Analysis of a Code-Based Countermeasure Against Side-Channel and Fault Attacks

Guillaume Barbu and Alberto Battistello(✉)

Oberthur Technologies, Security Group, Cité Photonique, Batiment ELNATH, 1er étage 1, allée des Lumières, 33600 Pessac, France
{g.barbu,a.battistello}@oberthur.com

Abstract. The design of robust countermeasures against Side-Channel Analysis or Fault Attacks is always a challenging task. At WISTP'14, a single countermeasure designed to thwart in the same effort both kinds of attacks was presented. This countermeasure is based on coding theory and consists in a specific encoding of the manipulated data acting in the same time as a random masking and an error detector. In this paper, we prove that this countermeasure does not meet the ambitious objectives claimed by its authors. Indeed, we exhibit a bias in the distribution of the masked values that can be exploited to retrieve the sensitive data from the observed side-channel leakage. Going further, we show that this bias is inherent to the nature of the encoding and that randomizing the code itself can be useful to reduce the bias but cannot completely fix the scheme.

Keywords: Side-channel analysis · Fault attacks · Coding theory · Countermeasure · AES

1 Introduction

Since the introduction of side-channel analysis and fault attacks against cryptographic implementations in the late 90s, the scientific community, both academic and industrial, has engaged a great effort in designing robust and efficient countermeasures to counteract these attacks. Usually, each countermeasure is designed to tackle only one of these two kinds of attacks. For instance, boolean masking [1] of key-dependent data is meant to avoid information leakage through a side-channel medium. On the other hand, time-redundant or data-redundant computations are implemented to detect fault injections during the execution of the algorithm.

Following the idea first introduced in [2] the authors of [3] proposed at WISTP'14 a new countermeasure named ODSM (for Orthogonal Direct Sum Masking) based on coding theory and showed how it could be applied to protect an AES implementation. Besides the application of code-based techniques, one of the novelty of ODSM is that the same countermeasure aims at defeating both side-channel analysis (SCA) and fault attacks (FA) at once.

S. Foresti and J. Lopez (Eds.): WISTP 2016, LNCS 9895, pp. 153–168, 2016.
DOI: 10.1007/978-3-319-45931-8_10

By introducing a random mask in the encoding of a sensitive data, ODSM aims at decorrelating the side-channel leakage from the value of the sensitive variable. At the same time, by taking advantage of the error detection capability of the code, the scheme also allows to control the integrity of the manipulated data and eventually to detect induced faults.

Although the proposed countermeasure is pretty elegant from a theoretic point of view and that a proof of security is presented in the original article, we demonstrate in the following that such a proposal fails at ensuring resistance against SCA.

This article is organized as follows. Section 2 recalls some basic concepts of coding theory and defines some notions regarding side-channel attacks. Section 3 introduces the ODSM countermeasure described in [3] and its application to the AES. Section 4 gives the result of our analysis of the masking scheme with regard to SCA. In Section 5 we provide further evidence of the insecurity of the scheme by mounting a template attack against a real device. Section 6 proposes some improvements of the countermeasure to achieve a better resistance against SCA while preserving the FA resistance. Finally Sect. 7 presents concluding remarks and future works.

2 Preliminaries

This section gives the elementary notions required to both apprehend the potential of an attacker and follow the masking scheme design of [3].

2.1 Passive Side-Channel Analysis

Previously used by several intelligence agencies, but formally introduced to the academic community by Kocher *et al.* only in 1996, Side-Channel Attacks have revolutionized the world of cryptography [4,5]. In particular it is now common knowledge that the observation of physical interactions between a hardware executing a cryptographic algorithm and the surrounding environment may allow retrieving values of internally manipulated sensitive variables. Simple Power Analysis, consists in retrieving a secret value by simple observation of the power consumption of the device during a process depending on this secret value. As an example, on naive RSA implementations, the execution time may leak critical information on the value of secret key bits.

More complex attacks like Differential Power Analysis (DPA) make use of the statistical dependency between the observed interactions and the sensitive values. Concretely, in order to retrieve the secret key of an embedded AES, an attacker asks the ciphering of several known messages and observes the power consumption of the processor during the execution of the algorithm. After collecting enough observations, by correlating the messages and the power consumptions the attacker can retrieve the secret AES key on unprotected implementations.

Finally, Higher-Order Side-Channel Analysis (HO-SCA) consists in the higher order statistics analysis of several leakages in order to retrieve the corresponding key. SPA, DPA and all their variants have been used to break the security of many algorithms [6,7], the obvious effect was the design of efficient and effective countermeasures. While many ideas have been found [1,8,9], it remains hard and very costly to provide efficient countermeasures that guarantee security versus high-order attacks.

In the following we present the countermeasure of [3] which aims at thwarting both faults and side-channel analysis at once. The countermeasure makes use of results from coding theory which we recall in the next section.

2.2 Notions of Coding Theory

The following gives the few definitions and notations necessary to ease the reading and understanding of the masking scheme presented in Sect. 3.

Definition 1 (Binary Linear Code). *A binary linear code $\mathcal{C}$ of length n and dimension k is a linear subspace of dimension k of the vector space $\mathbb{F}_2^n$.*

A word of the code $\mathcal{C}$ is then a vector w such that $w \in \mathcal{C}$.

Definition 2 (Supplement of vector space). *The supplement of the vector space $\mathcal{C}$ in $\mathbb{F}_2^n$ is the set of vectors $\mathcal{D}$ such that $\mathcal{C} \oplus \mathcal{D} = \mathbb{F}_2^n$, where $\oplus$ denotes the direct sum of two vector spaces.*

An element z in $\mathbb{F}_2^n$ can thus be decomposed uniquely as the sum of two elements c and d, respectively in $\mathcal{C}$ and $\mathcal{D}$:

$$z = c \oplus d \tag{1}$$

Definition 3 (Generating matrix). *The vectors of the basis of a linear code $\mathcal{C}$ forms the generating matrix of $\mathcal{C}$.*

In the following we will denote respectively by G and H the generating matrices of $\mathcal{C}$ and $\mathcal{D}$. Then every element of $\mathcal{C}$ (resp. $\mathcal{D}$) can be written uniquely as xG (resp. yH) for some x (resp. y) in $\mathbb{F}_2^{dim(\mathcal{C})}$ (resp. $\mathbb{F}_2^{dim(\mathcal{D})}$) and (1) can be rewritten as:

$$z = xG \oplus yH \tag{2}$$

Definition 4 (Dual code). *The dual code of $\mathcal{C}$ is the linear code $\mathcal{C}^\perp = \{w \in \mathbb{F}_2^n | \forall c \in \mathcal{C},\ c \cdot w = 0\}$.*

Definition 5 (Parity matrix). *The parity matrix of $\mathcal{C}$ is the generating matrix of $\mathcal{C}^\perp$.*

In the case where $\mathcal{C}^\perp = \mathcal{D}$, it comes straightforwardly that the dimension of $\mathcal{D}$ is $n - k$ if the dimension of $\mathcal{C}$ is k. Also the parity matrix of $\mathcal{C}$ is H.

Finally, we recall the proposition from [3] stating a necessary and sufficient condition to have $\mathcal{C}$ and $\mathcal{C}^\perp$ supplementary in $\mathbb{F}_2^n$:

Proposition 1. *Without loss of generality (a permutation of coordinates might be necessary), we can assume that the generating matrix of $\mathcal{C}$ is systematic, and thus takes the form $[I_k \| M]$, where I_k is the $k \times k$ identity matrix. The supplementary $\mathcal{D}$ of $\mathcal{C}$ is equal to $\mathcal{C}^{\perp}$ iff the matrix $I_k \oplus MM^T$ is invertible.*

This proposition is necessary for the construction of the masking scheme, as will be explained in the following section.

3 Orthogonal Direct Sum Masking and Its Application to AES

In this section, we introduce the Orthogonal Direct Sum Masking (ODSM) countermeasure as it was defined in [3]. Then we detail how the authors apply it in the case of an AES implementation.

3.1 Related Works

In [2], Bringer *et al.* introduced a masking scheme based on a specific encoding of the sensitive data and the corresponding mask. Following this scheme, the masking of sensitive data x with the random quantity m is obtained by computing $z = xF \oplus mG$, where:

- G is a generator matrix of the binary linear code $\mathcal{C}$ of length n, dimension k and minimum distance d
- F is a $k \times n$ matrix with k rows in $\{0,1\}^n$ all linearly independent of each other and not belonging to the binary linear code $\mathcal{C}$.

Recovering x from the encoded word z can be achieved by multiplying z by the parity matrix of $\mathcal{C}$: $zH^T = xFH^T \oplus mGH^T = xFH^T$. The authors then describe an application of this scheme to the AES.

However, as pointed out by Moradi in [10] certain limitations exist when choosing the matrix F in order to retrieve the sensitive value x from xFH^T. Namely, one should ensure that the application $x \mapsto xKH^T$ is bijective, *ie.* $(KH^T)^{-1}$ shall exist.

In addition, the author stresses that the application of the scheme to the AES requires a mask correction at each round of the cipher algorithm.

Finally, the work of Azzi *et al.* in [11] adapts the ODSM scheme to enhance the fault detection capability through non-linear functions. However this comes at the cost of additional computations with regards to the side-channel resistance property.

3.2 Orthogonal Direct Sum Masking

As previously stated, the construction of the ODSM lies on the fact that for the considered code $\mathcal{C}$, we have $\mathcal{C}^{\perp} \oplus \mathcal{C} = \mathbb{F}_2^n$. Indeed, in this case we have $G \cdot H^T = 0$

and H is the parity matrix of $\mathcal{C}$. Consequently, in (2) we can recover x and y from z:

$$x = zG^T(GG^T)^{-1} \tag{3}$$
$$y = zH^T(HH^T)^{-1} \tag{4}$$

The principle of the masking scheme consists in representing a sensitive k-bit data x by a n-bit data z according to (2), where y is an $(n-k)$-bit random mask.

The sensitive value x can be easily recovered from z by using (3). In addition, the integrity of the manipulated data z can be verified as often as required by checking the integrity of the mask y thanks to (4), which provides the security against fault injection.

Based on this principle, the authors suggest to perform computation within the encoded/masked representation. Actually they show how applying operations on the sensitive value x can be achieved by applying associated operations on the encoded value z. To reach this goal, they split the different operations required into three categories and show how to proceed in each one:

2-operand Operations. In this case, they focus their attention on the xor operation. Actually this case is quite straightforward since it is only necessary to encode the operand and perform the xor. For instance, supposing we need to xor a round key k_i to x, we would have to compute $z' = z \oplus k_i G$. And we can check that $x \oplus k_i$ is computed within the masking scheme:

$$z' = z \oplus k_i G = (x \oplus k_i)G \oplus yH \tag{5}$$

Binary Linear Operations. Let L be the matrix corresponding to the desired binary linear operation, then they suggest to construct a so-called *masked* binary linear operation whose corresponding matrix L' is constructed as follows:

$$L' = G^T(GG^T)^{-1}LG \oplus H^T(HH^T)^{-1}H \tag{6}$$

And we can check again that xL is correctly computed within the masking scheme when zL' is computed.

Nonlinear Transformations. In this case, a *masked* version S' of the transformation S can be computed:

$$\forall z \in \mathbb{F}_2^n,\ S'(z) = S(zG^T(GG^T)^{-1})G \oplus zH^T(HH^T)^{-1}H \tag{7}$$

$S(x)$ is correctly obtained within the masking scheme when computing $S'(z)$.

Following these guidelines, the authors claim that various computations can be carried out within the coset $\mathcal{C} \oplus d$ of the linear code $\mathcal{C}$, with d a mask randomly chosen in $\mathcal{D} = \mathcal{C}^\perp$.

3.3 ODSM in Practice: Application to the AES

The ODSM can be applied, in particular, to an implementation of the AES. For that purpose, the authors consider the 128-bit version of the cipher and propose to use the binary linear code of parameters [16,8,5] (meaning $n = 16$ and $k = 8$) which has a supplementary dual in $\mathbb{F}_2^{16}$.[1] Indeed, once the initial state has been encoded, the different operations of the AES can be straightforwardly constructed as previously described.

AddRoundKey. The round key bytes only need to be encoded before being added to the encoded state: $z = z \oplus kG$.

ShiftRows. The ShiftRows operation remains unchanged. It only processes 16-bit words instead of 8-bit.

MixColumns. MixColumns can be computed by using the linear application generated from the matrix of the XTIME application by (6).

SubBytes. For the SubBytes operation, two approaches were proposed in [3]. The first one is a look-up table approach which requires to precompute the 16-bit output $S'(z)$ for all z as per (7). We note that this method implies a quite heavy memory overhead as a 128 kB-table needs to be stored (2^{16} 16-bit values). The second approach actually performs the SubBytes outside of the code and thus involves the recomputation of a new masked S' transformation for each encryption to ensure a proper masking. It is then necessary to compute for all x in $\mathbb{F}_2^k$:

$$S'_{recomp}(x) = S(x \oplus x') \oplus x'', \quad \textit{with } x' \textit{ and } x'' \textit{ randomly chosen in } \mathbb{F}_2^k \tag{8}$$

The SubBytes is then performed as described in Algorithm 1. Our analysis is actually independent on the choice of either of the two approaches.

Algorithm 1. Masked SubBytes transformation on $z = xG \oplus yH$

$z = z \oplus x'G$;
$x = zG^T(GG^T)^{-1}$;
$x = S'_{recomp}(x)$;
$z' = xG \oplus yH$;
$z' = z' \oplus x''G$;
return z';

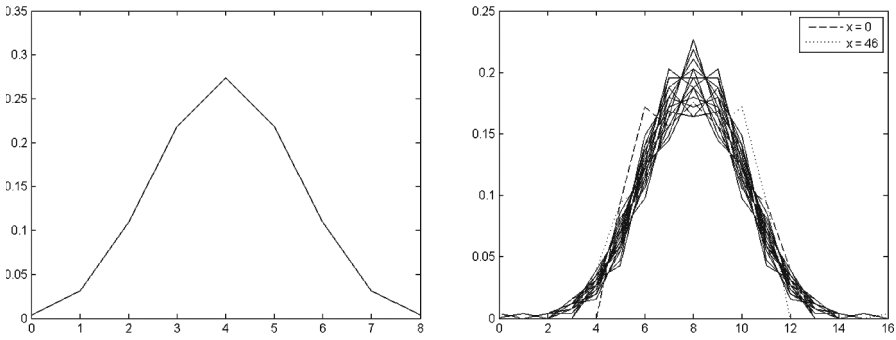
[1] For the generating and parity matrices G and H and for L and L' corresponding to the standard and *masked* versions of the XTIME linear application of this code, we refer the reader to [3].

4 Side-Channel Analysis of the Masking Scheme

In this section we provide a deep analysis of the side-channel resistance of the masking scheme suggested in [3]. We demonstrate that it is possible to mount a first-order side-channel attack versus the countermeasure meant to be resistant to high-order attacks.

4.1 Striking Differences

The authors of [3] proved the security of the ODSM scheme versus d-th order side-channel analysis, where $d + 1$ is the distance of the dual code $\mathcal{C}^{\perp}$. Thus, with respect to the parameters of [3], the code is proved to be secure versus 4-th order attacks, 5 being the minimal distance of the dual code. The proof of security relies on the observation that the expected value of the leakage is independent on the sensitive value x (up to the 4^{th} order statistical moments), after the encoding $xG \oplus yH$ with mask y.



(a) Boolean masking pdf.

(b) ODSM masking pdf.

Fig. 1. Pdf of the Hamming weights of the Boolean masking scheme vs ODSM masking scheme.

Figure 7 shows the expected probability density functions (pdf) of the Hamming weight (HW) of masked values for both boolean masking and the ODSM scheme. The results are obtained by collecting the distribution of the HW of $z = x \oplus y$ for Boolean masking (Fig. 1a), and $z = xG \oplus yH$ for ODSM, where x is fixed, and y takes all values in $\mathbb{F}_2^k$ (Fig. 1b).

As expected, in the boolean masking scheme the distributions are independent on the sensitive values, such that all distributions are superposed and only one curve is visible. On the other hand, in the ODSM scheme the distributions depend on the sensitive values and we can distinguish 22 different distributions, each one related to a particular set of sensitive values. In particular the distributions of the sensitive value $x = 0$ and $x = 46$ show striking differences.

We remark that an encoded value with HW of 0 can only be produced by encoding the sensitive value $x = 0$ with a mask $y = 0$. Similarly a HW of 16 can only be obtained when the encoded sensitive value equals 46. From Fig. 1b we thus observe that the extremum HW value 0 (resp. 16) is only present on the distribution of $x = 0$ (resp. $x = 46$). We further remark that for a sensitive encoded value equal to $x = 0$ (resp. $x = 46$), the HW of the encoded value can never be 4,3,2 or 1 (resp. 12, 13, 14, 15).

We show in the following that it is possible to exploit such striking differences by using 1^{st}-order statistics in order to retrieve the sensitive values.

4.2 Means-only Attack on the ODSM Distribution

We have noticed that the difference between the leakage distributions of the ODSM masking scheme and that of classical boolean masking scheme may lead to weaknesses that have not been taken into account in [3]. In this section we exhibit an actual attack that exploits these very differences to retrieve the key value by using only 1^{st}-order statistics on carefully selected leakages.

The basic idea of our attack comes by observing the distributions of Fig. 1b. The distributions present a left skewness (i.e.: asymmetry about the mean) for $x = 0$, and a right skewness when $x = 46$. While such skewness do not bias the average of all values, it does when the average is computed only on the leakages below a given Hamming weight. In practice we exploit the fact that the skewness preserves the mean only when computed on all values, but produces detectable biases on subsets of them.

Observing Fig. 1b one notice that the mean of all leakage values below 9 and those above 9 are not equals for all the 22 classes, in particular $z = 0G \oplus yH$ and $z = 46G \oplus yH$ should present remarkable differences due to the skewness. We thus argue that it provides a distinguisher for the values 0 and 46.

A more careful partitioning leads to even better and more accurate results. We divide the curves into two sets:

- a first set containing leakages with Hamming weight between 4 and 11,
- a second set for the remaining leakages.

The absolute difference of the two sets on theoretical distributions is depicted in Fig. 2, for each message, for all masks.

We further remark that this choice for the two sets allows to retrieve only the value corresponding to 46, as the skewness on 0 is not captured by such partitioning.

4.3 Simulations

In this section we show the results of the application of our attack described in Sect. 4.2 to simulated leakages of the ODSM scheme. For our simulations we computed the value $z = \text{Sbox}(m \oplus k_i)G \oplus yH$, where y is in $\mathbb{F}_{2^k}$ freshly regenerated at each execution. For each value z we computed the corresponding

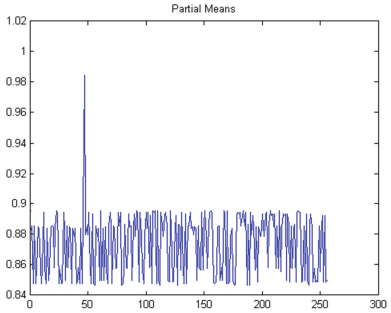


Fig. 2. Difference of the number of leakages between 4 and 11 and the rest.

leakage $\ell = HW(z) + B$, where B is a Gaussian noise with standard deviation σ. In order to evaluate the success rate of our attack with different noise levels, we have performed different campaigns where σ varies from 0 to 2. For each campaign we have simulated the leakage of 10, 000 computations for each byte value.

We start our attack by computing the minimum and the maximum values among all leakages. Then define the range s of all leakages as the difference between the maximum and minimum values, divided by 16. We then use this value to split the leakages into the two sets, the first containing those leakages whose value falls between $s * 4$ and $s * 11$ and the second with the remaining

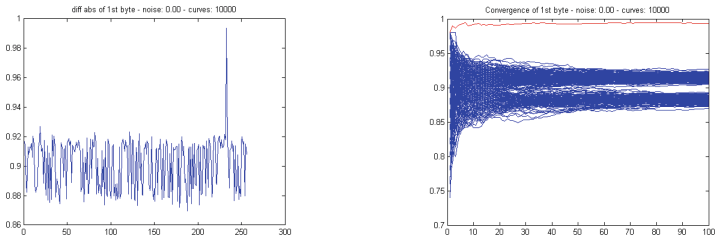


Fig. 3. Result of the attack with $\sigma = 0$ for 10, 000 leakages. (Color figure online)

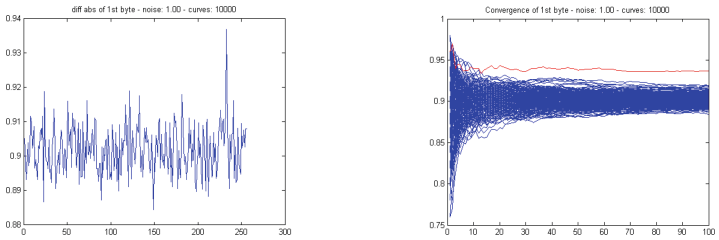


Fig. 4. Result of the attack with $\sigma = 1$ for 10, 000 leakages. (Color figure online)

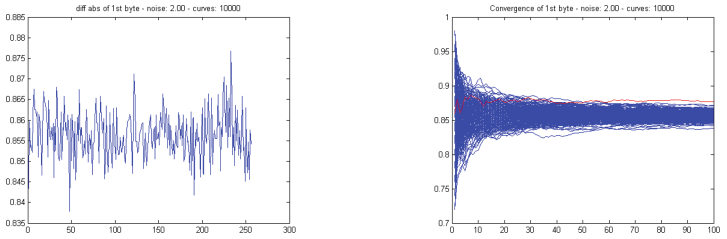


Fig. 5. Result of the attack with $\sigma = 2$ for 10, 000 leakages. (Color figure online)

leakages. We finally analyze the absolute difference of the two sets for each message. As observed in Fig. 2, only the message m which gives $\text{Sbox}(m \oplus k_i) = 46$ should produce a peak on the difference of means. Thus the peak found for a particular message m reveals $\text{Sbox}(m \oplus k_i) = 46$, so the attacker can retrieve the secret key byte $k_i = \text{Sbox}(46)^{-1} \oplus m$.

The key byte value used for our simulations is 43, thus we expect to obtain peaks for the message of value $233 = \text{Sbox}(46)^{-1} \oplus 43$. Figures 3, 4, 5 show the results obtained by using 10, 000 noisy executions for each message m. The left part corresponds to the value of the absolute differences for each message (thus for each of the 256 values we depict the value of the difference of means). The right part depicts the maximum value of the absolute difference for each

Algorithm 2. MEANS ATTACK ON AES-128 ODSM SCHEME.

```
// Find min and max values
l_max = max_{m∈F_2^k, 0≤i<♯curves}(leakage(m, i));
l_min = min_{m∈F_2^k, 0≤i<♯curves}(leakage(m, i));
// Derive HW boundaries
leakage_size=(l_max - l_min) / 16;
set1_limit = leakage_size × 4;
set2_limit = leakage_size × 11;
// Separate curves
for m from 0 to 255 do
    for i from 0 to 10,000 do
        if set1_limit ≤ leakage(m, i) ≤ set2_limit then
            set2(m)+= leakage(m, i);
        else
            set1(m)+= leakage(m, i);
        end
    end
end
// Select best candidate based on difference of means
best_message = max_m(abs(set2(m) − set1(m))/10,000);
return k = Sbox(46)^{-1} ⊕ best_message
```

key, sampled after each 100 curves. For right-side figures, the correct message hypothesis (233) is depicted in red.

We present the pseudocode of our attack in Algorithm 2.

We notice that our attack needs a huge number of curves to retrieve the key value even for relatively low noise simulations. For example, for $\sigma = 1$, we need about 384, 000 curves (1500 * 256) in order to retrieve the correct key hypothesis. The need for a considerable number of curves can be interpreted as a consequence of the masked values living in $\mathbb{F}_{2^16}^2$, and thus far more samples are required to obtain a representative sample of the underlying distribution. However, we remark that as soon as there is no noise, very few hundred traces are necessary to retrieve the correct key.

We want to stress the fact that despite the security proof given in [3], our attack shows that it is possible to retrieve the secret values protected with the ODSM scheme by using a first-order statistic on carefully selected leakages.

We finally insist on the fact that despite our attack applies here to the ODSM scheme with the parameters of [3], most choices of the code would succumb to such an attack.

5 Maximum Likelihood Attack

In this section we present a further attack to the countermeasure. As we have remarked in Section 4, the leakage corresponding to an HW of 0 can only be produced if both the sensitive value x and the mask y equal to 0. We thus suggest that it is possible to use a maximum likelihood (template) attack to distinguish the curves manipulating a variable z whose HW equals 0 from the others.

Template attacks are generally divided into two phases. In the first phase (profiling) the sensitive data is known to the attacker, for example she may employ an open sample, while in the second phase (attack) she tries to recover some unknown sensitive data by using new observations and the information collected during the profiling phase.

More formally, let us assume that the attacker retrieves a set of observations of the random variable z, where each observation has the form $\boldsymbol{\ell} = \varphi(z) + B$, where φ is an unknown function and B a Gaussian noise with standard deviation σ. She can estimate the expectation $\boldsymbol{\mu}_0$ and covariance Σ_0 of $\boldsymbol{\ell}$ when $z = 0$ and $\boldsymbol{\mu}_1, \Sigma_1$ when $z \neq 0$.

For a Gaussian distribution of expectation μ and covariance matrix Σ, the probability density function (pdf) of $\boldsymbol{\ell} \in \mathbb{R}^t$ is defined as:

$$f(\boldsymbol{\ell}) = \frac{1}{\sqrt{(2\pi)^t \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{\ell} - \boldsymbol{\mu})' \cdot \Sigma^{-1} \cdot (\boldsymbol{\ell} - \boldsymbol{\mu})\right) . \tag{9}$$

So by evaluating $f_{l|z=0}$ and $f_{l|z\neq 0}$ she obtains the likelihood that $z = 0$ was the manipulated value.

Experiments. We have tested the template attack against a real device and we provide in this section the results of our experiments.

The target of our experiment is an ATMega328P device. We have by-passed the decoupling capacitors of the device which may filter out useful signals. We have then connected the oscilloscope to the device and measured the difference of potential at the ends of a resistor placed between the ground pin of the ATMega328P and the ground of the device. We have finally pre-filtered the input of the oscilloscope at 20 Mhz and sampled the data at 100 Mhz. Our settings allow to obtain small curves while keeping as much information as possible.

We have then collected 250,000 leakages where we controlled the value of the mask. A random bit was used to select if $z = 0$ or $z \neq 0$ was used by the implementation. Knowledge of the random bit allowed us to split the set of acquisition between those with $z = 0$ and those with $z \neq 0$ to build templates. We also used this knowledge to verify the confidence of the likelihood distinguisher.

We show in Fig. 6 the differences between the expectation of the two sets. It is possible to distinguish important differences between them, in particular around points 250 and 300.

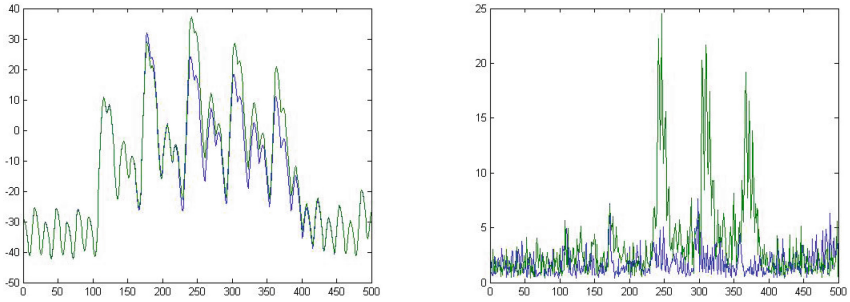


Fig. 6. Expectation and variance of real leakages when $z = 0$ and $z \neq 0$.

During attack phase we have acquired 250,000 more curves and tried to separate them into two sets. The knowledge of the value of the random bit allowed us to verify the success rate of the distinguisher. We have used 80 points to estimate the pdfs of Eq. 9. These points were chosen as those providing the highest variance between the expectations of the two sets. In this settings we obtained 99.84 % of correct detection rate.

Our attack thus demonstrates that even for real devices it is possible to break the countermeasure by using only first and second order statistical attacks.

6 Possible Fix-Ups and Residual Issues

As shown in Sects. 4 and 5 the differences between the distributions of the Hamming weight of masked values for different inputs can be exploited by an attacker to recover manipulated secrets. In this section we propose a method to improve the resistance of the scheme to the attacks that we have presented in this work while preserving the fault detection capability.

6.1 Conservative Shuffling

In this section we suggest how to add algorithmic noise to the ODSM scheme in order to defeat the attack introduced previously. We show that our countermeasure preserves the fault detection capability and the possibility to perform computation within the masking scheme.

Our method relies on shuffling the generating matrices G and H of the code $\mathcal{C}$ and $\mathcal{D}$, losing the systematic form of $\mathcal{C}$'s generating matrix. Successively applying permutations on the columns of these matrices allows to randomize the mappings between elements of $\mathbb{F}_2^k$ and $\mathcal{C}$ (resp. $\mathbb{F}_2^{n-k}$ and $\mathcal{D}$) by randomizing the codewords of $\mathcal{C}$ (resp. $\mathcal{D}$) itself. We can note that the properties of the associated codes remain unchanged as we only reorder the columns of the generating matrices. In particular the duality between $\mathcal{C}$ and $\mathcal{D}$ is preserved since we apply the same permutation on both matrices G and H. This can be seen by recalling that any permutation of n columns of a $k \times n$ matrix can be realized by multiplying from the right this matrix by a permutation matrix P. Further recalling that P is orthogonal ($PP^T = \mathbb{I}$), it comes straightforwardly that:

$$GP(HP)^T = GP(P^T H^T) = G(PP^T)H^T = GH^T = 0 \tag{10}$$

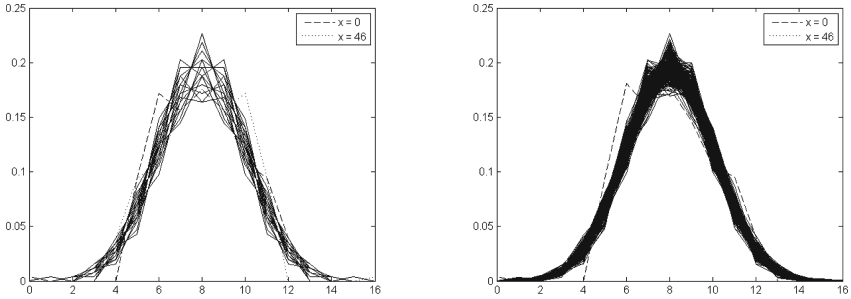
Such a process can be easily achieved at the cost of up to 12 bits of random used to select two columns to permute and an amount of circular shift. For the XOR operation, the permutation can be straightforwardly applied to the encoding of the operand: $z' = z \oplus k_i GP$. For the linear operation, the permutation needs to be reflected on the L' matrix of (6):

$$\begin{aligned} L'' &= (GP)^T (GP(GP)^T)^{-1} LGP \oplus (HP)^T (HP(HP)^T)^{-1} HP && (11)\\ &= P^T (G^T (GG^T)^{-1} LG)P \oplus P^T (H^T (HH^T)^{-1} H)P && (12)\\ &= P^T (G^T (GG^T)^{-1} LG \oplus H^T (HH^T)^{-1} H)P && (13)\\ &= P^T L' P && (14) \end{aligned}$$

For the non-linear operation, only the table recomputation approach seems to be achievable with reasonable overhead as the look-up table approach would require to recompute the S' table for all z. Algorithm 1 should then be adapted as exposed in Algorithm 3.

Algorithm 3. Masked SubBytes transformation on $z = xGP \oplus yHP$

$z = z \oplus x'GP$;
$x = z(GP)^T (GG^T)^{-1}$;
$x = S'_{recomp}(x)$;
$z' = xGP \oplus yHP$;
$z' = z' \oplus x''GP$;
return z';



(a) Original ODSM pdf. (b) Shuffled ODSM pdf.

Fig. 7. Hamming weights' pdf of encoded values for ODSM Vs Shuffled ODSM.

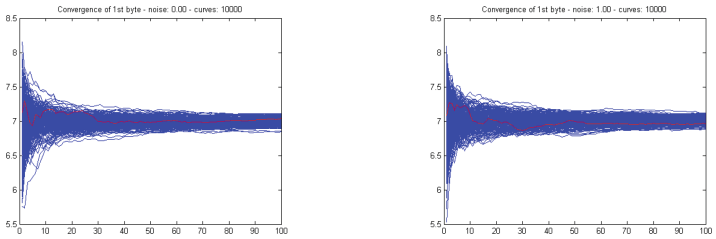


Fig. 8. Result of the attack with $\sigma = 0$ and $\sigma = 1$ for $10,000$ leakages.

Using such shuffled matrices for each encryption, we obtain the distribution depicted in Fig. 7b. Figure 7a is recalled for comparison purpose.

We can see that the distribution gives a much less explicit hint on the manipulated value compared to the original one. However we can still observe differences in the distributions for each value, supporting a residual weakness. Nevertheless, the attack described in Sect. 4 now fails even with a low noise level as can be seen in Fig. 8.

6.2 Residual Issue: Encoding 0

From the observation of Figs. 7a and b we can see that one potential weakness of the original scheme is not taken care of by our method. Indeed, we observe that the value $xG \oplus yH = 0\text{x}0000$ can only be obtained when $x = 0x00$.

By assuming that the attacker can detect the manipulation of the value 0x0000 then she directly knows the corresponding value 0x00 of the internal AES state. Such a weakness is not present in traditional boolean masking, where all masked values can be produced by all secret values.

Such an attack may not be merely theoretical since the hypothesis of retrieving the Hamming weight of internal values of more than one byte by SPA has

been exploited in recent publications [12,13] in order to retrieve the operands of a 128-bit scalar multiplication.

We further remark that a similar SPA weakness would affect the ODSM scheme for any choice of code. Indeed, since the codes $\mathcal{C}$ and $\mathcal{D}$ are complementary duals and from the definition of $\mathcal{C}$ and $\mathcal{D}$ we know that:

$$\forall\ z\ \in\ \mathbb{F}_2^n,\ \exists!\ (x,y)\ \in\ \mathbb{F}_2^k \times \mathbb{F}_2^{n-k} \text{ such that } z = xG \oplus yH$$

This holds in particular when $z = 0$, which is thus equivalent to $xG = yH$ and to $x = y = 0$. Consequently even when randomizing G and H, we can only observe a value of null Hamming weight when the sensitive value x *and* the mask y are null. Unfortunately, as we have shown in Sect. 5 such weakness may be exploited by using template attacks. We can nevertheless stress that in case the attacker cannot control the value of the mask she may not be able to build the templates and consequently the attack should not work.

7 Conclusion

The definition of new countermeasures tackling in the same effort side-channel analysis and fault attacks is definitely a challenging task. The ODSM scheme succeeds in providing both a way to detect errors and ensuring the independency of the mean and the variance of the Hamming weight of masked data. However in this work we demonstrate that the distributions of Hamming weights of the ODSM encoded data are actually dependent on the sensitive values being manipulated, which renders the scheme helpless against a side-channel attack considering only 1^{st}-order statistical moment of the observed leakage. Furthermore we have shown that some measures can be taken in order to reduce the leakage exposed when observing the Hamming weight distributions for a given sensitive value, although it turns out that the scheme cannot be made totally SCA-resistant. Still, countermeasures based on coding theory appear as promising candidates to improve the resistance of cryptographic implementations against both side-channel and fault attacks. In particular, the definition of methods allowing to perform the complete execution of an algorithm under the protection of the code is an interesting line of research for future works.

References

1. Coron, J.-S., Goubin, L.: On boolean and arithmetic masking against differential power analysis. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 231–237. Springer, Heidelberg (2000)
2. Bringer, J., Chabanne, H., Le, T.H.: Protecting aes against side-channel analysis using wire-tap codes. J. Cryptographic Eng. **2**, 129–141 (2012)
3. Bringer, J., Carlet, C., Chabanne, H., Guilley, S., Maghrebi, H.: Orthogonal direct sum masking. In: Naccache, D., Sauveron, D. (eds.) WISTP 2014. LNCS, vol. 8501, pp. 40–56. Springer, Heidelberg (2014)

4. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
5. Kocher, P., Jaffe, J., Jun, B.: Introduction to differential power analysis and related attacks. Technical report, Cryptography Research Inc. (1998)
6. Messerges, T.: Poweranalysis attacks and countermeasures for cryptographic algorithms. Ph.D. thesis, University of Illinois (2000)
7. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power analysis attacks of modular exponentiation in smartcards. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 144–157. Springer, Heidelberg (1999)
8. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
9. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
10. Moradi, A.: Wire-tap codes as side-channel countermeasure - an FPGA-based experiment. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 341–359. Springer, Heidelberg (2014)
11. Azzi, S., Barras, B., Christofi, M., Vigilant, D.: Using linear codes as a fault countermeasure for nonlinear operations: application to AES and formal verification. In: PROOFS: Security Proofs for Embedded Systems (2015)
12. Belaïd, S., Fouque, P.-A., Gérard, B.: Side-channel analysis of multiplications in GF(2^{128}). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 306–325. Springer, Heidelberg (2014)
13. Belad, S., Coron, J.S., Fouque, P.A., Grard, B., Kammerer, J.G., Prouff, E.: Improved side-channel analysis of finite-field multiplication. Cryptology ePrint Archive, Report 2015/542 (2015). http://eprint.iacr.org/