Masking the whole AES

In the following, we describe how to mask an AES computation at the $d$th order using SSS masking. We will assume that the secret key has been previously masked and that its $d+1$ shares are provided as input to the algorithm (otherwise a straightforward 1st-order attack would be possible). At the beginning of the computation, the state $s$ (holding the plaintext) is split into $d+1$ shares $(x_0, y_0)$, $(x_1, y_1), \ldots, (x_d, y_d)$ with respect to Shamir's secret sharing scheme. In the next sections, we describe how to perform the different AES transformations on the state shares in order to guarantee the completeness as well as the $d$th-order security.

**Masking AddRoundKey** The AddRoundKey stage at round $r$ consists in XOR-ing the $r$th round key $k_r$ to the state. The masked key schedule provides $d+1$ shares $(x_i, k_{r,i})_i$ for every round key $k_r$. The XOR operation is then processed as described in section 3.1: $M(\mathbf{s} \oplus k_r) \to (x_i, y_i \oplus k_{r,i})_{i=0..d}$

**Masking ShiftRows** As the ShiftRows layer operates on each byte separately and does not change their value, we have: $M(\mathsf{ShiftRows}(\mathbf{s})) = \mathsf{ShiftRows}(M(\mathbf{s}))$

**Masking MixColumn** Since each output byte of $\mathsf{MixColumns}_c$ can be expressed as a linear function of the bytes of the input state over GF(256) we have:
$\mathsf{MixColumns}_c(M(s_0), M(s_1), M(s_2), M(s_3)) = (M(s'_0), M(s'_1), M(s'_2), M(s'_3))$.

This suggests to perform the following steps to securely process $\mathsf{MixColumns}_c$ on the masked representation of the state columns.

$$\begin{cases} M(s'_0) = (x_i, y'_{0,i}) \leftarrow (x_i,\ \mathsf{xtimes}(y_{0,i} \oplus y_{1,i}) \oplus tmp_i \oplus y_{0,i}) \\ M(s'_1) = (x_i, y'_{1,i}) \leftarrow (x_i,\ \mathsf{xtimes}(y_{1,i} \oplus y_{2,i}) \oplus tmp_i \oplus y_{1,i}) \\ M(s'_2) = (x_i, y'_{2,i}) \leftarrow (x_i,\ \mathsf{xtimes}(y_{2,i} \oplus y_{3,i}) \oplus tmp_i \oplus y_{2,i}) \\ M(s'_3) = (x_i, y'_{2,i}) \leftarrow (x_i,\ y'_{0,i} \oplus y'_{1,i} \oplus y'_{2,i} \oplus tmp_i). \end{cases} \tag{3}$$

where $tmp_i = y_{0,i} \oplus y_{1,i} \oplus y_{2,i} \oplus y_{3,i}$ and where xtimes denotes a look-up table for the function $x \mapsto \mathtt{02} \cdot x$. The completeness holds because the single operation that modify the random factors $(a_i)_{i=1..d}$ is the xtimes one, and is applied similarly to each share.

**Masking SubByte** The SubBytes transformation consists in applying the AES S-box S to each byte of the state. In order to mask this transformation, we apply the secure S-box computation described in Section 3.3 to the $(d+1)$-tuple of every byte shares of the state.

**KeySchedule** Finally, since the round key derivation is a composition of the previous transformations, it can be protected using the exact same methods as previously described.

**Overall complexity** In order to give an idea of the global complexity of the scheme, and to compare it to the boolean masking, we give in Table 3 the overall number of operations involved in the ciphering.

**Table 3.** Complexity of cipher implementations

| Masking | XORs/ANDs | Table look-ups | Random bits | RAM (bits) | ROM (bits) |
|---|---|---|---|---|---|
| $1O$ boolean | 17640 | 16144 | 16896 | 312 | 6128 |
| $2O$ boolean | 37800 | 32272 | 46080 | 352 | 6128 |
| $3O$ boolean | 65640 | 54160 | 87552 | 400 | 6128 |
| $1O$ SSS | 31760 | 37296 | 16240 | 400 | 6128 |

# 4 Security analysis

In what follows, we shall consider that an intermediate variable $U_i$ is associated with a leakage variable $L_i$ representing the information leaking about $U_i$ through side channel. We will assume that the leakage can be expressed as a deterministic *leakage function* $\varphi$ of the intermediate variable $U_i$ with an independent additive noise $B_i$. Namely, we will assume that the leakage variable $L_i$ satisfies:

$$L_i = \varphi(U_i) + B_i \ . \tag{4}$$

In the following, we call $d^{th}$*-order leakage* a tuple of $d$ leakage variables $L_i$ corresponding to $d$ different intermediate variables $U_i$ that jointly depend on some sensitive variable. As already argued in Sect. 3.4, when an implementation is correctly protected by SSS masking, no first-order leakage of sensitive information occurs. This directly comes from Shamir's secret sharing scheme security. In the following we will focus on higher orders attacks against protected implementations, secured by boolean or SSS masking.

## 4.1 Information Theoretic Analysis

In order to evaluate the information leaked by $1O$-SSS masking and to compare it to that of various orders Boolean masking, we compute, as suggested in [23], the theoretical mutual information $I(S|L_d)$ for a class discrete variable $S$ of the secret, and a $d$-order leakage $L_d$, with respect to increasing noise standard deviation $\sigma$. Namely we consider the three following leakages:

- $2^{nd}$-order leakage of $1O$-Boolean masking with targeted variables $(Z \oplus m_1, m_1)$
- $3^{rd}$-order leakage of $2O$-Boolean masking with targeted variables $(Z \oplus m_1 \oplus m_2, m_1, m_2)$
- $2^{nd}$-order leakage of $1O$-SSS masking with targeted variables $((x_1, a \cdot x_1 \oplus Z), (x_2, a \cdot x_2 \oplus Z))$

The variables $Z$, $m_1$, $m_2$ and $a$ are assumed uniformly distributed over GF(256) and mutually independent, and $x_1$, $x_2$ are assumed uniformly distributed over $\mathrm{GF}(256)^*$ with $x_1 \neq x_2$. For each kind of leakage, we compute the mutual information between $Z$ and the tuple of leakages in the Hamming weight ($HW$) model with Gaussian noise: the leakage $L_i$ related to a variable $U_i$ is distributed according to equation (4) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ (the different $B_i$'s are also assumed independent). In this context, the signal-to-noise ratio (SNR) of the leakage is defined as $\mathrm{Var}\left[\varphi(U_i)\right]/\mathrm{Var}\left[B_i\right] = 2/\sigma^2$.

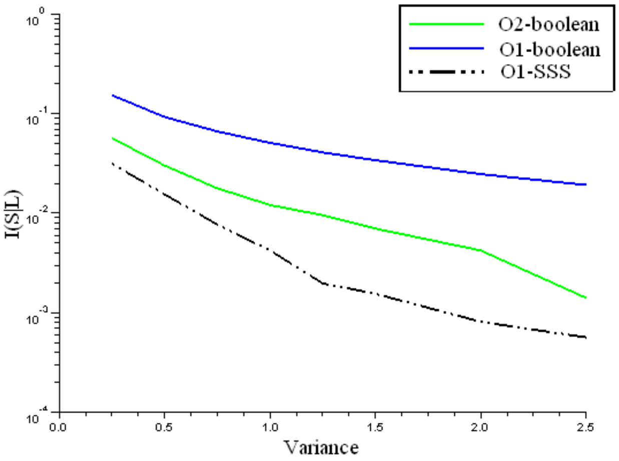

**Fig. 1.** Mutual Information values with respect to $\sigma^2$ (logarithmic scale).

Fig. 1 shows the mutual information values obtained for each kind of leakage with respect to an increasing noise standard deviation. These results demonstrate the information leakage reduction implied by the use of SSS masking. As expected, SSS masking leaks less information than first and second order Boolean masking for the considered Signal to Noise ratios (SNRs). We will now see to which extent this reduction also applies to the efficiency of SCA on SSS masking.

### 4.2 Higher-Order DPA Evaluation

Let us assume that $Z$ depends on the plaintext and of a subkey $k^\star$, and let us denote by $Z(k)$ the hypothetic value of $Z$ for a guess $k$ on $k^\star$. In a *higher-order DPA* (HO-DPA) [13, 18], the attacker tests the guess $k$ by estimating the correlation coefficient $\rho\left[\hat{\varphi}(Z(k)), \mathcal{C}(\mathbf{L})\right]$, where $\mathcal{C}$ is a *combining function* that converts the multivariate leakage $\mathbf{L}$ into a univariate signal and where $\hat{\varphi}$ is a *prediction function* chosen such that $\hat{\varphi}(Z)$ is correlated as much as possible to $\mathcal{C}(\mathbf{L})$. The guess $k$ leading to the greatest correlation (in absolute value) is selected as key-candidate. In [12], the authors show that the number of traces required to mount a successful DPA attack is roughly quadratic in $\rho^{-1}$ where $\rho$ is the correlation coefficient $\rho\left[\hat{\varphi}(Z), \mathcal{C}(\mathbf{L})\right]$ (that is the expected correlation for the correct key guess). The latter can therefore be used as a metric for the efficiency of a (HO-)DPA attack.

The analysis conducted in [18] states that a good choice for $\mathcal{C}$ is the *normalized product combining*:

$$\mathcal{C} : \mathbf{L} \mapsto \prod_i (L_i - \mathrm{E}\left[L_i\right]). \tag{5}$$

Although the effectiveness of the normalized product combining has been only studied in [18] in the context of Boolean masking, we can argue that this combining function stays a natural choice against any kind of masking since $\rho\left[\hat{\varphi}(Z(k)), \mathcal{C}(\mathbf{L})\right]$ is related to the *multivariate correlation*[3] between $\hat{\varphi}(Z(k))$ and every coordinate of $\mathbf{L}$ [24]. Besides, in the presence of (even little) noise in the side-channel leakage, the HO-DPA with normalized product combining is nowadays the most efficient unprofiled attack against Boolean masking in the literature (see for instance [18, 24, 17]).

From Corollary 8 in [18], the optimal correlation $\rho_{\mathrm{SSS}}$ for the correct key hypothesis can be obtained as:

$$\rho_{\mathrm{SSS}} = \sqrt{\frac{\mathrm{Var}\left[\mathrm{E}\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right]\right]}{\mathrm{Var}\left[\overline{L}_1 \times \overline{L}_2\right]}} \tag{6}$$

Formally, when the leakage satisfies (4) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ , the coefficient $\rho_{\mathrm{SSS}}$ obtained for the 2-nd order leakage of 1-st order SSS masking satisfies:

$$\rho_{\mathrm{SSS}} = \sqrt{\frac{n^3 \cdot (2^{n+1} - 4^n - 1)}{\alpha_2 \cdot \sigma^4 + \alpha_1 \cdot \sigma^2 + \alpha_0}}, \tag{7}$$

where $n$ is the bit-size of $Z$, and

$$\begin{aligned}
\alpha_2 &= 192 \cdot 2^n - 2^{4n+4} - 64 - 208 \cdot 4^n + 96 \cdot 8^n \\
\alpha_1 &= (40 \cdot 8^n - 64 \cdot 4^n - 8 \cdot 16^n + 32 \cdot 2^n)n^2 \\
&\quad +(88 \cdot 8^n + 128 \cdot 2^n - 2^{4n+4} - 168 \cdot 4^n - 32)n \\
\alpha_0 &= (8^n - 3 \cdot 4^n + 6 \cdot 2^n - 4)n^4 + (-4 \cdot 16^n + 14 \cdot 8^n - 16 \cdot 4^n + 2 \cdot 2^n + 4)n^3 \\
&\quad +(23 \cdot 8^n - 4 \cdot 16^n - 44 \cdot 4^n + 34 \cdot 2^n - 8)n^2 + (10 \cdot 4^n - 3 \cdot 8^n - 9 \cdot 2^n + 2)n
\end{aligned} \tag{8}$$

*Remark 1.* In order to endorse our choice of targeted variables, we also computed the correlation coefficient corresponding to another 2rd-order leakage of SSS masking targeting the pair $(a, a \cdot x + Z)$ with the corresponding pair of prediction functions: the Dirac function $\delta_0$ ($\delta_0(x) = 0 \Leftrightarrow x \neq 0$) and the Hamming weight.

[3] What we call multivariate correlation here is the straightforward generalization of the correlation coefficient to more than two variables (see [24]).

We observed for several values of $n$ and $\sigma$ that the correlation coefficient was always lower than $\rho_{\mathrm{SSS}}$.

Regarding Boolean masking, it has been shown in [20] that the correlation $\rho_{\mathrm{bool}}$ corresponding to HO-DPA with normalized product combining against $d$th-order Boolean masking satisfies (in the Hamming weight model):

$$\rho_{\mathrm{bool}} = (-1)^d \frac{\sqrt{n}}{(n+4\sigma^2)^{\frac{d+1}{2}}} \ . \tag{9}$$

From (7) and (9), we define the ratio $\nu$ as: $\nu = \frac{\rho_{\mathrm{SSS}}}{\rho_{\mathrm{bool}}}$.

Let us denote by $N_{\mathrm{SSS}}$ (resp. $N_{\mathrm{bool}}$) the number of leakage measurements for a successful attack on SSS masking (resp. Boolean masking). According to [12], $N_{\mathrm{SSS}}$ and $N_{\mathrm{bool}}$ are roughly quadratic in the values of the correlation values. Hence the ratio $\frac{N_{\mathrm{SSS}}}{N_{\mathrm{bool}}}$ satisfies:

$$\frac{N_{\mathrm{SSS}}}{N_{\mathrm{bool}}} \approx \frac{1}{\nu^2} \ . \tag{10}$$

Table 4 illustrate this relation giving the value $1/\nu^2$ for different values of SNRs. Values greater (resp. lower) than 1 indicate that SSS masking is more (resp. less) secure than the considered attack.

**Table 4.** Ratio $1/\nu^2$ for several Boolean masking orders with respect to 1$O$-order SSS masking.

| Attack \ SNR | $+\infty$ | 1 | 1/2 | 1/5 | 1/10 |
|---|---|---|---|---|---|
| 3O-DPA on 2O Boolean Masking | 4544.83 | 899.99 | 374.33 | 94.17 | 22.70 |
| 4O-DPA on 3O Boolean Masking | 568.10 | 56.25 | 15.60 | 1.96 | 0.21 |
| 5O-DPA on 4O Boolean Masking | 71.01 | 3.52 | 0.65 | 0.04 | 0.002 |

Due to (10), SSS masking is more secure than $d$th-order Boolean masking if and only if $|\nu| \leq 1$. Comparing the resistance of the Boolean masking and SSS masking against HO-DPA thus amounts to study when $|\nu| \leq 1$ is satisfied. We can note that $|\nu|$ is an increasing function of $\sigma$ and a decreasing function of $n$. Let us denote by $\vartheta$ the maximal variance of the noise such that $|\nu| \leq 1$ is satisfied. For the first values of $d$, we have:

$$\vartheta = \begin{cases} +\infty & \text{if } d = 1, \\ 282.2683 & \text{if } d = 2, \\ 13.2072 & \text{if } d = 3, \\ 3.4036 & \text{if } d = 4. \end{cases} \tag{11}$$

Eventually Fig. 2 sums up our main theoretical results. In particular, it illustrates the fact that the coefficient $\rho_{\mathrm{SSS}}$ is lower than $\rho_{\mathrm{bool}}$ (computed for $d = 3$) only when the noise variance $\sigma^2$ is lower than 13.2072.

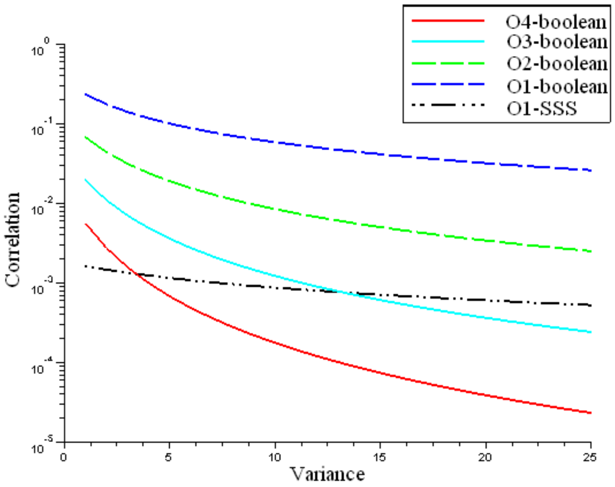

**Fig. 2.** Correlation values with respect to $\sigma^2$ (logarithmic scale).

### 4.3 Attack simulations

In order to confront our theoretical analyses to practical evaluation, we performed several attacks simulations. We then applied several side-channel distinguishers to leakage measurements simulated in the Hamming weight model with Gaussian noise. The leakage measurements have been simulated as samples of the random variables $L_i$ defined according to equation (4) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ (the different $B_i$'s are also assumed independent). For all the attacks, the sensitive variable $Z$ was chosen to be an AES S-box output of the form $\mathsf{S}(M \oplus k^\star)$ where $M$ represents a varying plaintext byte and $k^\star$ represents the key byte to recover.

**Side-channel distinguishers.** We applied two kind of side-channel distinguishers: higher-order DPA such as described in Sect. 4.2 and higher-order MIA. In a HO-MIA [15, 7], the distinguisher is the mutual information: the guess $k$ is tested by estimating $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$. As mutual information is a multivariate operator, this approach does not involve a combining function.

**Targeted variables.** Each attack was applied against the leakages of SSS masking, and Boolean masking. The target variables are those listed in Sect. 4.1 for $Z$ being $\mathsf{S}(X \oplus k^\star)$.

**Prediction functions.** For each DPA, we choose $\hat{\varphi}$ to be the optimal prediction function :

$$\hat{\varphi} : z \mapsto \mathrm{E}\left[\mathcal{C}(\mathbf{L}) | Z = z\right] . \qquad (12)$$

This leads us to select the Hamming weight function in the attacks against both SSS and Boolean masking of any order.

For the MIA attacks, we choose $\hat{\varphi}$ such that it maximizes the mutual information $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$ for $k = k^\star$ while ensuring that the mutual information is lower for $k \neq k^\star$. In our case, every HO-MIA against both SSS and Boolean masking is performed with $\hat{\varphi} = \mathrm{HW}$ since the distribution of $(\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))$ (resp. $(\mathrm{HW}(Z \oplus a_0 \cdot x_0, x_0), \mathrm{HW}(Z \oplus a_0 \cdot x_1, x_1)))$ only depends on $\mathrm{HW}(Z)$. Therefore

$$\mathrm{I}\big(Z; (\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))\big) = \mathrm{I}\big(\mathrm{HW}(Z); (\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))\big).$$

Note that the same relation holds at every masking order.

**Pdf estimation method.** For the (HO-)MIA attacks, we use the histogram estimation method with rule of [8] for the *bin-widths* selection.

**Attack simulation results.** Each attack simulation is performed 100 times for various SNR values ($+\infty$, 1, 1/2, 1/5 and 1/10). Table 5 summarizes the number of leakage measurements required to observe a success rate of 90% in retrieving $k^\star$ for the different attacks.These results shows the security improvement provided by SSS masking with respect to boolean masking. This gain of security can be explained by the fact that an attacker does not have direct access to the mask $a_1 \cdot x_i$, hence the relation between the key and the targeted variables is much more noisy than for boolean masking.

**Table 5.** Number of leakage measurements for a 90% success rate.

| Attack \ SNR | $+\infty$ | 1 | 1/2 | 1/5 | 1/10 |
|---|---|---|---|---|---|
| Attacks against Boolean Masking | | | | | |
| 2O-DPA on 1O Boolean Masking | 150 | 500 | 1500 | 6000 | 20 000 |
| 2O-MIA on 1O Boolean Masking | 100 | 5000 | 15 000 | 50 000 | 160 000 |
| 3O-DPA on 2O Boolean Masking | 1500 | 9000 | 35 000 | 280 000 | $> 10^6$ |
| 3O-MIA on 2O Boolean Masking | 160 | 160 000 | 650 000 | $> 10^6$ | $> 10^6$ |
| Attacks against SSS Masking | | | | | |
| 2O-DPA on 1O SSS Masking | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ |
| 2O-MIA on 1O SSS Masking | 500 000 | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ |
| 3O-DPA on 2O SSS Masking | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ |
| 3O-MIA on 2O SSS Masking | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ | $> 10^6$ |

## 5 Conclusion

In this paper we propose a new alternative to boolean masking to secure implementations of AES against side channel attacks using Shamir's Secret Sharing scheme to share sensitive variables. We give implementation results and conduct a security analysis that clearly show that our scheme can provide a good complexity-security trade-off compared to boolean masking. In particular, on smart card implementation, where SNR value is around 1/2, $1O$ SSS masking provides both a better security and complexity than $3O$ boolean masking. On hardware implementations where the noise can be drastically reduced, $1O$ SSS masking is to be compared to $4^{th}$ order boolean masking, which increase the advantage of SSS masking. These results show that the opening to secret sharing and secure multi-party computation can provide a good alternative to boolean masking. This may be an interesting way to thwart HO-SCA. It is an open research topic to try the security and complexity of such a masking using other kinds of secret sharing scheme.

## References

1. Mehdi-Laurent Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.
3. É. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
4. S. Chari, C.S. Jutla, J.R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Wiener [26], pages 398–412.
5. Christophe Clavier and Kris Gaj, editors. *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*. Springer, 2009.
6. Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*. Springer, 2010.
7. Benedikt Gierlichs, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. Cryptology ePrint Archive, Report 2009/228, 2009. http://eprint.iacr.org/.
8. Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
9. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
10. Yuval Ishai, Amit Sahai, and David 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.
11. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Wiener [26], pages 388–397.
12. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
13. T.S. Messerges. Using Second-order Power Analysis to Attack DPA Resistant Software. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded*

*Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

14. D. Pointcheval, editor. *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*. Springer, 2006.
15. Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security – ANCS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518. Springer, 2009.
16. Emmanuel Prouff and Matthieu Rivain. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and Franois-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
17. Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis (Extended Version). To appear in the Int. Journal of Applied Cryptography (IJACT), 2010.
18. Emmanuel Prouff, Matthieu Rivain, and Régis Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.*, 58(6):799–811, 2009.
19. Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In *Cryptographic Hardware and Embedded Systems – CHES 2011*, Lecture Notes in Computer Science. Springer.
20. Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In Clavier and Gaj [5], pages 171–188.
21. Kai Schramm and Christof Paar. Higher Order Masking of the AES. In Pointcheval [14], pages 208–225.
22. Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.
23. François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
24. Francois-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order dpa. Cryptology ePrint Archive, Report 2010/180, 2010. http://eprint.iacr.org/.
25. Manfred von Willich. A technique with an information-theoretic basis for protecting secret data from differential power attacks. In *IMA int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 44–62. Springer, 2001.
26. M.J. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.