Affine Masking against Higher-Order Side Channel Analysis*

Guillaume Fumaroli[1], Ange Martinelli[1], Emmanuel Prouff[2], and Matthieu Rivain[3]

[1] Thales Communications
{guillaume.fumaroli, jean.martinelli}@fr.thalesgroup.com
[2] Oberthur Technologies
e.prouff@oberthur.com
[3] CryptoExperts
matthieu.rivain@cryptoexperts.com

Abstract. In the last decade, an effort has been made by the research community to find efficient ways to thwart side channel analysis (SCA) against physical implementations of cryptographic algorithms. A common countermeasure for implementations of block ciphers is Boolean masking which randomizes by the bitwise addition of one or several random value(s) to the variables to be protected. However, advanced techniques called *higher-order SCA* attacks exist that overcome such a countermeasure. These attacks are greatly favored by the very nature of Boolean masking. In this paper, we revisit the *affine masking* initially introduced by Von Willich in 2001 as an alternative to Boolean masking. We show how to apply it to AES at the cost of a small timing overhead compared to Boolean masking. We then conduct an in-depth analysis pinpointing the leakage reduction implied by affine masking. Our results clearly show that the proposed scheme provides an excellent performance-security trade-off to protect AES against higher-order SCA.

1 Introduction

Side Channel Analysis is a cryptanalytic technique that consists in analyzing the *side channel leakage* (*e.g.* the power consumption, the electromagnetic emanations) 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 processed. Some of these variables are *sensitive* in the sense that they are related to secret data, and recovering information on them therefore enables efficient key recovery attacks [12, 2, 8].

A very common countermeasure to protect implementations of block ciphers against SCA is to randomize the sensitive variables by masking techniques [3, 10]. The principle is to add one or several random value(s) (called *mask(s)*) to every sensitive variable occurring during the computation. Masks and masked

* Full version of the paper published in the proceedings of SAC 2010.

variables propagate throughout the cipher in such a way that every intermediate variable is independent of any sensitive variable. This strategy ensures that the instantaneous leakage is independent of any sensitive variable, thus rendering SCA difficult to perform. The masking can be characterized by the number of random masks used per sensitive variable. A masking that involves d random masks is called a *d^{th}-order masking*. Such a masking 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 [16, 27, 22]. However, the noise effects imply that the difficulty of carrying out a d^{th}-order SCA in practice increases exponentially with d [3]. The d^{th}-order SCA resistance (for a given d) is thus a good security criterion for implementations of block ciphers. Unfortunately, only a few higher-order masking schemes exist and they are costly in timings [27, 5, 23, 24].

Instead of looking for perfect security against d^{th}-order SCA, 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. Merely all masking schemes proposed in the literature are based on *Boolean masking* where masks are introduced by exclusive-or (XOR). This masking enables securing implementations against first-order SCA quite efficiently, however it is especially vulnerable to higher-order SCA [16, 18] due to the intrinsic physical properties of electronic devices. Indeed, when the mask is introduced by XOR, then every bit of the mask randomizes one bit of the sensitive variable. Since each bit of a processed variable usually independently contributes to the leakage, the information leaking about the mask and the information leaking about the masked variable can be efficiently combined to unmask the variable.

A first work towards the direction of practical – instead of perfect – security against d^{th}-order SCA has been published by von Willich [30]. It argues that affine masking offers an improved SCA resistance compared to standard first-order masking schemes. However, implementation issues are not taken into account and the paper does not explain how to apply affine masking to usual block ciphers such as AES or DES. Moreover, von Willich defines the affine masking over the vector space $\text{GF}(2)^n$. When defined in such a way, it implies the generation of an invertible $n \times n$ binary matrix, and n scalar products over GF(2) each time a sensitive variable must be masked. Those steps, and especially the scalar products, are very costly when applied in software. A natural idea to deal with this issue is to define the operations over the field $\text{GF}(2^n)$ in place of the vector space $\text{GF}(2)^n$. The addition operation stays unchanged and the field multiplication is a particular case of the matrix product (the security analysis conducted in this paper shows that both operations offer similar SCA resistance). The idea of masking sensitive data with a multiplicative mask in a field structure was first proposed in [1] to protect an AES implementation. However it was shown in [9] that such a masking is insecure since, by nature, it fails in masking the zero value. A similar zero-value first-order flaw was subsequently exploited in [6] to break the linear masking proposed to protect DES in

[10]. These works clearly show that letting the zero value unmasked renders a masking scheme insecure.

Our contribution. In this paper, we propose a practical application of affine masking to AES. 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 $G(z) = r_1 \cdot z \oplus r_0$, where $(r_1, r_0) \in \mathrm{GF}(2^n)^* \times \mathrm{GF}(2^n)$ is a pair of random values generated before each new execution of the algorithm. Our scheme is very efficient as it maintains the same compatibility as Boolean masking (which is a particular case of our scheme for $r_1 = 1$) with the linear transformations of the algorithm. In the second part of the paper, we conduct an in-depth analysis which shows that the joint use of a multiplicative mask with a Boolean mask greatly improves the resistance of the scheme to higher-order SCA. As we argue, the multiplicative mask enables to complicate the relationship between the unmasked data and the leakage while the Boolean mask prevents the zero-value leakage. Our analysis pinpoints the leakage reduction resulting from affine masking as well as its improved higher-order SCA resistance.

2 Securing AES with Affine Masking

The AES is an iterated block cipher algorithm. It is composed of several rounds that operate on a 4×4 array of bytes denoted by $\mathbf{s} = (s_{i,j})_{0 \leq i,j \leq 3}$ and termed the *state*. At the beginning of AES in encryption mode, the state is initialized with the plaintext.

Let us first briefly introduce the outlines of our method. Initially, both the state $\mathbf{s}$ and the master key $\mathbf{mk} = (mk_{i,j})_{0 \leq i,j \leq 3}$ are masked by applying a randomly generated affine transformation G to each $s_{i,j}$ and $mk_{i,j}$. Then, all the original transformations of the cipher are adapted in order to process on and to return affinely masked variables. Should the additive part of the mask cancel out within the computation, a temporary additive mask is introduced in order to avoid any potential zero-value first-order flaw. Eventually, the ciphertext $(c_{i,j})_{0 \leq i,j \leq 3}$ is simply recovered by applying the inverse mapping G^{-1} to each part of the final value of the masked state (which contains the values $G(c_{i,j})$).

Before explaining how the AES implementation can be adapted to securely operate on a state masked by an affine transformation G, we briefly recall the algorithm specifications in the following section.

2.1 The AES Encryption Algorithm

We recall in this section the four main operations involved in the AES encryption algorithm. For each of them, we denote by $\mathbf{s} = (s_{i,j})_{0 \leq i,j \leq 3}$ the state at the input of the transformation, and by $\mathbf{s}' = (s'_{i,j})_{0 \leq i,j \leq 3}$ the state at the output of the transformation.

1. AddRoundKey: Let $\mathbf{k} = (k_{i,j})_{0 \le i,j \le 3}$ denote the round key. Each byte of the state is XOR-ed with the corresponding round key byte:
$$(s'_{i,j}) \leftarrow (s_{i,j}) \oplus (k_{i,j}).$$
2. SubBytes: each byte of the state passes through the 8-bit AES S-box S:
$$s'_{i,j} \leftarrow \mathsf{S}(s_{i,j}) \ .$$
3. ShiftRows: each row of the state is cyclically shifted by a certain offset:
$$s'_{i,j} \leftarrow s_{i,j-i \bmod 4} \ .$$
4. MixColumns: each column of the state is modified as follows:
$$(s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}) \leftarrow \mathsf{MixColumns}(s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}) \ ,$$
where MixColumns implements the following operations:
$$\begin{cases} s'_{0,c} \leftarrow (\mathsf{02} \cdot s_{0,c}) \oplus (\mathsf{03} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} \leftarrow s_{0,c} \oplus (\mathsf{02} \cdot s_{1,c}) \oplus (\mathsf{03} \cdot s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} \leftarrow s_{0,c} \oplus s_{1,c} \oplus (\mathsf{02} \cdot s_{2,c}) \oplus (\mathsf{03} \cdot s_{3,c}) \\ s'_{3,c} \leftarrow (\mathsf{03} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\mathsf{02} \cdot s_{3,c}) \end{cases}$$
where $\cdot$ denotes the multiplication and the addition in the field $\mathrm{GF}(2)[X]/(X^8 + X^4 + X^3 + X + 1)$, and where 02 and 03 respectively denote the elements X and $X+1$. In the following, we will assume that MixColumns is implemented as
$$\begin{cases} s'_{0,c} \leftarrow \mathsf{xtimes}(s_{0,c} \oplus s_{1,c}) \oplus tmp \oplus s_{0,c} \\ s'_{1,c} \leftarrow \mathsf{xtimes}(s_{1,c} \oplus s_{2,c}) \oplus tmp \oplus s_{1,c} \\ s'_{2,c} \leftarrow \mathsf{xtimes}(s_{2,c} \oplus s_{3,c}) \oplus tmp \oplus s_{2,c} \\ s'_{3,c} \leftarrow s'_{0,c} \oplus s'_{1,c} \oplus s'_{2,c} \oplus tmp \end{cases}$$
where $tmp = s_{0,c} \oplus s_{1,c} \oplus s_{2,c} \oplus s_{3,c}$ and where xtimes denotes a look-up table for the function $x \mapsto \mathsf{02} \cdot x$.

2.2 Affine Masking applied to AES

To secure the state manipulations thanks to the affine masking countermeasure every manipulation of $\mathbf{s}$ is replaced by a manipulation of $\mathbf{G}(\mathbf{s}) = (G(s_{i,j}))_{0 \le i,j \le 3}$. In the following, we assume that G is defined with respect to a pair (r_1, r_0) of random elements of $\mathrm{GF}(256)^* \times \mathrm{GF}(256)$ as:
$$G : x \in \mathrm{GF}(256) \longmapsto r_1 \cdot x \oplus r_0 \in \mathrm{GF}(256) \ .$$
In the sequel, $G(x)$ shall be called the *G-representation* of x and the variables r_1 and r_0 shall be referred to as the *multiplicative mask* and and the *additive mask* respectively.

Let us now explain how the four main AES primitives can be easily adapted to securely operate on a state masked by an affine transformation G. We shall denote by $\mathbf{G}(\mathbf{s}) = (G(s_{i,j}))_{0 \le i,j \le 3}$ the masked state at the input of each transformation, by $\mathbf{G}(\mathbf{s}') = (G(s'_{i,j}))_{0 \le i,j \le 3}$ the masked state at the output, and by $\mathbf{G}(\mathbf{k}) = (G(k_{i,j}))_{0 \le i,j \le 3}$ the masked representative of the current round key.

1. To securely compute the G-representation of the output of $\mathsf{AddRoundKey}$ from the G-representations of the input state and the round key, each byte $G(s)$ of the state is XOR-ed with the corresponding round key byte $G(k)$ as follows:
$$G(s') \leftarrow (((G(s) \oplus r) \oplus G(k)) \oplus r_0) \oplus r \ .$$
where r is randomly chosen in GF(256). The method is essentially based on the following observation: each masked output byte $G(s'_{i,j})$ can be computed as $G(s'_{i,j}) = G(s_{i,j} \oplus k_{i,j}) = G(s_{i,j}) \oplus G(k_{i,j}) \oplus r_0$. A temporary random mask has to be introduced to ensure that the state bytes are always masked affinely and not only linearly.
2. To process the S-box transformations, we propose to use a new look-up table $\tilde{\mathsf{S}}$ that is recomputed at each new AES execution from both G and S such that for every $x \in \mathrm{GF}(256)$, we have:
$$\tilde{\mathsf{S}}[G(x)] = G(\mathsf{S}[x]) \ . \tag{1}$$
Remark 1. In Sect. 2.3, we propose several methods to generate the new S-box S with various time-memory trade-offs.

It can be easily checked that processing $\tilde{\mathsf{S}}$ on the G-representation of a byte $s_{i,j}$ results in the G-representation of $s'_{i,j} = \mathsf{S}[s_{i,j}]$. Securing the $\mathsf{SubBytes}$ transformation with the affine masking thus simply consists in applying $\tilde{\mathsf{S}}$ to each byte of the state:
$$G(s'_{i,j}) \leftarrow \tilde{\mathsf{S}}[G(s_{i,j})] \ .$$
3. Since we have $\mathsf{ShiftRows}(\mathbf{G}(\mathbf{s})) = \mathbf{G}(\mathsf{ShiftRows}(\mathbf{s}))$ and since $\mathsf{ShiftRows}$ operates on each byte separately, it can be directly applied on $\mathbf{G}(\mathbf{s})$ without introducing any flaw:
$$\mathbf{G}(\mathbf{s}') \leftarrow \mathsf{ShiftRows}(\mathbf{G}(\mathbf{s})) \ .$$
4. Since each output byte of $\mathsf{MixColumns}$ can be expressed as a linear function of the bytes of the input state over GF(256), it can be checked that we have:
$$\begin{aligned}&\mathsf{MixColumns}(G(s_{0,c}), G(s_{1,c}), G(s_{2,c}), G(s_{3,c})) \\ &\qquad = (G(s'_{0,c}), G(s'_{1,c}), G(s'_{2,c}), G(s'_{3,c})).\end{aligned}$$
This suggests to perform the following steps to securely process $\mathsf{MixColumns}$ on the G-representation of the state columns.
$$\begin{cases} tmp \leftarrow r \oplus G(s_{0,c}) \oplus G(s_{1,c}) \oplus G(s_{2,c}) \oplus G(s_{3,c}) \\ G(s'_{0,c}) \leftarrow \mathsf{xtimes}(G(s_{0,c}) \oplus r' \oplus G(s_{1,c})) \oplus tmp \oplus G(s_{0,c}) \oplus r \oplus \mathsf{xtimes}(r') \\ G(s'_{1,c}) \leftarrow \mathsf{xtimes}(G(s_{1,c}) \oplus r' \oplus G(s_{2,c})) \oplus tmp \oplus G(s_{1,c}) \oplus r \oplus \mathsf{xtimes}(r') \\ G(s'_{2,c}) \leftarrow \mathsf{xtimes}(G(s_{2,c}) \oplus r' \oplus G(s_{3,c})) \oplus tmp \oplus G(s_{2,c}) \oplus r \oplus \mathsf{xtimes}(r') \\ G(s'_{3,c}) \leftarrow r \oplus G(s'_{0,c}) \oplus G(s'_{1,c}) \oplus G(s'_{2,c}) \oplus tmp \end{cases}$$
To ensure that the state bytes are always masked affinely and not only linearly, two temporary random masks $r, r' \in \mathrm{GF}(256)$ have to be introduced. Moreover, the operations above must be processed from left to right.

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

2.3 Time-Memory Trade-Offs

Affine masking requires 32 computations of G in order to mask both the plaintext and the key, and eventually 16 computations of G^{-1} in order to unmask the ciphertext. Field multiplications and inversions involved in affine masking can be efficiently implemented with the well-known log/alog tables technique as long as conditional statements are avoided to thwart timing attacks (see Appendix A for an example of such an implementation).

Essentially, the processing of $G(s_{i,j})$, $G^{-1}(s_{i,j})$ and $\tilde{S}(s_{i,j})$ may be conducted on-the-fly or may involve pre-computations. Both strategies have different impacts on time and storage costs.

The best time-memory trade-off consists in using two look-up tables for G and $\tilde{\mathsf{S}}$, and in processing one field multiplication and one addition each time G^{-1} must be performed on a state element. The different steps of the look-up table generations of G and $\tilde{S}$ are summarized in Algorithm 1.

Algorithm 1
INPUT: $r_0 \in \mathrm{GF}(256)$, $r_1 \in \mathrm{GF}(256)^*$, and the LUT S for the AES S-box
OUTPUT: The LUTs for G and $\tilde{\mathsf{S}}$

1. **for** $i = 0$ **to** 255 **do**
2. $\quad G[i] \leftarrow r_1 \cdot i \oplus r_0$
3. **for** $i = 0$ **to** 255 **do**
4. $\quad \tilde{\mathsf{S}}[G[i]] \leftarrow G[\mathsf{S}[i]]$
5. **return** $(G, \tilde{\mathsf{S}})$

As G^{-1} is not stored as a look-up table, each byte $\tilde{s}$ of the final output state has to be unmasked by processing $s \leftarrow r_1^{-1} \cdot (\tilde{s} \oplus r_0)$.

This way of implementing the affine masking requires the storage of 512 bytes for the look-up tables G and $\tilde{\mathsf{S}}$. It also involves 256 multiplications in the field GF(256) and 256 XORs to generate G, while $\tilde{\mathsf{S}}$ is generated using look-ups only. The initial masking of the plaintext and the key only requires 32 table look-ups. Unmasking implies a total of 16 inversions and 16 multiplications in the field GF(256).

As an alternative to the previous algorithm, two variants can be proposed.

1. **First variant.** $\tilde{\mathsf{S}}$, G and G^{-1} are pre-computed using three look-up tables in order to save on-the-fly computations. Masking both the plaintext and the key involves 32 table look-ups and unmasking the ciphertext involves 16 table look-ups. This method requires the storage of 3×256 bytes for these look-up tables. It also involves 256 multiplications in the field GF(256) to generate G.

2. **Second variant.** This variant involves a single look-up table for $\tilde{\mathsf{S}}$ and performs every other operation on-the-fly. It requires the storage of 256 bytes for this look-up table. It also involves 2×256 multiplications in the field GF(256) to generate $\tilde{\mathsf{S}}$, and 32 multiplications for the initial masking of the plaintext and the key. Unmasking implies a total of 16 inversions and 16 multiplications in the field GF(256).

2.4 Implementation Results

In this section, we compare several AES implementations protected by affine masking, first-order Boolean masking and second-order Boolean masking. The codes are written in assembly language for an 8051-based 8-bit architecture. More details about these countermeasures can be found in the respective papers [15, 23, 27]. Table 1 lists the timing and memory performances of each implementation.

Table 1. Comparison of AES implementations.

Method	Reference	Cycles	RAM (bytes)	ROM (bytes)
Unprotected Implementation				
No Masking	Na.	2×10^3	32	1150
Provably Secure First-Order SCA Resistant Implementation				
First-Order Boolean Masking	[15]	9×10^3	$256 + 35$	1744
Affine Masking (ref. implem.)	This paper	29×10^3	$512 + 37$	2857
Affine Masking (1^{st} var.)	This paper	28×10^3	$768 + 36$	2985
Affine Masking (2^{nd} var.)	This paper	38×10^3	$256 + 37$	3252
Provably Secure Second-Order SCA Resistant Implementation				
Second-Order Boolean Masking	[27]	594×10^3	$512 + 90$	2336
Second-Order Boolean Masking	[23]	672×10^3	$256 + 86$	2215

Table 1 shows that the implementation of AES protected by affine masking is 3.2 to 4.2 times slower than the one protected by first-order Boolean masking, whereas the memory overhead is either +0% (2^{nd} variant) or +100% (reference implementation) or +200% (3rd variant). When compared to the second-order Boolean masking proposed in [27] and [23], the affine masking of AES is 17.7 times faster with the third variant and 20.5 times faster with the first variant.

As every intermediate variable of the computation is affinely masked, we keep a perfect security with respect to first-order SCA. Moreover, as argued in the next section, we significantly increase the resistance of the implementation against higher-order SCA. In view of the implementation performances depicted in Table 1, this rise in security has been obtained at the cost of a very small overhead when compared to the overhead of provably secure second-order Boolean masking. This demonstrates that affine masking is a sound alternative

to second-order Boolean masking to increase the security of an implementation against higher-order SCA.

3 Resistance to Higher-Order SCA

Affine masking is not inherently perfectly secure against higher-order SCA. It can for instance be checked that several pairs of intermediate variables of the scheme proposed in Sect. 2 depend on sensitive variables. We however argue in this section that affine masking is much more resistant than the widely-used Boolean masking. To highlight this statement, we quantify the information leakage reduction provided by affine masking and we study the efficiency of higher-order DPA [16, 22] against it. For comparison purposes, we apply the same analysis to Boolean masking. We eventually give the results of several attack experiments in order to check the reliability of our theoretical analysis with respect to practical attack scenarios.

3.1 Leakage of Affine Masking

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* φ of the intermediate variable U_i with an independent additive noise B_i. Namely, we will assume that the leakage variable L_i satisfies:

$$L_i = \varphi(U_i) + B_i \ . \tag{2}$$

In the following, we shall call d^{th}*-order leakage* a tuple of d leakage variables L_i corresponding to d different intermediate variables U_i that jointly dependent on some sensitive variable. As already argued in Sect. 2, when an implementation is correctly protected by affine masking (*i.e.* when every sensitive variable is affinely masked), no first-order leakage of sensitive information occurs. This is a consequence of the action of the random additive mask R_0. However, as detailed hereafter, second-order and third-order information leakages do occur in the presence of affine masking.

Second-order leakage. To recover sensitive information when affine masking is applied, one must at least consider the joint leakage of two different intermediate variables U_1 and U_2 that share common masks. Those variables can thus be assumed to satisfy:

$$\begin{cases} U_1 = G(Z_1) = R_1 \cdot Z_1 \oplus R_0 \\ U_2 = G(Z_2) = R_1 \cdot Z_2 \oplus R_0 \end{cases} , \tag{3}$$

where R_1 and R_0 are random variables defined over $\mathrm{GF}(2^n)^*$ and over $\mathrm{GF}(2^n)$ respectively and where Z_1 and Z_2 are sensitive variables. A particular case is $Z_2 = 0$ which amounts to target the pair $(G(Z_1), R_0)$.

In the following, we shall assume that R_1 and R_0 are uniformly distributed over $\mathrm{GF}(2^n)^*$ and over $\mathrm{GF}(2^n)$ respectively and that they are mutually independent of the pair (Z_1, Z_2), and of each other. After denoting by Z the sensitive variable $Z_1 \oplus Z_2$, we obtain the following lemma.

Lemma 1. *The pairs (U_1, U_2) and $(G(Z), R_0)$ are identically distributed.*

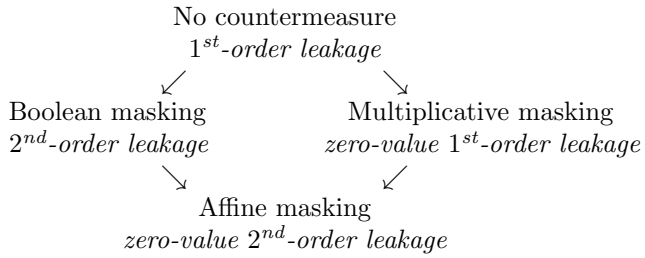
Proof. Let $R'_0 = R_1 \cdot Z_2 \oplus R_0$. We have $U_1 = R_1 \cdot Z \oplus R'_0$ and $U_2 = R'_0$. Moreover, R'_0 is uniformly distributed and mutually independent of both Z and R_1, which concludes the proof. □

Lemma 1 shows that the second-order leakage corresponding to a pair of sensitive variables (Z_1, Z_2) both affinely masked is equivalent to the the second-order leakage on a sensitive variable $Z = Z_1 \oplus Z_2$ that is affinely masked and on the corresponding additive mask R_0. For this reason, in the following, we shall only consider a second-order leakage corresponding to a pair $(G(Z), R_0)$, with Z being possibly the sum of two sensitive variables. The analysis hereafter shall further make use of the following lemma.

Lemma 2. *The random pair $\big((L_1, L_2)|Z = z\big)$ is identically distributed for every $z \in \mathrm{GF}(2^n)^*$ and the random pair $\big((L_1, L_2)|Z = 0\big)$ has a distinct distribution.*

Proof. Since Z equals $Z_1 \oplus Z_2$, we have $\big((L_1, L_2)|Z = z\big) = \big(\varphi(R'_1 \oplus R'_0) + B_1, \varphi(R'_0) + B_2\big)$, where $R'_1 = R_1 \cdot z$ and $R'_0 = R_1 \cdot Z_2 \oplus R_0$. For every $z \in \mathrm{GF}(2^n)^*$, R'_1 and R'_0 are uniformly distributed over $\mathrm{GF}(2^n)^*$ and $\mathrm{GF}(2^n)$ respectively, and they are mutually independent. Therefore the distribution of $\big((L_1, L_2)|Z = z\big)$ is the same for every $z \in \mathrm{GF}(2^n)^*$. On the other hand, $Z = 0$ implies $Z_1 = Z_2$ and therefore $\big((L_1, L_2)|Z = 0\big) = \big(\varphi(R'_0) + B_1, \varphi(R'_0) + B_2\big)$. Since φ is not constant by definition, the distribution of $(\varphi(R'_0), \varphi(R'_0))$ differs from the distribution of $(\varphi(R'_1 \oplus R'_0), \varphi(R'_0))$ and it follows that the distribution of $((L_1, L_2)|Z = 0)$ differs from the distribution of $((L_1, L_2)|Z \in \mathrm{GF}(2^n)^*)$. □

Lemma 2 shows that the second-order leakage (L_1, L_2) only reveals information about whether Z equals 0 or not (*i.e.* whether Z_1 equals Z_2 or not). Such a leakage can be thought as a *zero-value second-order leakage* analogously to the *zero-value first-order leakage* of multiplicative masking [1, 9]. Intuitively, we have the following diagram where each arrow indicates an additional security level.



Third-order leakage. To get more information about Z, a natural idea is to exploit (L_1, L_2) together with the leakage L_3 on the multiplicative mask $U_3 = R_1$. Indeed, while the pair (U_1, U_2) only reveals whether Z equals 0 or not, the triplet (U_1, U_2, U_3) does reveal the full value of Z by:

$$Z = U_3^{-1} \cdot (U_1 \oplus U_2) = R_1^{-1} \cdot (G(Z_1) \oplus G(Z_2)). \tag{4}$$

However, the information about the U_i's that leak through the L_i's does not enable a simple recovery as in (4). In fact, we expect that extracting information on Z through side channels is more difficult when affine masking is applied in place of Boolean masking. Indeed, when the mask is introduced by bitwise addition, then each bit of the mask acts on a single bit of the sensitive variable. In this case, every bit of the masked variable depends on a single bit of the mask. Since bits of processed variables usually contribute to the leakage independently, the information leaking about the mask and the information leaking about the masked variable can be efficiently combined to unmask the variable. This can be illustrated in the Hamming weight leakage model (where $\varphi = \mathrm{HW}$) by the important correlation between $\mathrm{HW}(Z)$ and either $|\mathrm{HW}(Z \oplus R_0) - \mathrm{HW}(R_0)|$ or $(\mathrm{HW}(Z \oplus R_0) - n/2)(\mathrm{HW}(R_0) - n/2)$ [16, 22]. When a further mask is introduced by multiplication in $\mathrm{GF}(2^n)^*$, the additive mask still prevents from a zero-value first-order leakage (as for multiplicative masking [9]), and the new multiplicative mask ensures that every bit of the masked variable depends on every bit of both the sensitive variable and the multiplicative mask. In this case, it is legitimate to expect that the information leaked by side channel is much more difficult to exploit to recover information about Z. For instance, there is no evident way to combine $\mathrm{HW}(Z \cdot R_1)$ and $\mathrm{HW}(R_1)$ to construct a variable with high correlation with $\mathrm{HW}(Z)$. In order to validate this intuition, we conduct in the next section an information theoretic evaluation of the leakages resulting from affine masking and different kinds of masking.

3.2 Information Theoretic Evaluation

In order to evaluate the information revealed by affine masking leakages (first-order and second-order) we follow the information theoretic approach suggested in [28]. Namely we compute the mutual information between the sensitive variable Z and either the pair of leakages (L_1, L_2) or the triplet of leakages (L_1, L_2, L_3). For comparison purposes, we proceed similarly for Boolean masking and multiplicative masking. We list hereafter the leakages we consider and the underlying leaking variables:

- 2^{nd}-order leakage of 1^{st}-order Boolean masking: $(Z \oplus R_0, R_0)$
- 3rd-order leakage of 2^{nd}-order Boolean masking: $(Z \oplus R_0 \oplus R_0', R_0, R_0')$
- 1^{st}-order leakage of multiplicative masking: $R_1 \cdot Z$
- 2^{nd}-order leakage of multiplicative masking: $(R_1 \cdot Z, R_1)$
- 2^{nd}-order leakage of affine masking: $(R_1 \cdot Z \oplus R_0, R_0)$
- 3^{rd}-order leakage of affine masking: $(R_1 \cdot Z \oplus R_0, R_0, R_1)$

The variables Z, R_0, R'_0 and R_1 are assumed to be uniformly distributed (over GF(256) for the former and over GF(256)* for R_1) and mutually independent. For each kind of leakage, we computed the mutual information between Z and the tuple of leakages in the Hamming weight model with Gaussian noise: the leakage L_i related to a variable U_i is distributed according to (2) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ (the different B_i's are also assumed to be mutually independent). In this context, the signal-to-noise *ratio* (SNR) of the leakage is defined as $\mathrm{Var}\left[\varphi(U_i)\right] / \mathrm{Var}\left[B_i\right] = 2/\sigma^2$.

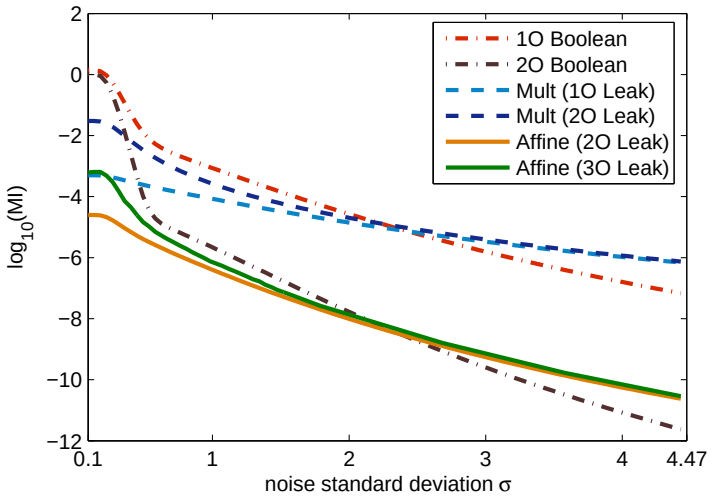


Fig. 1. Mutual information ($\log_{10}$) between the leakage and the sensitive variable over an increasing noise standard deviation.

Fig. 1 shows the mutual information values obtained for each kind of leakage with respect to an increasing noise standard deviation over $[0.1, 4.47]$ (*i.e.* a decreasing SNR over $[\frac{1}{10}, 200]$). These results demonstrate the information leakage reduction implied by the use of affine masking. As expected, affine masking leaks less information than multiplicative masking and first-order Boolean masking for all SNRs. We further observe that affine masking leaks less information than second-order Boolean masking when σ is lower than 2.36, that is when the SNR is greater than 0.36. This first analysis allows us to conclude that affine masking is less leaky than 2^{nd}-order Boolean masking when the amount of noise in the leakage is small. On the other hand, these results illustrate that a 2^{nd}-order SCA security is asymptotically better than a 1^{st}-order SCA security even if the masking relation is more complicated in the latter case. Similarly, we see that for a low noise amount, multiplicative masking is more resistant than 1^{st}-

order Boolean masking although it does not thwart 1st-order SCA while Boolean masking does.

Fig. 1 also confirms our intuition regarding the information provided by the leakage on the multiplicative mask. Observing the obtained mutual information for affine masking and for multiplicative masking, we note that the information gained from the leakage on the multiplicative mask is low. This phenomenon amplifies when σ increases and, beyond $\sigma \approx 2$ the distance between the mutual information curves almost vanishes for both kinds of masking. This means that when the noise is sufficiently strong, the leakage on the multiplicative mask does not provide useful information anymore and only the zero-value leakage reveals sensitive information.

In this section, we have quantified the impact of affine masking on the reduction of the information leakage. We will now see to which extent this reduction also applies to the efficiency of side channel attacks on affine masking.

3.3 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) [16, 22], 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 as much as possible correlated to $\mathcal{C}(\mathbf{L})$. The guess k leading to the greatest correlation in absolute value is selected as key-candidate. In [14], the authors show that the number of traces required to mount a successful DPA attack is roughly quadratic in ρ^{-1} where ρ 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 [22] 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 [22] in the context of Boolean masking, 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*[4] between $\hat{\varphi}(Z(k))$ and every coordinate of $\mathbf{L}$ [29]. 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 [22, 29, 24]). For those reasons, it is natural to study how efficient is a HO-DPA

[4] What we call multivariate correlation here is the straightforward generalization of the correlation coefficient to more than two variables (see [29]).

with normalized product combining against affine masking compared to Boolean masking.

In [22], it is also shown that the best choice for $\hat{\varphi}$ given $\mathcal{C}$ is:

$$\hat{\varphi} : z \mapsto \mathrm{E}\left[\mathcal{C}(\mathbf{L})|Z = z\right]. \tag{6}$$

As explained in [22], the attacker may not be able to evaluate $\hat{\varphi}$ without knowing the exact distribution of $\mathbf{L}$ given Z (as in a profiled attack scenario). In a security evaluation context, it however makes sense to assume that the attacker has this ability. As proved in Appendix B, the optimal prediction function $\hat{\varphi}$ computed according to (6) for the zero-value 2^{nd}-order leakage of affine masking is an affine transformation of the dirac function δ_0 defined as[5]:

$$\delta_0(z) = \begin{cases} 1 & \text{if } z = 0\ , \\ 0 & \text{if } z \neq 0\ . \end{cases} \tag{7}$$

Therefore, we have $\rho\left[\hat{\varphi}(Z(k)), \mathcal{C}(\mathbf{L})\right] = \pm\rho\left[\delta_0(Z(k)), \mathcal{C}(\mathbf{L})\right]$, that is, the attack performs similarly with $\hat{\varphi}$ and δ_0.

Remark 2. Computing $\rho\left[\delta_0(Z(k)), \mathcal{C}(\mathbf{L})\right]$ amounts to performing a zero-value DPA attack as in [9] but on the combined leakage $\mathcal{C}(\mathbf{L})$. After assuming that $Z(k)$ is uniformly distributed over $\mathrm{GF}(2^n)$, it can indeed be checked that the covariance between $\delta_0(Z(k))$ and $\mathcal{C}(\mathbf{L})$ (which is the discriminating element in the correlation) equals $\frac{2^n-1}{2^n}\mathrm{E}\left[\mathcal{C}(\mathbf{L})|Z(k) \neq 0\right] - \frac{1}{2^n}\mathrm{E}\left[\mathcal{C}(\mathbf{L})|Z(k) = 0\right]$.

When the leakage satisfies (2) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ (*i.e.* when the Hamming weight leakage model with Gaussian noise is assumed), we show in Appendix B that the coefficient ρ_{aff} obtained for the zero-value second-order leakage of affine masking satisfies:

$$\rho_{\text{aff}} = \frac{n}{(4\sigma^2 + n)\sqrt{2^n - 1}}, \tag{8}$$

where n is the bit-size of Z.

We also computed the correlation coefficient corresponding to the 3^{rd}-order leakage of affine masking. We did not obtained explicit formulae for this coefficient but we observed for several values of n and σ that it was always lower that ρ_{aff}. This suggests that HO-DPA with normalized product combining works better against the 2^{nd}-order leakage of affine masking than against the 3^{rd}-order one. From our analysis, we therefore concluded that ρ_{aff} not only quantifies the resistance of affine masking against 2^{nd}-order DPA, but also that against HO-DPA in general.

Regarding Boolean masking, it has been shown in [25] that the correlation ρ_{bool} corresponding to HO-DPA with normalized product combining against d^{th}-order Boolean masking satisfies (in the Hamming weight model):

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

[5] This is actually true whatever the leakage function and noise distribution as a direct consequence of Lemma 2.

Let us denote by N_{aff} (resp. N_{bool}) the number of leakage measurements for a successful attack on affine masking (resp. Boolean masking). Since, according to [14], N_{aff} and N_{bool} are respectively roughly quadratic in the values of the inverse of the correlation coefficients, the *ratio* $\frac{N_{\text{aff}}}{N_{\text{bool}}}$ satisfies:

$$\frac{N_{\text{aff}}}{N_{\text{bool}}} \approx \left(\frac{\rho_{\text{bool}}}{\rho_{\text{aff}}}\right)^2 = \frac{2^n-1}{n}\left(\frac{1}{n+4\sigma^2}\right)^{1-d} . \tag{10}$$

Let ν denote the value $\frac{2^n-1}{n}\left(\frac{1}{n+4\sigma^2}\right)^{1-d}$. In view of (10), affine masking is more resistant to HO-DPA than d^{th}-order Boolean masking if and only if $\nu \geq 1$. Comparing the resistance of Boolean masking and affine masking against HO-DPA thus amounts to study when $\nu \geq 1$ is satisfied. Let us study this inequality with respect to d:

- When $d = 1$, we have $\nu \geq 1$ for all $n \geq 1$ whatever σ. We deduce that affine masking is more resistant to HO-DPA than first-order Boolean masking for all SNRs. Moreover, from (10), we expect that HO-DPA against first-order Boolean masking required around 32 times more leakage measurements than against affine masking whatever σ.
- For $d = 2$, $\nu \geq 1$ if and only if $\sigma^2 \leq (2^n - n^2 - 1)/4n$. This implies that for the case of AES where $n = 8$, affine masking is more resistant to HO-DPA than 2^{nd}-order Boolean masking if $\sigma \leq 2.44$, which corresponds to a SNR greater than 0.335.
- For $d \geq 3$, ν is always smaller than 1 for every $n \geq 1$. Affine masking is hence less resistant to HO-DPA than 3^{rd}-order Boolean masking for all SNRs.

Eventually Fig. 2 plots the correlation values ρ_{bool} for $d \in \{1, 2, 3\}$, ρ_{aff} (2O-DPA against affine masking) as well as the correlation values obtained for the third-order DPA against affine masking. It illustrates the fact that the correlation corresponding to the 3^{rd}-order leakage of affine masking is always lower than that corresponding to the 2^{nd}-order leakage of affine masking. Moreover and as expected, it shows that the coefficient ρ_{aff} is always lower than ρ_{bool} for $d = 1$, always greater than ρ_{bool} for $d = 3$, and lower than ρ_{bool} $d = 2$ only when $\sigma \leq 2.44$.

3.4 Attack Experiments

In order to confront the theoretical analyses conducted in the previous sections to practice, we performed several attack experiments. In a first place, we applied several side-channel distinguishers to leakage measurements simulated in the Hamming weight model with Gaussian noise. We not only applied (HO)-DPA, but also two other kinds of attacks, namely (higher-order) *Mutual Information Analysis* (MIA) and *Template Attacks* (TA). We chose to test these three side-channel distinguishers against the different kinds of masking firstly because they are the most widely used in the literature, and secondly because they represent a brand spectrum of adversary capabilities. As already mentioned, HO-DPA with

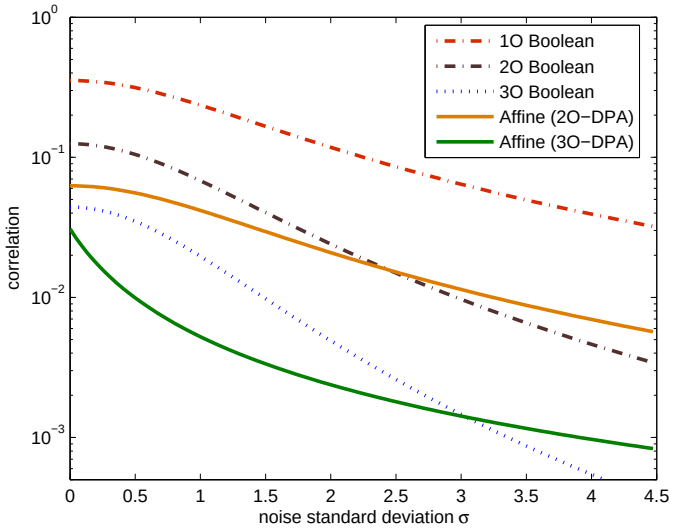


Fig. 2. Correlation values with respect to σ (logarithmic scale).

normalized product combining is the most efficient unprofiled attack against Boolean masking. On the other hand HO-MIA does not rely on a specific combining function, which is of interest for a fair comparison between Boolean and affine masking. Eventually, assuming that the adversary's templates are perfect, template attacks are the best possible attacks and hence they give the maximal security level reached by each kind of masking. Our methodology enabled us to observe how the different attacks perform against affine masking and to compare its resistance with that of the Boolean/multiplicative masking for different SNRs. Afterward, we performed some attacks against real power consumption measurements of smart-card implementations in order to check our observations in a real-world context.

Attack simulations. The leakage measurements have been simulated as samples of the random variables L_i defined according to (2) 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 $\mathsf{S}(X \oplus k^\star)$ where X represents a varying plaintext byte and $k^\star$ represents the key byte to recover.

Side-channel distinguishers. We applied higher-order DPA such as described in Sect. 3.3 and we also applied higher-order MIA (HO-MIA) and template attacks. In a higher-order MIA [21, 7], the correlation coefficient is replaced by the mutual information: the guess k is tested by estimating $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$. Since the

mutual information is a multivariate operator, this approach does not involve a combining function. In a template attack [4, 17], the attacker owns some *templates* of the leakage that he previously acquired during a profiling phase (see for instance [26, 13]). More precisely, he has some estimations of the probability distributions $(\ell, z) \mapsto \Pr[\mathbf{L} = \ell | Z = z]$. Based on those estimations, the attacker tests a guess k by estimating the likelihood $\Pr[k^\star = k | \mathbf{L}, X]$.

Target variables. Each attack was applied against the leakages of affine masking, multiplicative masking and Boolean masking. The target variables are those listed in Sect. 3.2 for Z being $\mathsf{S}(X \oplus k^\star)$.

Prediction functions. For each (HO-)DPA, we chose $\hat{\varphi}$ to be the optimal prediction function (6). As explained in Sect. 3.3, this leads us to select the dirac function δ_0 in the attacks against the zero-value 2^{nd}-order leakage of affine masking (resp. the zero-value 1^{st}-order leakage of multiplicative masking) and, according to [25], to select the Hamming weight function in the attacks against Boolean masking of any order.

For the (HO-)MIA attacks, we chose $\hat{\varphi}$ such that it maximizes the mutual information $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$ for $k = k^\star$ while ensuring discrimination (*i.e.* the mutual information must be lower for $k \neq k^\star$). As a direct consequence of Lemma 2, we chose $\hat{\varphi} = \delta_0$ to attack the zero-value 2^{nd}-order leakage. Naturally, we did the same choice for the zero-value 1^{st}-order leakage of multiplicative masking. For the third-order MIA on affine masking (and second-order MIA on multiplicative masking), $\hat{\varphi}$ was chosen to be the identity function since it maximizes $\mathrm{I}(\hat{\varphi}(Z); \mathbf{L})$. However, for the attack to succeed with such a choice, the target sensitive variable Z must be such that the function $X \mapsto Z = f_{k^\star}(X)$ (where X is the plaintext part involved in Z) is not injective [8, 21]. This constrained us to slightly modify the target variables for these attacks. Against affine masking, we targeted an affinely masked S-box output $G(\mathsf{S}(X \oplus k^\star))$ and an affinely masked plaintext byte $G(X')$ (together with the multiplicative mask R_1), which by Lemma 1 yields a non-injective function $(X, X') \mapsto Z = \mathsf{S}(X \oplus k^\star) \oplus X'$. Against multiplicative masking, we targeted the bitwise addition between two S-box outputs, which yields a non-injective function $(X, X') \mapsto Z = \mathsf{S}(X \oplus k^\star) \oplus \mathsf{S}(X' \oplus k^{\star\prime})$. Eventually, every HO-MIA against Boolean masking was performed with $\hat{\varphi} = \mathrm{HW}$ since the distribution of $(\mathrm{HW}(Z \oplus R_0), \mathrm{HW}(R_0))$ only depends on $\mathrm{HW}(Z)$, and therefore $\mathrm{I}\big(Z; (\mathrm{HW}(Z \oplus R_0), \mathrm{HW}(R_0))\big) = \mathrm{I}\big(\mathrm{HW}(Z); (\mathrm{HW}(Z \oplus R_0), \mathrm{HW}(R_0))\big)$ (the same argument holds for every masking order).

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

Leakage templates. For the template attacks, the attacker's templates were assumed to be perfect. In our context, this means that the attacker is aware of $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ for every i, and he uses this knowledge to evaluate the real probabilities $\Pr[\mathbf{L} | Z]$.

Attack simulation results. Each attack simulation was performed 100 times for various SNR values ($+\infty$, 1, 1/2, 1/5 and 1/10), that is, for several noise standard deviation values (0, $\sqrt{2}$, 2, $\sqrt{10}$ and $2\sqrt{5}$). Table 2 summarizes the number of leakage measurements required to observe a success rate of 90% in retrieving $k^\star$ for the different attacks.

Table 2. Number of leakage measurements for a 90% success rate.

Attack \ SNR	$+\infty$	1	1/2	1/5	1/10
Unprofiled 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	$> 10^6$	$> 10^6$
Unprofiled Attacks against Multiplicative Masking					
1O-DPA on Multiplicative Masking	900	1500	2500	4000	7500
1O-MIA on Multiplicative Masking	700	2500	3500	5500	15000
2O-DPA on Multiplicative Masking	2500	7500	20 000	60 000	220 000
2O-MIA on Multiplicative Masking	4000	35 000	55 000	100 000	200 000
Unprofiled Attacks against Affine Masking					
2O-DPA on Affine Masking	6500	20 000	45 000	170 000	650 000
2O-MIA on Affine Masking	5500	100 000	600 000	$> 10^6$	$> 10^6$
3O-DPA on Affine Masking	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
3O-MIA on Affine Masking	100 000	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
Profiled Attacks					
2O-TA on Boolean Masking	20	500	1200	7000	20 000
3O-TA on 2O Boolean Masking	20	8000	35 000	300 000	$> 10^6$
1O-TA on Multiplicative Masking	500	1300	1900	4000	7000
2O-TA on Multiplicative Masking	60	900	1400	4000	8000
2O-TA on Affine Masking	1300	15 000	45 000	200 000	$> 10^6$
3O-TA on Affine Masking	260	15 000	35 000	200 000	10^6

The results presented in Table 2 show the significant gain of security induced by affine masking compared to multiplicative and first-order Boolean masking. Some more specific observations are reported hereafter.

– **Affine masking *versus* multiplicative masking.** In all scenarios, affine masking is more resistant than multiplicative masking. When the SNR decreases, the resistance of affine masking increases faster than that of multiplicative masking. This is a consequence of the fact that affine masking is perfectly secure against first-order attacks which is not the case of multiplicative masking.

– **Affine masking *versus* Boolean masking.** When compared to first-order Boolean masking, a successful HO-DPA requires between 30 and 40 more leakage measurements against affine masking. For low noises (*i.e.* high SNRs), HO-DPA is also less efficient against affine masking than against second-order masking. The tide is turned when noise increases, which corroborates that higher-order masking combined with noise provides good resistance to SCA [3]. These results validate the theoretical analysis done in Sect. 3.3, where it is expected that affine masking is around 32 times more resistant than first-order Boolean masking and more resistant than second-order Boolean masking only when the SNR is greater than 0.335. On the other hand, the results of template attacks confirm that affine masking is always more resistant than first-order Boolean masking, and that it is also more resistant than second-order Boolean masking for high SNRs. It is interesting to note the strong correlation between the information theoretic evaluation of Sect. 3.2 and the efficiency of template attacks. We see that template attacks are more efficient against second-order Boolean masking than against affine masking when the SNR is greater than $1/2$ (*i.e.* $\sigma < 2$) which corresponds to the situation where the information leakage of affine masking is lower than that of second-order Boolean masking according to Fig. 1. This observation is in accordance with the argumentation of [28] that the mutual information metric is related to the efficiency of template attacks.

– **$3^{\mathbf{rd}}$-order attacks against affine masking.** It can be observed that targeting the multiplicative mask to mount a third-order attack against affine masking does not improve the efficiency of unprofiled attacks. On the contrary, they become clearly inefficient. This is quite natural for the third-order DPA using the product combining since unlike for an additive mask, such a combination is not suitable to remove a multiplicative mask. Therefore, the contribution of the third leakage to the combined leakage mainly acts as a noise, which renders the attack inefficient. For third-order MIA, the efficiency loss may result from the fact that precise estimations of 3-variate densities require significantly more samples than for bivariate densities which slows down the attack efficiency convergence. For template attacks, targeting the multiplicative mask improves the attack efficiency for high SNRs. However when the noise increases the efficiency of 2O-TA and 3O-TA against affine masking become similar. Once again, this corroborates the information theoretic evaluation of Sect. 3.2 which shows that the information provided by the third-order leakage of affine masking get closer to that provided by the second-order leakage as the noise increases.

– **(HO-)MIA *versus* (HO-)DPA.** (HO-)MIA attacks are always less efficient than the corresponding (HO-)DPA. A possible explanation is that the measurements are simulated in the Hamming weight model which is a situation more favorable to DPA attacks than to MIA attacks. A second possible explanation is that the rule proposed in [8] for the bin-widths selection in the MIA is not suitable when targeting affine masking. This point is let for further research.

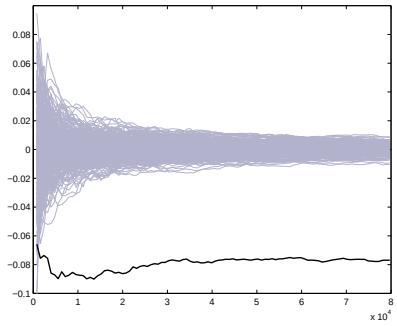
Practical attacks. The attack experiments reported in the following figures have been carried out against software implementations of the AES S-box protected with (1) affine masking, or (2) multiplicative masking, or (3) first-order Boolean masking or (4) second-order Boolean masking. The codes were executed on a 8051 microcontroller. We measured a SNR between 0.5 and 0.6 and our estimation of the correlation coefficient between the hamming weight of the manipulated data and the corresponding leakage was around 0.4.

We chose to only apply non-profiled side-channel attacks since the adversaries considered by the embedded security industry often have a limited access to the device and are not able to chose the values of the sensitive data that are manipulated. For each category of attacks (HO-)DPA and (HO-)MIA and for each targeted countermeasure, we moreover chose to only implement the most efficient attack strategy (*e.g.* we did not apply 3O-DPA against affine masking since our simulations show that it is much less efficient than the 2O-DPA). The results of these attacks are plotted in Fig. 3. The x-axis refers to the number of measurements and the y-axis refers to the correlation coefficient values for the (HO)-DPA and to the logarithm of the mutual information for the 2O-MIA.

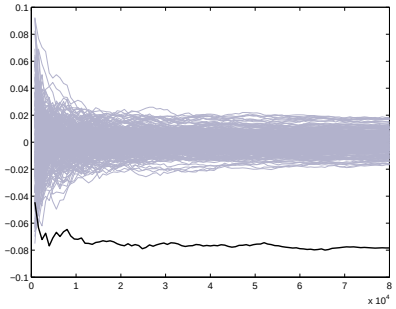
It can be noticed that our attack experiments corroborate quite well our attack simulations for an SNR equal to 1/2. The multiplicative masking is broken by the first-order DPA with dirac prediction function after around 1000 measurements, whereas the first-order Boolean masking is defeated by the second-order DPA with Hamming weight prediction after around 1600 measurements. As expected, the affine masking resists much better to the second-order DPA. It needs more than 62000 measurements to unambiguously discriminate the correct key. The *ratio* between the two second-order DPA is approximatively 38 which is close to the *ratio* value 32 predicted by our theoretical analysis. The fact that the information does not perfectly leak in the Hamming weight model probably explain the small difference. Eventually, our 2O-MIA attack experimentations are also in accordance with our attacks simulated for an SNR equal to 1/2. The first-order Boolean masking is broken with around 1200 measurements, whereas 80000 measurements do not allow us to break the affine masking by 2O-MIA.

4 Conclusion

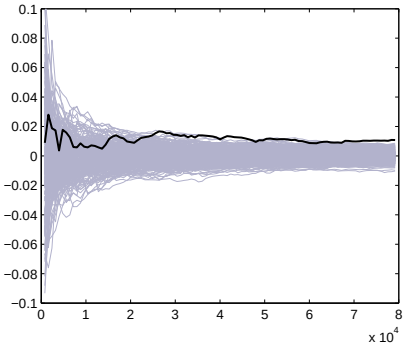
In this paper, we introduced affine masking as an alternative to the commonly used Boolean masking to protect implementations of block ciphers against side channel analysis. The principle is to mask each sensitive variable both additively and multiplicatively in order to complicate the masking relation and therefore achieve better higher-order resistance in practice. We described an affine masking scheme for AES and we provided some implementation results for our scheme. Moreover, we conducted an in-depth analysis which demonstrates that affine masking significantly improves the resistance to higher-order SCA compared to Boolean masking. This analysis together with our implementation tests clearly show that the proposed scheme provides a good performance-security trade-off compared to existing countermeasures.



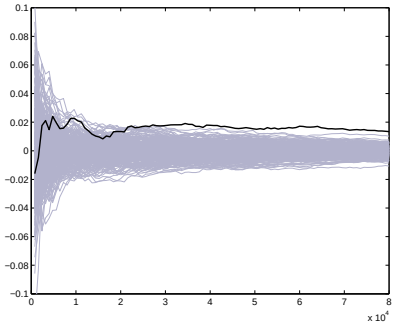
(a) 1O-DPA against multiplicative masking.



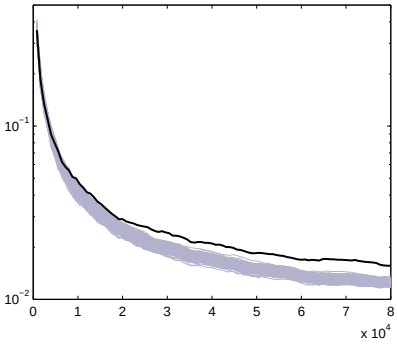
(b) 2O-DPA against Boolean masking.



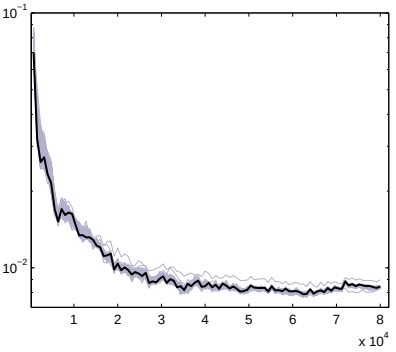
(c) 2O-DPA against affine masking.



(d) 3O-DPA against Boolean masking.



(e) 2O-MIA against Boolean masking.



(f) 2O-MIA against affine masking.

Fig. 3. Practical attacks on a software AES implementation.

References

1. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç. 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. E. 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.
3. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Wiener [31], pages 398–412.
4. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In Kaliski Jr. et al. [11], pages 13–29.
5. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Paillier and Verbauwhede [19], pages 28–44.
6. G. Fumaroli, E. Mayer, and R. Dubois. First-Order Differential Power Analysis on the Duplication Method. In K. Srinathan, C. P. Rangan, and M. Yung, editors, *Progress in Cryptology – INDOCRYPT 2007*, volume 4859 of *LNCS*, pages 210–223. SV, 2007.
7. B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede. Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. Cryptology ePrint Archive, Report 2009/228, 2009. http://eprint.iacr.org/.
8. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
9. J. Golić and C. Tymen. Multiplicative Masking and Power Analysis of AES. In Kaliski Jr. et al. [11], pages 198–212.
10. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Ç. 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.
11. B. Kaliski Jr., Ç. Koç, and C. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*. Springer, 2002.
12. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Wiener [31], pages 388–397.
13. K. Lemke-Rust and C. Paar. Gaussian mixture models for higher-order side channel analysis. In Paillier and Verbauwhede [19], pages 14–27.
14. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks – Revealing the Secrets of Smartcards*. Springer, 2007.
15. T. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
16. T. Messerges. Using Second-order Power Analysis to Attack DPA Resistant Software. In Ç. 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.

17. E. Oswald and S. Mangard. Template Attacks on Masking—Resistance is Futile. In M. Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 243–256. Springer, 2007.
18. E. Oswald, S. Mangard, C. Herbst, and S. Tillich. Practical Second-order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In Pointcheval [20], pages 192–207.
19. P. Paillier and I. Verbauwhede, editors. *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*. Springer, 2007.
20. D. Pointcheval, editor. *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*. Springer, 2006.
21. E. Prouff and M. Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *Applied Cryptography and Network Security – ANCS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518. Springer, 2009.
22. E. Prouff, M. Rivain, and R. Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.*, 58(6):799–811, 2009.
23. M. Rivain, E. Dottax, and E. Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In T. Baignères and S. Vaudenay, editors, *Fast Software Encryption – FSE 2008*, Lecture Notes in Computer Science, pages 127–143. Springer, 2008.
24. M. Rivain and E. Prouff. Provably secure higher-order masking of aes. In S. Mangard and F.-X. Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010. ISBN 978-3-642-15030-2.
25. M. Rivain, E. Prouff, and J. Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.
26. W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*. Springer, 2005.
27. K. Schramm and C. Paar. Higher Order Masking of the AES. In Pointcheval [20], pages 208–225.
28. F.-X. Standaert, T. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
29. 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. Cryptology ePrint Archive, Report 2010/180, 2010. http://eprint.iacr.org/.
30. M. 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.
31. M. Wiener, editor. *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

A Computing the product in GF(256)

Affine masking involves multiplications in the field $\mathrm{GF}(2^n)$. The most efficient way to implement the product in the field GF(256) is to use log/alog tables. These tables are constructed using the fact that all non-zero elements in a finite field $\mathrm{GF}(2^n)$ can be obtained by exponentiation of a generator in this field. For a generator α of $\mathrm{GF}(256)^*$ we define $\log(\alpha^i) = i$ and $\mathrm{alog}(i) = \alpha^i$. This results are stored in two tables of $2^n - 1$ words of n bits.

If a, b are non-zero, then the product $a \cdot b$ can be computed using log/alog tables as

$$a \cdot b = \mathrm{alog}[(\log(a) + \log(b)) \bmod (2^n - 1)]. \tag{11}$$

In order to compute the addition modulo $2^n - 1$, let a, $b \in \mathrm{GF}(2^n)$, and let c denote the carry associated with the operation $a + b \bmod (2^n)$. Then, $a + b \bmod (2^n - 1)$ can be computed from $a + b \bmod (2^n)$ and c as follows.

Algorithm 2
INPUT: $a, b \in \mathrm{GF}(2^n)$
OUTPUT: $s = a + b \bmod (2^n - 1)$

1. $s \leftarrow a + b \bmod 2^n$
2. $s \leftarrow s + c \bmod 2^n$
3. **if** $s = 2^n - 1$ **then** $s = 0$
4. Return s

In the case of affine masking, one of the two operators of the product is always non-zero. Let $\delta_{\{0\}}(i)$ be the indicator function of the set $\{0\}$. Assuming that a is always non-zero, then the product $a \cdot b$ can be computed as

$$a \cdot b = (1 - \delta_{\{0\}}(b)) \times \mathrm{alog}[(\log(a) + \log(b)) \bmod (2^n - 1)]. \tag{12}$$

Similarly the inversion of a non-zero element $a \in \mathrm{GF}(2^n)$ can be implemented using log/alog tables as

$$a^{-1} = \mathrm{alog}[-\log(a) \bmod (2^n - 1)]. \tag{13}$$

B Optimal Correlation for 2O-DPA on Affine Masking

Let us assume that the leakages L_1 and L_2 follow the Hamming weight model with Gaussian noise, that is the leakage function is defined as $\varphi(\cdot) = \delta + \mathrm{HW}(\cdot)$ where δ is a constant and the noises B_i's have (independent) Gaussian distributions $\mathcal{N}(0, \sigma^2)$. Normalized product combining was proven in [22] to be the most efficient known combining function for exclusive-or masked variables. After normalization, the two leakage variables $\overline{L}_1 = L_1 - \mathrm{E}[L_1]$ and $\overline{L}_2 = L_2 - \mathrm{E}[L_2]$ respectively satisfy:

$$\overline{L}_1 = -\frac{n}{2} + \mathrm{HW}(R_1 Z \oplus R_0) + B_1 \tag{14}$$

and

$$\overline{L}_2 = -\frac{n}{2} + \mathrm{HW}(R_0) + B_2, \tag{15}$$

where B_1 and B_2 are assumed to be two independant random variables with mean 0 and standart deviation σ.

The optimal correlation ρ_{aff} for the correct key hypothesis can be obtained from Corollary 8 in [22]:

$$\rho_{\mathrm{aff}} = \sqrt{\frac{\mathrm{Var}\left[\mathrm{E}\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right]\right]}{\mathrm{Var}\left[\overline{L}_1 \times \overline{L}_2\right]}} \tag{16}$$

Since $R_1 Z$ is uniformly distributed over $\mathrm{GF}(2^n)$, the formula for $\mathrm{Var}\left[\overline{L}_1 \times \overline{L}_2\right]$ given in [22] in the Boolean masking setting can also be applied in our context:

$$\mathrm{Var}\left[\overline{L}_1 \times \overline{L}_2\right] = \frac{n^2}{16} + \frac{n}{2}\sigma^2 + \sigma^4. \tag{17}$$

We are left with evaluating $\mathrm{Var}\left[\mathrm{E}\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right]\right]$. We have the following result.

Proposition 1. *Let $\overline{L}_1$ and $\overline{L}_2$ satisfy* (14) *and* (15)*. Then for every $z \in \mathrm{GF}(2)^n$, we have*

$$E\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right] = \begin{cases} \frac{n}{4} & \text{if } z = 0, \\ -\frac{n}{4(2^n-1)} & \text{otherwise.} \end{cases} \tag{18}$$

Proof. Since $\mathrm{E}\left[B_1\right] = \mathrm{E}\left[B_2\right] = 0$, we have

$$\mathrm{E}\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right] = -\frac{n^2}{4} + \mathrm{E}\left[\mathrm{HW}(R_1 Z \oplus R_0) \times \mathrm{HW}(R_0) | Z = z\right]. \tag{19}$$

Let $\alpha(z) = \mathrm{E}\left[\mathrm{HW}(R_1 Z \oplus R_0) \times \mathrm{HW}(R_0) | Z = z\right]$. We have

$$\alpha(0) = \mathrm{E}\left[\mathrm{HW}(R_0)^2\right] = \frac{n^2 + n}{4}. \tag{20}$$

Now let $z \neq 0$ and $X = R_1 z$. Then

$$\begin{aligned} \alpha(z) &= \mathrm{E}\left[\mathrm{HW}(X \oplus R_0) \times \mathrm{HW}(R_0)\right] && (21) \\ &= \frac{1}{2^n(2^n-1)} \sum_{r_0} \mathrm{HW}(r_0) \sum_{x \neq 0} \mathrm{E}\left[\mathrm{HW}(x \oplus r_0)\right] && (22) \\ &= \frac{1}{2^n(2^n-1)} \sum_{r_0} \mathrm{HW}(r_0) \times \left(2^n \frac{n}{2} - \mathrm{HW}(r_0)\right) && (23) \end{aligned}$$

By simplifying (23), we get

$$\alpha(z) = \frac{n^2}{4} - \frac{n}{4(2^n-1)}. \tag{24}$$

Finally, (19), (20) and (24) leads to (18). □
□

From (18), we obtain

$$\mathrm{Var}\left[\mathrm{E}\left[\overline{L}_1 \times \overline{L}_2 | Z = z\right]\right] = \frac{n^2}{16(2^n - 1)}. \tag{25}$$

Finally, (16), (17) and (25) leads to

$$\rho_{\mathrm{aff}} = \frac{n}{(4\sigma^2 + n)\sqrt{2^n - 1}}. \tag{26}$$

As an illustration to (26), Table 3 gives some values of the optimal correlation for $n \in \{1, \ldots, 8\}$ and $\sigma \in \{0, 1, 5, 10\}$ in the affine masking setting.

Table 3. Optimal correlation for 2O-DPA on affine masking

$\sigma \backslash n$	1	2	3	4	5	6	7	8
0	1.00000	0.57735	0.37796	0.25820	0.17961	0.12599	0.08874	0.06262
1	0.20000	0.19245	0.16198	0.12910	0.09978	0.07559	0.05647	0.04175
5	0.00990	0.01132	0.01101	0.00993	0.00855	0.00713	0.00581	0.00464
10	0.00249	0.00287	0.00281	0.00256	0.00222	0.00186	0.00153	0.00123