J Cryptogr Eng (2012) 2:129–141
DOI 10.1007/s13389-012-0034-2

REGULAR PAPER

Protecting AES against side-channel analysis using wire-tap codes

Julien Bringer · Hervé Chabanne · Thanh Ha Le

Received: 7 October 2011 / Accepted: 30 May 2012 / Published online: 24 June 2012

Abstract We introduce a general protection of data against side channel analysis (SCA) based on wire-tap codes. We focus in this paper on an application for the AES cipher. We analyse the behaviour of our countermeasure against different kinds of SCA. Our results show that this protection is an excellent alternative to classical masking methods as it comes with the secrecy property of wire-tap coding, practical resistance against first and second-order DPA. Moreover, we point out that it brings two novel features: the possibility to unmask without the knowledge of the mask and its capability to detect some faults.

Keywords Side channel analysis · Wire-tap codes · AES cipher

1 Introduction

Side-channel analysis (SCA) has been introduced to recover secret data that are inputs of a cryptographic algorithm. SCA exploits the physical characteristics of the physical token where the algorithm is implemented. Namely, it exploits its leakage (timing, power consumption, electromagnetic emanations) made during the execution of this algorithm. Statistical treatments are then performed to extract secret data from all the captures.

One-time pad is proven information-theoretically secure. It simply consists in adding to a plaintext a random value. This kind of encryption can be seen as a basis for masking methods used in order to prevent SCA [3,6]. On one hand, the simplicity of the needed operations is certainly an asset for such a protection. Typically, for many block ciphers, this corresponds to the so-called boolean masking which simply consists in an exclusive-or performed between the input of the cipher and a random value, here called mask. This led in recent years to a lot of papers devoted to the implementations as [1,20].

On the other hand, these masking methods leak information when an adversary has access to several intermediate leakage values obtained from the encryption of a message under the same key. This gives birth to sophisticated attacks as high-order side-channel analysis. For instance, different high-order differential power analysis (DPA) methods have been discussed in [3,8,12,19]. The mutual information analysis (MIA) in high-order analysis is also formalized in [5,18].

To enhance security against higher-order DPA, a few masking techniques are suggested. In particular, it is possible to use several random additive masks at the same time [20] to ensure provable security against higher-order DPA (d masks for security against $(d-1)$th-order DPA). Affine masking is another solution that was first discussed in [24] and recently further studied in [4] that combines additive masking and multiplicative masking (either as a linear multiplication by a matrix or as a field multiplication). Compared with multiple padding, [4] shows that the affine masking is provably secure only at first order DPA but that it can achieve higher order resistance against DPA in practice (i.e. an important number

This work has been partially funded by the JST/ANR SPACES (Security evaluation of Physically Attacked Cryptoprocessors in Embedded Systems) project.

J. Bringer · H. Chabanne · T. H. Le
Morpho, Issy-Les-Moulineaux, France
e-mail: julien.bringer@morpho.com

T. H. Le
e-mail: thanh-ha.le@morpho.com

H. Chabanne (✉)
Télécom ParisTech, Paris, France
e-mail: chabanne@telecom-paristech.fr;
herve.chabanne@morpho.com

of consumption traces is needed). Affine masking is based on a mapping $x \in \{0,1\}^k \mapsto x.L \oplus m$ where m is a random element in $\{0,1\}^k$ and L is either a random linear bijection in $\{0,1\}^k$ [24] or a random element in $GF(2^k)$ [4].

Note that many other masking methods, not based directly on additive or multiplicative masking, have been designed for block ciphers implementations, in particular for the AES, see for instance [1,2,9,14,17]. Nonetheless, improvements or new methods are still requested to increase the resistance against more and more advanced high-order attacks.

In this paper, we consider another secrecy system, namely the wire-tap channel. [25] has shown a way to convey information confidentially whenever a receiver enjoys a better channel than its adversary does. Indeed, thanks to coding theory, it is proved that when an adversary has access to less than a certain amount of the transmitted data, he has no clue on what the original information was (cf. Sect. 2). We here show how this coding problem can be brought to SCA countermeasures. One has to imagine the wire-tapping opponent in this context as the eavesdropper making its leakage measurements. Wire-tap security thus ensures a natural resistance against different captures of the same computation.

We introduce a new masked version of the AES following to this principle of wire-tap channel. Our masking method is based on the mapping $x \in \{0,1\}^k \mapsto x.L \oplus c \ \in \{0,1\}^n$ where c is a random codeword from a well-chosen code of length n and L is a linear transformation from $\{0,1\}^k$ into a subspace of $\{0,1\}^n$. This method can be interpreted as an improvement of the affine masking to the wire-tap channel context. Our proposal has the property that the length of the message is expanded during the protection. Due to wire-tap secrecy this gives us a secure simple power analysis resistant masking, and we show that it also achieves first and second-order resistance in practice. As a side effect of the presence of error correcting codewords, we can efficiently detect the presence of some faults during an execution of the algorithm.

Our protection technique has the unique property that one can unmask without the need to use the mask itself. In [23], the authors show how to mount an High Order DPA attack on the AES using at the same time both the masked value and its mask. With respect to this result, we can stress the importance of this asset of our proposal. To the best of our knowledge, our countermeasure is the sole masking solution with this feature.

Finally, we also confront our method of protection of AES to different scenarios of side channel analysis.

2 Wire-tap codes

2.1 Wire-tap principle

The wire-tap channel problem was first introduced by Wyner [25] in 1975 and later extended in [16]. The context is a transmission channel (noisy or not) between two parties Alice and Bob where an eavesdropper Eve can only see a degraded version of the transmitted messages. Assume that the channel between Alice and Bob is noiseless, then it means that the transmission channel from Alice to Eve (or from Bob to Eve) is either with noise or with erasures. We consider here the second case, i.e. the erasure wire-tap channel [16]. This is illustrated by Fig. 1 where BEC(ϵ) stands for Binary Erasure Channel with a probability of ϵ to have an erasure.

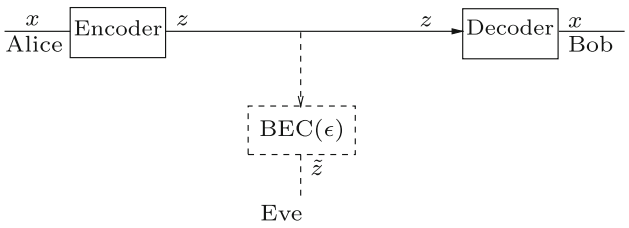

Fig. 1 Erasure wire-tap channel

The principle to encode a secret message in the context of an erasure wire-tap channel is the following. To transmit k-bit messages, a binary linear code C of length n is chosen and each message is associated to a coset of C in such a way that it is not possible for Eve to gain information on the transmitted message when observing at most $\mu < n$ bits of the encoded message. More precisely, let

- G be a generator matrix of size $(n-k) \times n$ of C, with rows $(g_1, \ldots, g_{n-k})$,
- $L = (l_1, \ldots, l_k)$ be a family of k linearly independent vectors from $\{0,1\}^n$, not in C,
- $m = (m_1, \ldots, m_{n-k})$ be a uniformly random vector of $(n-k)$ bits,

then a message $x = (x_1, \ldots, x_k)$ in $\{0,1\}^k$ is encoded as

$$z = x.L \oplus mG \tag{1}$$

where $x.L = x_1 l_1 \oplus \cdots \oplus x_k l_k \in \{0,1\}^n$ and $c = mG$ is the random codeword related to m ($c = m_1 g_1 \oplus \cdots \oplus m_{n-k} g_{n-k}$). Here L and G are in fact chosen such that $(g_1, \ldots, g_{n-k}, l_1, \ldots, l_k)$ span the entire space vector $\{0,1\}^n$.

Let H be the parity matrix of C, i.e. the $k \times n$ matrix such that for all $c \in C$, $Hc^T = 0$.

2.2 Wire-tap secrecy

In this construction, the information rate of the channel is $R = k/n$ (1 minus the information rate of C). Tolerating an information rate below 1 enables to resist to the leakage of some bits to an eavesdropper.

Indeed, consider the received message $\tilde{z}$ by Eve. Eve has to search which coset of C corresponds to $\tilde{z}$. A coset will correspond to $\tilde{z}$ if it contains at least one vector that is equal to $\tilde{z} \in \{0,1\}^n$ in the unerased positions. The maximum value

for the total number of cosets of C that correspond to $\tilde{z}$ is 2^k in this construction. When this is reached for any input message, this means that the conditional entropy $\mathrm{H}(X|\tilde{Z} = \tilde{z})$ is equal to the entropy of X, i.e. k. This leads to perfect secrecy. A way to ensure this condition is given by the following result.

Lemma 1 [15,22] *Let C be a linear code with a parity matrix $H = (h_1 \cdots h_n)$ of size $k \times n$ where h_i is the i-th column of H. Consider an eavesdropper's observation $\tilde{z} \in \{0, 1, ?\}^n$ with $n - \mu$ erased positions given by $\{i \text{ s.t. } z_i = ?\} = \{i_1, i_2, \ldots, i_{n-\mu}\}$. Then the original message x remains perfectly secret if and only if the sub-matrix $\tilde{H} = (h_{i_1} \cdots h_{i_{n-\mu}})$ is of rank k.*

Therefore, to achieve perfect secrecy in the above construction against eavesdropping of any μ bits of the transmitted message, it is necessary and sufficient that the parity matrix H of the code C satisfies that all of its $k \times (n - \mu)$ sub-matrices have rank k. It is known [10] that this is equivalent to having a minimum distance equal to $\mu + 1$ for the code generated by H, i.e. for the dual code of C. This implies that $\mu \leq n - k$ and the optimal choice for given parameters n, k is to take for C the dual code of an $[n, k, d]$ binary linear code with the greatest possible minimum distance d.

Various constructions for efficient wire-tap channels are studied under the constraints of high information rate, encoding/decoding performances and approaching secrecy capacity (see for instance [22]). In our case, we will consider small values for n and k so we will rely on more classical code constructions.

3 Protecting AES with wire-tap codes

In the following, we explain—based on the application to AES—how to protect a block cipher algorithm against side-channel analysis by embedding wire-tap codes.

3.1 Brief description of AES

AES [13] takes as input a 16-byte block and consists mainly in the iteration of a round. The 16-byte block is represented as a 4×4 square called a *state* and is subject to the following operations:

- `AddRoundKey`: this operation adds a round key to the state by a bitwise XOR operation;
- `SubBytes`: the non-linear byte substitution operates independently on each byte (each byte is replaced with another according to a lookup table);
- `ShiftRows`: it is a permutation on the 16 bytes where each row of the state is shifted cyclically a certain number of times;
- `MixColumns`: this operation treats each column of the state as a 4-byte vector and multiplies it by a matrix.

The high-level description of the AES-128 algorithm is given below:

1. Key Expansion: round keys $K_0, \ldots, K_{10}$ are derived from the cipher key using Rijndael's key schedule,
2. `AddRoundKey`(K_0)
3. for i from 1 to 9:
   (a) `SubBytes`;
   (b) `ShiftRows`;
   (c) `MixColumns`;
   (d) `AddRoundKey`(K_i).
4. `SubBytes`;
5. `ShiftRows`;
6. `AddRoundKey`(K_{10}).

3.2 Our proposal

3.2.1 Notations

Let k be the number of bits in the secret data handled in a cryptographic algorithm. The objective is to encode the k data bits into $n > k$ bits, such that an adversary who observes a bit subset of size $\mu < n$ can gain no information on the secret data. We choose a code C with a generator matrix G and a parity matrix H, a set $L = (l_1, \ldots, l_k)$ of k linearly independent vectors from $\{0, 1\}^n \backslash C$ and we encode a vector $x = (x_1, \ldots, x_k) \in \{0, 1\}^k$ as explained in Sect. 2:

$$z = x.L \oplus c \tag{2}$$

where $x.L = \sum_{i=1}^{k} x_i l_i$ (sum modulo 2) and $c = m.G$ is a random codeword.

The code C is constructed in advance by selecting its parity matrix H as the generator matrix of an $[n, k, \mu + 1]$ binary linear code and L can be computed later. The k linearly independent n-bit vectors $l_1, \ldots, l_k$ can be fixed for each target component. For example for smart cards, the set $L = (l_1, \ldots, l_k)$ can be separately computed for each card during the personalization step.

Remark 1 In the following, we will consider $k = 8$, i.e. that the wire-tap masking is applied at the byte level (fitting to input's size for AES S-boxes).

When transforming the encryption algorithm to operate on masked data, the main overhead comes from the definition of new S-boxes compatible with such wire-tap representation. As for classical additive masking, for efficiency reasons we may prefer to use the same mask for all bytes (and all rounds). We consider this option in the following description of the masked algorithm.

Nevertheless, if memory size is not a concern we can easily use different codewords for each byte and each round (cf. Sect. 3.2.4).

3.2.2 Generation of L

We give below a short description of an algorithm to randomly generate L.

In order to compute the set $L = (l_1, \ldots, l_k)$ of k linearly independent n-bit vectors which are not in C, we can proceed as follows:

- Take randomly an n-bit vector l_{test} then verify if the vector is in the code C by applying the parity matrix H (e.g. by computing the product Hl_{test}^T).
  - If the product Hl_{test}^T is zero, which means that the vector l_{test} is in the linear code C, then restart by taking another random vector.
  - If the product Hl_{test}^T is different from zero, define $l_1 = l_{\text{test}}$ and continue to generate l_2 until l_k.
- After each iteration, a Gaussian elimination algorithm can be used to determine if the generated vectors are independent. For instance, when constructing l_i, if $l_1, \ldots, l_{i-1}$, l_{test} are correlated then iterate again with another l_{test} to find l_i.

For our setting, this generic algorithm is efficient enough as n and k remain quite small.

The set L can also be regenerated from time to time after some number of utilizations.

3.2.3 Pre-configuration

In the context of the wire-tap code masking, we first define new operations $\texttt{AddRoundKey}'$, $\texttt{ShiftRows}'$ and $\texttt{MixColumns}'$ from the original linear operations

$\texttt{AddRoundKey}$, $\texttt{ShiftRows}$ and $\texttt{MixColumns}$

such that they are compatible with the wire-tap representation $x.L \oplus c$.

$\texttt{AddRoundKey}'$ has to expand the key by wire-tap encoding before the exclusive-OR operation. $\texttt{ShiftRows}'$ and $\texttt{MixColumns}'$ are modified such that they operate linearly on a larger state. More details are given in Sect. 3.2.5.

3.2.4 Preliminary step

The preliminary step below is computed before encryption. Let c, c' be two random codewords from the code C, then we define a new S-box $\texttt{SubBytes}'$ as:

$$\texttt{SubBytes}'(x.L \oplus c) = \texttt{SubBytes}(x).L \oplus c'$$

The look-up table associated to $\texttt{SubBytes}'$ is then defined over $\{0, 1\}^n$. This leads in general to a size of $2^n \times n$ bits. Here, as only 2^k entries are possible, we can manage to store it on $2^k \times n \times 2$ by representing it as a list of 2^k couples of n-bits input/output vectors; the factor of expansion becomes $2n/k$. For efficiency reasons, we use the same codewords, i.e. the same $\texttt{SubBytes}'$, for all rounds of the algorithm.

Nevertheless, independent choices could be envisaged if the memory size is sufficient. This requires at most $2^k \times n \times 320$ bits. Using a different codeword for each byte and each round ensures in principle higher security as wire-tapping by an adversary would need in that case to success in a shorter time frame. As for classical masking, the trade-off between using one transformed S-box and at most 160 different transformed S-boxes has to be tuned depending on the context.

By default, new codewords are drawn before each encryption. Note however that due to the secrecy property of the wire-tap codes, you could use several times the same codewords if you can ensure that the overall number of leaked bits remains lower than the security bound μ.

Remark 2 For instance, with $n = 12$ (respectively $n = 16$), the required memory for one transformed S-box is 768 bytes (resp. 1,024 bytes); this corresponds to an expansion factor of 3 (resp. 4).

3.2.5 Transformation of the encryption algorithm

We explain below the new version of the algorithm when protected by wire-tap encoding. This corresponds to the operations that are executed at each encryption. Let x the input message to be encrypted.

We have the following steps (to simplify the description of the masking we consider only one byte for x, whereas of course the whole state is to be masked):

- Take c and c' at random and define $\texttt{SubBytes}'$ as above.
- Draw another random codeword c_1 and let $c_2 \leftarrow c \oplus c_1$.
- Compute $z \leftarrow x.L \oplus c_1$
- Compute $z \leftarrow \texttt{AddRoundKey}'(K_0)(z) \oplus c_2$
  (i.e. $z = \texttt{AddRoundKey}'(K_0)(x.L \oplus c_1) \oplus c_2 = (K_0 \oplus x).L \oplus c$)
- for i from 1 to 9
  - $z \leftarrow \texttt{SubBytes}'(z)$
    (this gives $z = \texttt{SubBytes}(K_0 \oplus x).L \oplus c$')
  - $z \leftarrow \texttt{ShiftRows}'(z)$
    (it leads to $z = \texttt{ShiftRows} \circ \texttt{SubBytes}(K_0 \oplus x).L \oplus \texttt{ShiftRows}'(c')$)
  - $z \leftarrow \texttt{MixColumns}'(z)$
    ($z = \texttt{MixColumns} \circ \texttt{ShiftRows} \circ \texttt{SubBytes}(K_0 \oplus x).L \oplus \texttt{MixColumns}' \circ \texttt{ShiftRows}'(c')$)
  - $z \leftarrow \texttt{AddRoundKey}'(K_i)(z) \oplus c_2 = z \oplus K_i.L \oplus c_2$
  - $z \leftarrow z \oplus c_1 \oplus \texttt{MixColumns}' \circ \texttt{ShiftRows}'(c')$
    i.e. $z = (K_i \oplus \texttt{MixColumns} \circ \texttt{ShiftRows} \circ \texttt{SubBytes}(K_0 \oplus x)).L \oplus c$

- $\texttt{SubBytes}'$
- $\texttt{ShiftRows}'$
- $\texttt{AddRoundKey}'(K_{10}) \oplus c_3 \oplus \texttt{ShiftRows}'(c')$ with some random codeword c_3
  (this leads to $y.L \oplus c_3$ where y is the output value of the encryption via a non-masked AES implementation)
- Apply the parity matrix H and invert $H(y.L)^T$ to obtain the final result y (see Sect. 5 for an efficient solution)

Note that, above, the input message, the intermediate messages and the round keys as well are always masked by some codewords when manipulated.

$\texttt{AddRoundKey}'$, $\texttt{ShiftRows}'$ and $\texttt{MixColumns}'$ are defined in the following way to be compatible with the wire-tap representation (cf. Sect. 3.2.3):

- $\texttt{AddRoundKey}'(K_i)(x.L) = \big(\texttt{AddRoundKey}(K_i)(x)\big).L = (x \oplus K_i).L$, for all $x \in \{0,1\}^8$, for all $i \in \{0,\ldots,10\}$;
- $\texttt{ShiftRows}'(x^4.L^4) = \texttt{ShiftRows}(x^4).L^4$ for all $x^4 = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}) \in \{0,1\}^{4\times 8}$;
- $\texttt{MixColumns}'(x^4.L^4) = \texttt{MixColumns}(x^4).L^4$, for all $x^4 \in \{0,1\}^{4\times 8}$.

where $\texttt{AddRoundKey}'(K_i)$ is basically the XOR between its input and $K_i.L$ and $\texttt{ShiftRows}'$, $\texttt{MixColumns}'$ are defined to be linear operations over 4 n-bit words ($\texttt{ShiftRows}$ operates row per row of the AES state and $\texttt{MixColumns}$ column per column of the AES state, the state being a 4×4 bytes array). This representation increases at most the size by a factor n/k as the expansion is linear. Above, $L^4 = (L^{(1)}, L^{(2)}, L^{(3)}, L^{(4)})$ corresponds to the concatenation of four vectors of $k = 8$ linearly independent n-bit vectors (either four different or four times the same L) and $x^4.L^4$ is $x^{(1)}.L^{(1)} + x^{(2)}.L^{(2)} + x^{(3)}.L^{(3)} + x^{(4)}.L^{(4)}$.

These calculations are easy to perform because the operations are linear and done only once after the generation of L (that is set for once or renewed after some number of uses; and thus inexpensive in performance with respect to execution of an encryption). They can be done by both software and hardware implementations.

4 Properties and extensions

4.1 Simple power analysis

The consumption of the new S-boxes $\texttt{SubBytes}'$ now depends on inputs/outputs encoded via the wire-tap codes. Thus by definition, if an adversary recovers less than μ bits through one or several observations of the encryption through the same choices of codewords c and c', then by Lemma 1 he cannot gain information on the original inputs/outputs of $\texttt{SubBytes}$. Moreover, as the length n (and consequently μ) can be chosen large at the time of the set-up of the implementation, it is theoretically possible—assuming[1] that the number of bits that can be read out by an adversary is bounded by a constant independent of n—to thwart side-channel analysis with large number of leaked bits.

4.2 First order DPA

Similarly to classical additive masking, the resistance against Differential Power Analysis at the first order is achieved thanks to the masking of the values $x.L$. The mask is (or derived from) a random codeword which is not a random vector from $\{0,1\}^n$. But as illustrated by our experiments in Sect. 6, this difference affects the security only when the size $n-k$ of the code C (thus the number of possible codewords) is too small. When $n-k$ increases, the difficulty increases as well and this enables to obtain parameters which achieve resistance against first order DPA that is almost the same as the resistance obtained with classical additive masking technique.

4.3 Higher order DPA

In our construction, the length of the encoded bytes can be increased. This is an important difference with classical masking technique and this is an interesting feature when studying second-order DPA. The results in Sect. 6 confirm that the efficiency of the second-order side-channel attack decreases when the size n increases.

Moreover, we illustrate in Sect. 6.4 another way to achieve implementations that are resistant to second order DPA in practice by keeping the vector L unknown. This is somehow close to affine masking techniques from [4,24]: we show in Sect. 6.4 that the specific structure of wire-tap codes—while enabling new properties that are discussed further below—still enables us to achieve in practice second-order resistance.

4.4 Unmasking without masks

An additional feature that comes with our technique is the possibility to unmask without the knowledge of the final codeword used. This is due to the parity matrix H that cancels out all codewords. It avoids to manipulate the mask at the last step of the algorithm that is usually a phase where an

[1] This assumption may seem unusual at first glance. However, note that for side-channel analysis we usually measure consumption traces that depend on the Hamming weight of the variables. This is in fact equivalent to the knowledge of one bit.

adversary could try to capture the mask when retrieved from the memory.

A usual scenario for a DPA attack is when the adversary knows the ciphertexts and wants to attack the last rounds. For instance, one can imagine a classical masking scheme. In this case, [23] considers two relevant points in time which occur during a small period of time and which are dependent on the masked value and its mask:

- The adversary measures the effect of the masked operations during the last rounds of the encryption.
- At the end of the AES encryption, the adversary measures the consumption when the random mask is retrieved to unmask the result.

To mount its HO-DPA attack, the adversary has to combine the elements coming from these two points in time.

In this context, the possibility to unmask without the use of the mask is clearly an asset.

To increase the difficulty, we can moreover choose to precompute the masking version of the final round key ($K_{10}.L \oplus c_3 \oplus \texttt{ShiftRows}'(c')$) at some random place during the encryption.

For specific choices of C and L, this possibility can be generalized to all unmasking steps by using the parity matrix H_1 (respectively H_2) of the code

$$\texttt{MixColumns}'(\texttt{ShiftRows}'(C))$$

(resp. $\texttt{ShiftRows}'(C)$), and choosing L not in these codes either. See Sect. 5 for an explanation on the way to unmask directly with a parity matrix.

5 Practical construction

To achieve efficient encoding/decoding phases, we can choose H in a systematic form, i.e. $H = \left(I_k | M\right)$ where I_k is the identity matrix of size $k \times k$ and M a matrix of size $k \times (n-k)$. This implies that the generator matrix G can be written as $G = \left(-M^T | I_{n-k}\right)$. Now define for $i \in \{1, \ldots, k\}$, l_i the vector with all zeros but a 1 at position i, then an encoded message of x is $z = (x_1, \ldots, x_k, 0, \ldots, 0) \oplus c$ for some random codeword c. To recover x, one then simply applies H to obtain:

$$\begin{aligned} Hz^T &= H(x_1, \ldots, x_k, 0, \ldots, 0)^T \oplus Hc^T \\ &= I_k.(x_1, \ldots, x_k)^T \oplus M.(0, \ldots, 0)^T \oplus 0 \\ &= (x_1, \ldots, x_k)^T \end{aligned}$$

This can be seen as additive masking with some additional algorithmic noise to enable syndrome decoding through the parity check matrix. This is one option studied in our evaluations in Sect. 6. We also analyse the situation where L is chosen more randomly in order to mix the bits of x and thus to hide its Hamming weight.

In practice, the possible values for the choice of M are directly related to the existing $[n, k = 8, \mu + 1]$ binary linear codes as discussed in Sect. 2. For instance, the highest possible μ for $n = 16$ is $\mu = 4$ as the highest minimum distance is obtained for a [16, 8, 5] code (cf. [7]). This can lead to $H = \left(I_8 | M\right)$ with:

$$M = \begin{pmatrix} 10001011 \\ 11000101 \\ 11100010 \\ 01110001 \\ 10111000 \\ 01011100 \\ 00101110 \\ 00010111 \end{pmatrix}$$

For higher value of μ, the length n can be increased: for instance we can use an [25, 8, 9] code to tolerate up to $\mu = 8$ leaked bits.

5.1 Unmasking

This is a good construction to illustrate how to unmask (or to transform a masked value into another masked one) directly at an intermediate step without using the original mask thanks to the parity matrix properties. Assume that you have

$$z = x \oplus \texttt{MixColumns}'(\texttt{ShiftRows}'(c'))$$

and that you want to obtain $x \oplus c$ without making use of the mask c'. In this setting, you can proceed as follows:

1. $z \leftarrow z \oplus (c_1, \ldots, c_k, 0, \ldots, 0)$
2. Compute $z \leftarrow H_1 z^T$ to obtain $z = x \oplus (c_1, \ldots, c_k)$
3. Output $(z_1, \ldots, z_k, c_{k+1}, \ldots, c_n)$

This doing a given mask can be manipulated only to mask, and not at the unmasking step: this reduces at the minimum the risk of an adversary eavesdropping the mask.

5.2 Fault detection

Finally it is possible to use the wire-tap encoding of a message to add a kind of fault detection step by exploiting the fact that an encoded message x has a special format ($z = x.L \oplus c$): Apply H to z^T to obtain $H(x.L)^T$, invert the result to recover x, compute $z' = x.L \oplus c$ and check whether $z = z'$.

For H chosen in a systematic form and

$$x.L = (x_1, \ldots, x_k, 0, \ldots, 0)$$

as explained above, some specific faults can be detected: if there are some bit flips in the $n-k$ last positions of z, i.e. $z = (x_1, \ldots, x_k, y_1, \ldots, y_{n-k}) \oplus c$ for some vector $y = (y_1, \ldots, y_{n-k})$. The operation Hz^T will lead to $x^T \oplus My^T$ for which the check will fail if My^T is a non-zero vector.

Note that although the use of coding techniques and fault detection is not new, see for instance [11], the main idea here is to be able to combine masking and some detection capability on the same method.

6 Evaluation results

6.1 Power consumption models

Let $V_K = \{V_{K,i}\}_{i=1..N}$ denote the predictions of an intermediate value corresponding to messages m_i and a key hypothesis K; $W = \{W_i\}_{i=1..N}$ denote the values of N side-channel signals at the instant t where the intermediate value is manipulated. We define $\varphi(V_K)$ a selection function. In this analysis $\varphi(V_K)$ represents the Hamming weight of V_K, i.e. the selection function φ is the Hamming weight of the intermediate values.

In our evaluations, we observe the output of an AES Sbox. We also consider that the relation between the power consumption and the Hamming weight of the Sbox output is linear. The observations W_i corresponding to messages m_i are then simulated as: $W_i = \varphi(V_{K_0,i}) + B$ where K_0 is the correct key and B is a Gaussian noise whose variance is σ^2.

Masking is a common solution to protect the device against first-order analysis. The idea behind this technique is to mask intermediate values by random masks. Let consider the intermediate value $V_{K_0,i}$ corresponding to message m_i. With the masking technique, the device manipulates the masked data $V_{K_0,i} \oplus M_i$ instead of the direct value $V_{K_0,i}$. Let t_1 and t_2 denote the instants when the mask M_i and the masked data $V_{K_0,i} \oplus M_i$ are manipulated. The power consumption W_1 at t_1 and W_2 at t_2 are simulated as follows:

$$W_{1,i} = \varphi(M_i) + B_1$$
$$W_{2,i} = \varphi(V_{K_0,i} \oplus M_i) + B_2$$

where B_1 and B_2 are Gaussian noises whose variances are $\sigma_1^2 = \sigma_2^2 = \sigma^2$.

The second-order analysis aims at studying the dependency between three components: the observations W_1 at t_1 and W_2 at t_2 and the prediction $\varphi(V_K)$. In [12], Messerges focused on one bit and proposed to compute the correlation factor between the value of this bit and the absolute difference of two observations W_1 and W_2 (the abs-diff-DPA method). Chari et al. [3] suggested to calculate the correlation factor between $\varphi(V_K)$ and the product of two observations W_1 and W_2 (the product-DPA method). In [19], it was shown that by replacing the product W_1W_2 by the centered product $(W_1 - E(W_1))(W_2 - E(W_2))$, the product-DPA attack is significantly improved (the centered-product-DPA method).

Here, we have followed the strategies that are described above in the case of additive masking but by adapting them to the wire-tap code: we simulate the intermediate values as n-bit variables and we add additional Gaussian noise. We consider two instant t_1 and t_2 corresponding to the following leakages:

$$W_{1,i} = \varphi(c_i) + B_1$$
$$W_{2,i} = \varphi(x_{K_0,i}.L \oplus c_i) + B_2$$

where $x_{K_0,i}$ is the Sbox output corresponding to the correct key and the message m_i; codewords c_i is generated by the generator matrix G.

In our evaluation of the scheme, we use the Correlation Power Analysis (CPA) and Mutual Information Analysis (MIA) methods to evaluate the resistance of the proposed solution in first-order and second order analysis. In first-order analysis, only $W_{2,i}$ is considered while in second-order analysis both $W_{1,i}$ and $W_{2,i}$ are combined. For the second-order CPA, the centered-product method is used to optimize the analysis as shown in [19].

6.2 Evaluation index

In order to evaluate our results, we use the success rate of order 1 (or simply success rate) defined in [21] as the security metric. The success rate represents the probability that the correct key is sorted first by an adversary. In our evaluation, 1,000 repeated experiments were computed to calculate the success rate.

6.3 Matrix L is known

In this section, we suppose that the matrix L is known, it means that the attacker can compute the value of $x_{K,i}.L$ where $x_{K,i}$ is the prediction of the Sbox output corresponding to the key hypothesis K and the message m_i. The random codewords c_i are generated by the generator matrix G and unknown for attacker.

Two types of matrix L are considered. The first one has a simple form by choosing $L_{\text{simple}} = (l_1, \ldots, l_k)$ $(k = 8)$ as follows:

$$l_1 = [1, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0]$$
$$l_2 = [0, 1, 0, 0, 0, 0, 0, 0, 0, \ldots, 0]$$
$$l_3 = [0, 0, 1, 0, 0, 0, 0, 0, 0, \ldots, 0]$$
$$l_4 = [0, 0, 0, 1, 0, 0, 0, 0, 0, \ldots, 0]$$
$$l_5 = [0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0]$$
$$l_6 = [0, 0, 0, 0, 0, 1, 0, 0, 0, \ldots, 0]$$

$l_7 = [0, 0, 0, 0, 0, 0, 1, 0, 0, \ldots, 0]$

$l_8 = [0, 0, 0, 0, 0, 0, 0, 1, 0, \ldots, 0]$

The second type has a specific form which generates mixtures of the bits of x. As explained previously, the k linearly independent n-bit vectors $L = (l_1, \ldots, l_k)$ can be different from a card to another and randomly generated. According to our analysis, we observe that when L is known the resistance of the wire-tap based masking against first-order and second-order analysis does not really depend on the choice of the form of L. We obtain similar results with a simple-form matrix L and a specific-form matrix L. Therefore, in this section, we present only the results related to matrices L in the simple form. The evaluation results corresponding to specific-form matrices L are given in Appendix A.

In the following subsections, we evaluate the influence of the code C on the resistance of the wire-tap based masking. We distinguish two types of code: optimal codes that are defined as the dual of a code $[n, k, \mu + 1]$ that achieves the highest minimum distance at given length n and dimension k (i.e. codes optimal with respect to wire-tap security), and non-optimal codes for wire-tap security. In the sequel, the optimal codes used are the dual code of the best codes constructed thanks [7] of dimension $k = 8$ from length $n = 8$ to $n = 17$ ([9, 8, 2], [10, 8, 2], [11, 8, 2], [12, 8, 3], [13, 8, 4], [14, 8, 4], [15, 8, 4], [16, 8, 5], [17, 8, 6]).

All evaluations in this section are performed with three noise level: $\sigma = 0$ (noise-free condition), $\sigma = 1$ (weak-noise condition) and $\sigma = 3$ (middle-noise condition). The number of messages is fixed at 1,000 messages.

6.3.1 Optimal code

First-order analysis: Figure 2 represents the variation of the success rate of first-order analysis in function of n when considering simple-form matrices L and optimal codes. Two methods which are the CPA based on correlation factor and the MIA based on mutual information are evaluated. The results show that when $n \leq 12$, the success rate of CPA is low while the one of MIA is much higher. In fact, when n is close to k, the number of the codewords c, which is (2^{n-k}), becomes too small. The linear dependence between the side-channel signals and the Hamming weight of intermediate values is broken using optimal codes. However there still exists other dependences which can be revealed by mutual information analysis. This result shows the advantage of MIA compared to CPA when exploiting non-linear dependences between two variables.

When $n > 12$, the success rate of both CPA and MIA is close to zero and these methods do not allow to find out the correct key.

In order to verify the previous result, we compute the variation of the correlation factor and the mutual information between the Hamming weight of the prediction and side-channel signals which is given in Fig. 3. This figure shows a high mutual information between two variables when $n < 12$, which is due to a small number of codewords. It also confirms the success rate variation given in Fig. 2.

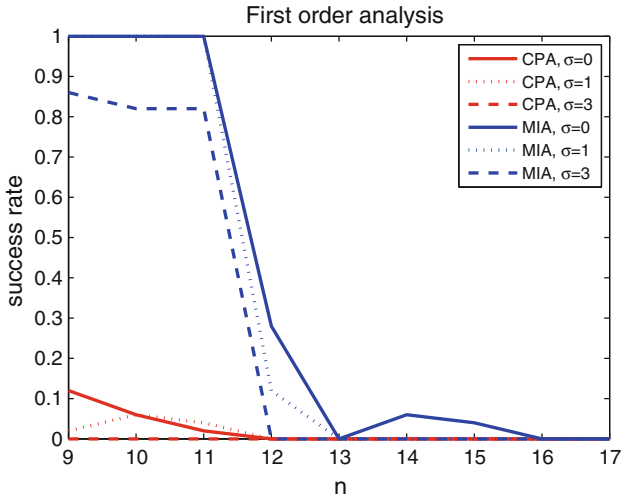

Fig. 2 First-order analysis with L_{simple} and optimal codes: variation of the success rate in function of n

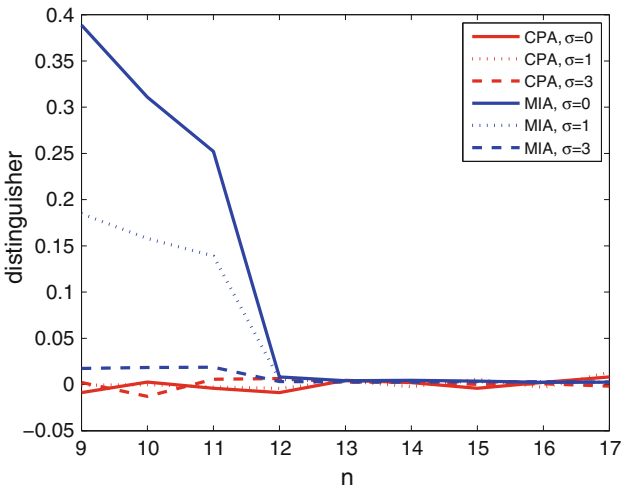

Fig. 3 First-order analysis with L_{simple} and optimal codes: variation of the mutual information and correlation factor in function of n

Second-order analysis: Figure 4 represents the success rate of second-order analysis. We use the center-product method for CPA and we compute the mutual information of the triplet $\{W_1, W_2, \varphi(V_{K,i})\}$ for MIA. The obtained result shows that the center-product CPA is better than MIA in second-order analysis. When the noise level is weak, the secret key can be revealed (for all values of n) with only 1,000 messages. When the noise level is more significant, the attack becomes more difficult with $n > 12$.

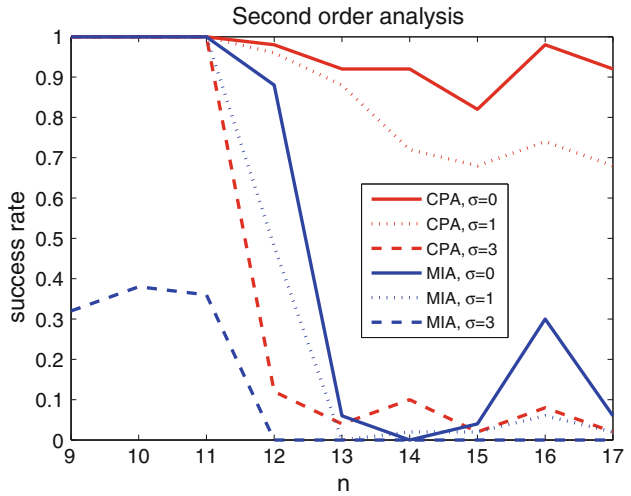

Fig. 4 Second-order analysis with L_{simple} and optimal codes: variation of the success rate in function of n

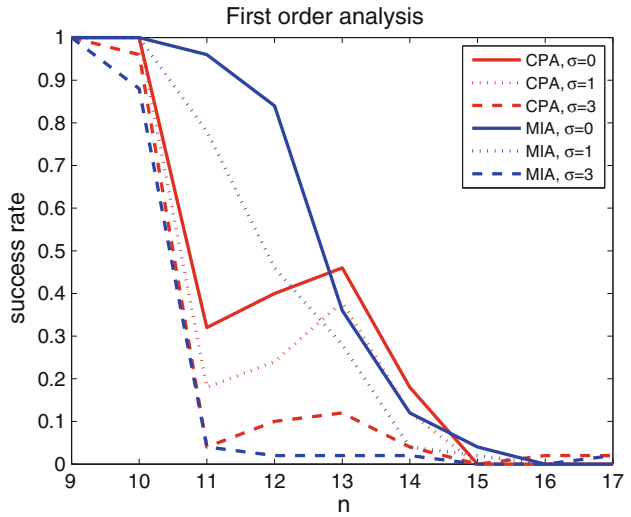

Fig. 5 First-order analysis with L_{simple} and a non-optimal code: variation of the success rate in function of n

6.3.2 Non-optimal code

In this paragraph, we evaluate the resistance of wire-tap based masking when non-optimal codes are used. The simulation conditions (the noise level, the number of messages) are identical to the case of optimal codes.

First-order analysis: In order to observe the difference between wire-tap maskings using optimal and non-optimal codes, let us compare Fig. 5 and Fig. 2. The interesting remark is the different behaviors of CPA between the two cases. In Fig. 2 (optimal code), the success rate of CPA is always very low for every value of n while in Fig. 5 (non-optimal code), it is higher, particularly when there is no noise and n is small. It means that using a non-optimal code, the linear dependence

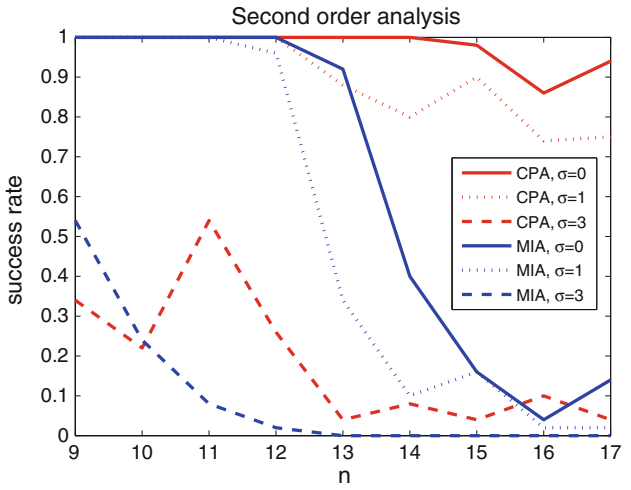

Fig. 6 Second-order analysis with L_{simple} and a non-optimal code: variation of the success rate in function of n

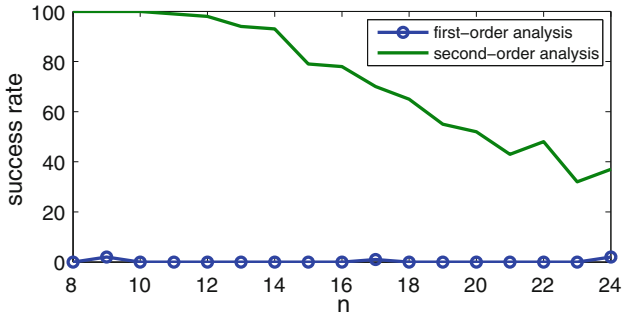

Fig. 7 Evaluation with L_{simple} and random mask r: variation of the success rate in function of n

between the Hamming weight of the prediction and side-channel signals is not totally broken and it is still exploitable by CPA when $n < 16$.

Second-order analysis: The second-order analysis are also performed in the same condition as in the optimal code case and the results are given in Fig. 6. When comparing this figure to Fig. 4, we see that they are quite similar. The second order analysis is still possible and centered-product CPA is better than MIA.

6.3.3 Random masking

We here analyze the differences between wire-tap encoding and the situation where we replace the codeword c by a random mask r of n bits, i.e. somehow affine masking with projection in a larger space. This way, the k effective bits will be spread and masked in n bits as follows: $z = x.L \oplus r$. Note that if the random mask is used, the unmasking without mask is not possible.

Figure 7 represents the variation of the success rate in function of n for both first-order and second-order analysis. When $n = k = 8$, we obtain a kind of affine masking (and if L is of simple form, this leads to simple additive masking).

For the first-order analysis, the success rate is always close to 0, it means that the attack in the first order is impossible. This result confirms once again the previous observation: for wire-tap based masking, the first-order attack works when n is small because of a small number of codewords c.

For the second-order analysis, the success rate decreases when n increases: the projection of intermediate values in a larger space improves the resistance against the second-order analysis.

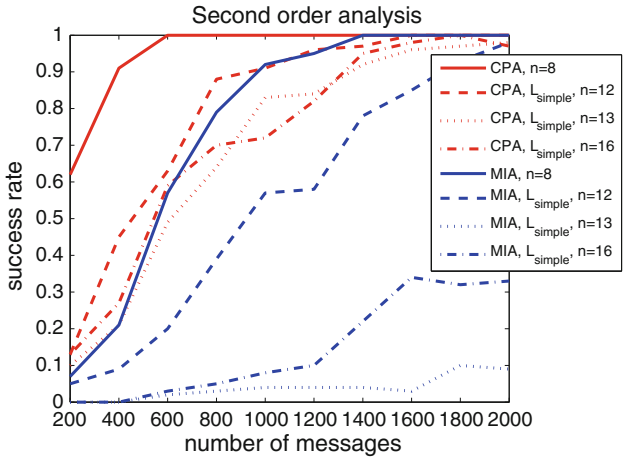

Fig. 8 Second-order analysis: variation of success rate in function of the number of messages N

6.4 Matrix L is unknown

Using wire-tap based masking, the masked values depend on both L and c. In Sect. 6.3, we suppose that the matrix L is known by the attacker. In this section, we want to observe the behavior of the wire-tap based masking in the case where L is unknown for an adversary. It can be the case when L is randomly generated and kept unknown for each device. Thus, both L and c can not be guessed by an adversary.

According to the previous analysis, in this section we consider only optimal codes. Two families of matrices L (the simple ones and the specific ones) are evaluated.

Intuitively, simple-form matrices imply a strong dependence between the Hamming weight of x and the one of $x.L$. On the contrary, more complex specific-form matrices, which are kept unknown, generate mixtures of the bits of x and the dependence between the Hamming weight of x and the Hamming weight of $x.L$ does not hold. The adversary will not be able to estimate the values of $x.L$ without knowing L.

We consider the following methods corresponding to three ways to compute z:

- Classical masking: $z = x \oplus r$
- Wire-tap masking with L_{simple}: $z = x.L_{\text{simple}} \oplus c$
- Wire-tap masking with L_{specific}: $z = x.L_{\text{specific}} \oplus c$

For instance, for the choice of the parity matrix H obtained as the generator matrix of an [16, 8, 5] code (cf. Sect. 5), we can have a family $L_{\text{specific}} = (l_1, \ldots, l_8)$ as follows:

$$l_1 = [0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1]$$
$$l_2 = [1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1]$$
$$l_3 = [0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1]$$
$$l_4 = [0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1]$$
$$l_5 = [0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0]$$
$$l_6 = [1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1]$$
$$l_7 = [1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$
$$l_8 = [0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1]$$

First-order analysis: Even with a large number of messages, the success rate is always close to 0 for the three methods, it means that all of them are resistant against first-order analysis.

Second-order analysis: The results of our simulation experiments show that with specific-form matrices (the third method), the second-order analysis does not allow to find out the secret key even with a large number of messages (evaluated with more than 20,000 messages). The success rate is always close to zero. Therefore, in Fig. 8, we show only the variation of the success rate in the second-order analysis for classical masking method and the wire-tap based masking method with L_{simple}.

To simplify the figure, we select only several values of n to show in Fig. 8. The figure represents the variation of the success rate in function of number of messages varying from 200 to 2,000. The noise level is $\sigma = 1$. According to Fig. 8, the additive masking and the wire-tap masking using simple-form matrices L are not resistant against second-order analysis.

The results confirm once again the following important point: when the set L is more complex (generating mixtures of the bits of x) and unknown, the proposed solution achieves practical resistance against second-order analysis, that is not the case for classical additive masking or the masking using a simple or a known L, thus proving practical resistance of higher order.

6.5 Evaluation synthesis

This paragraph aims at giving a synthesis of evaluation results and proposing a parameter choice strategy when using wire-tap masking.

Our first advice is to use optimal codes which give not only a better performance in term of coding but also a higher resistance against side-channel analysis.

In the case where the matrix L is known or common for different devices, the value n should be higher than 12 to have a good resistance against first-order analysis. Moreover, as shown in Sect. 6.3, when L is known, second-order analysis are still possible if the noise level is weak. Therefore, an implementation using wire-tap masking should consider other means to reduce the efficiency of second-order analysis, for example by minimizing the signal-to-noise ratio or by adding random jitter or clock to make synchronization more difficult.

In the case where the matrix L is unknown and personalized for each device, according to Sect. 6.4, analysis in the first order or higher orders are not possible if L has a complex form and generates mixtures of x. In this case, the generation of L (if L is generated on card), the storage of L, the transfer of this matrix from a memory to another or to CPU and the manipulation of L during the operation $x.L$ should be carefully considered to keep its confidentiality. Finally, using the parameters $n = 12$ or $n = 13$ with unknown randomly generated matrix L seems to offer a good trade-off performance overhead versus security as it ensures a minimum wire-tap security level ($\mu = 2$ or $\mu = 3$).

7 Conclusion

In this paper we have introduced a new kind of protection against SCA based on the wire-tap coding.

From wire-tap coding security, it leads to provable security with respect to Simple Power Analysis. Our analysis and experiments in correlation and mutual information analysis show that it also brings in practice sound countermeasures against first and second order SCA. This first step would be advantageously completed by a hardware implementation to assess its strength with other protection methods. Moreover, it offers new extra properties such as the possibility to unmask the data at the end of the algorithm without the need to use the mask again and a natural fault detection capacity.

Acknowledgments The authors would like to thank the reviewers for their remarks as well as the Editor-in-Chief Cetin K. Koc.

Appendix A: Evaluation results when L is known and has a specific form

A.1 Optimal code

See Figs. 9, 10.

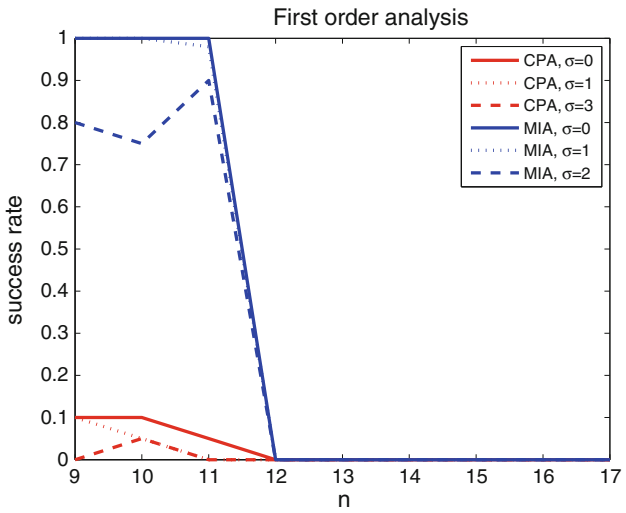

Fig. 9 First-order analysis with specific matrices L and an optimal code: variation of the success rate in function of n

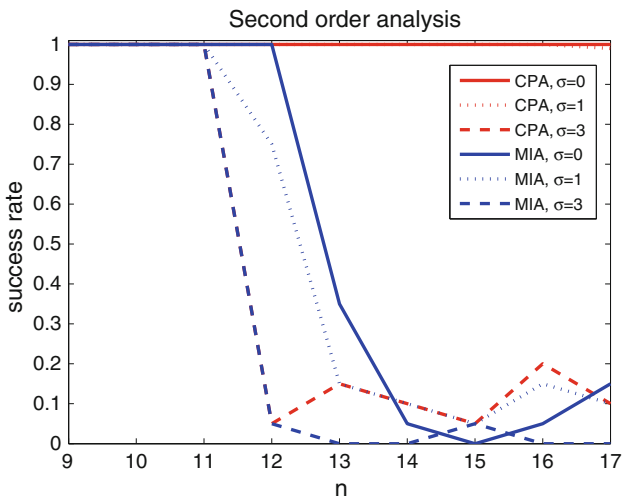

Fig. 10 Second-order analysis with specific matrices L and an optimal code: variation of the success rate in function of n

A.2 Non-optimal code

See Figs. 11, 12.

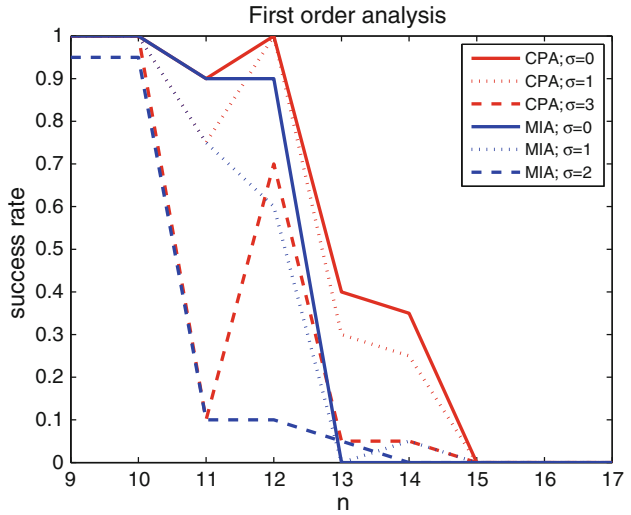

Fig. 11 First-order analysis with specific matrices L and a non-optimal code: variation of the success rate in function of n

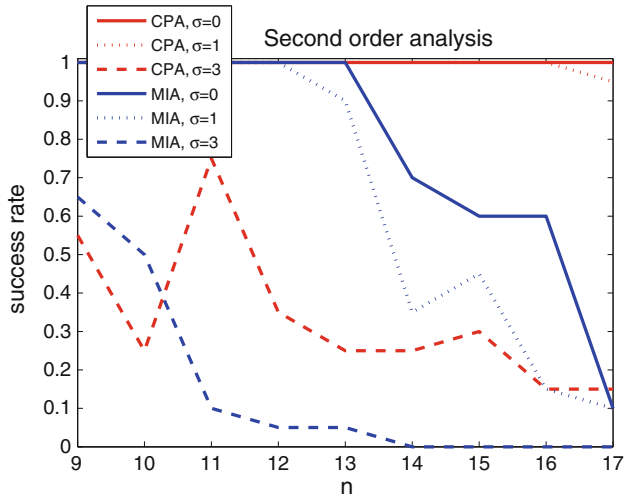

Fig. 12 Second-order analysis with specific matrices L and a non-optimal code: variation of the success rate in function of n

References

1. Akkar, M.L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES, Lecture Notes in Computer Science, vol. 2162, pp. 309–318. Springer, Berlin (2001)
2. Blömer, J., Guajardo, J., Krummel, V.: Provably secure masking of aes. In: Handschuh, H., Hasan, M.A. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 3357, pp. 69–83. Springer, Berlin (2004)
3. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer, Berlin (1999)
4. Fumaroli, G., Martinelli, A., Prouff, E., Rivain, M.: Affine masking against higher-order side channel analysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 6544, pp. 262–280. Springer, Berlin (2010)
5. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting higher-order dpa attacks:. In: Pieprzyk, J. (ed.) CT-RSA, Lecture Notes in Computer Science, vol. 5985, pp. 221–234. Springer, Berlin (2010)
6. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Ç.K., Paar, C. (eds.) CHES, Lecture Notes in Computer Science, vol. 1717, pp. 158–172. Springer (1999)
7. Grassl, M.: Code tables: bounds on the parameters of various types of codes. http://www.codetables.de/, visited in 2010
8. Joye, M., Paillier, P., Schoenmakers, B.: On second-order differential power analysis. In: Rao, J.R., Sunar, B. (eds.) CHES, Lecture Notes in Computer Science, vol. 3659, pp. 293–308. Springer, Berlin (2005)
9. Li, Y., Sakiyama, K., Kawamura, S., Komano, Y., Ohta, K.: Security evaluation of a dpa-resistant s-box based on the fourier transform. In: Qing, S., Mitchell, C.J., Wang, G. (eds.) ICICS, Lecture Notes in Computer Science, vol. 5927, pp. 3–16. Springer, Berlin (2009)
10. MacWilliams, F.J., Sloane, N.J.A.: The theory of error correcting codes. North-Holland, Amsterdam (1977)
11. Medwed, M., Schmidt, J.M.: Coding schemes for arithmetic and logic operations—how robust are they? In: Youm, H.Y., Yung, M. (eds.) WISA, Lecture Notes in Computer Science, vol. 5932, pp. 51–65. Springer, Berlin (2009)
12. Messerges, T.S.: Using second-order power analysis to attack dpa resistant software. In: Koç, Ç.K., Paar, C. (eds.) CHES, Lecture Notes in Computer Science, vol. 1965, pp. 238–251. Springer, Berlin (2000)
13. National Institute of Standards and Technology: Advanced Encryption Standard (FIPS PUB 197) (2001). http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf
14. Oswald, E., Mangard, S., Pramstaller, N.: Secure and efficient masking of aes—a mission impossible? Cryptology ePrint Archive, Report 2004/134 (2004). http://eprint.iacr.org/
15. Ozarow, L.H., Wyner, A.D.: Wire-tap channel II. Bell Syst. Tech. J. **63**(10), 2135–2157 (1984)
16. Ozarow, L.H., Wyner, A.D.: Wire-tap channel ii. In: EUROCRYPT, pp. 33–50 (1984)
17. Prouff, E., Giraud, C., Aumônier, S.: Provably secure s-box implementation based on fourier transform. In: Goubin, L., Matsui, M. (eds.) CHES, Lecture Notes in Computer Science, vol. 4249, pp. 216–230. Springer, Berlin (2006)
18. Prouff, E., Rivain, M.: Theoretical and practical aspects of mutual information based side channel analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS, Lecture Notes in Computer Science, vol. 5536, pp. 499–518. Springer, Berlin (2009)
19. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
20. Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA, Lecture Notes in Computer Science, vol. 3860, pp. 208–225. Springer, Berlin (2006)
21. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT, Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer, Berlin (2009)
22. Thangaraj, A., Dihidar, S., Calderbank, A.R., McLaughlin, S.W., Merolla, J.M.: Capacity achieving codes for the wire tap channel with applications to quantum key distribution. CoRR cs.IT/0411003 (2004)
23. Tillich, S., Herbst, C.: Attacking state-of-the-art software countermeasures-a case study for aes. In: Oswald E., Rohatgi P.

(eds.) CHES, Lecture Notes in Computer Science, vol. 5154, pp. 228–243. Springer, Berlin (2008)
24. von Willich, M.: A technique with an information-theoretic basis for protecting secret data from differential power attacks. In: Honary B. (ed.) IMA International Conference on Lecture Notes in Computer Science, vol. 2260, pp. 44–62. Springer, Berlin (2001)
25. Wyner, A.D.: The wire-tap channel. Bell Syst. Tech. J. **54**(8), 1355–1387 (1975)