Two Power Analysis Attacks against One-Mask Methods

Mehdi-Laurent Akkar[1], Régis Bévan[2], Louis Goubin[3]

[1] Texas Instruments
821 Avenue Jack-Kilby, BP 5, 06271 Villeneuve-Loubet Cedex, France
`akkar@ti.com`
[2] Oberthur Card Systems
25 rue Auguste Blanche, 92800 Puteaux, France
`r.bevan@oberthurcs.com`
[3] Axalto Cryptography Research & Advanced Security
36-38 rue de la Princesse, BP 45, 78431 Louveciennes Cedex, France
`LGoubin@axalto.com`

Abstract. In order to protect a cryptographic algorithm against Power Analysis attacks, a well-known method consists in hiding all the internal data with randomly chosen masks.
Following this idea, an AES implementation can be protected against Differential Power Analysis (DPA) by the "*Transformed Masking Method*", proposed by Akkar and Giraud at CHES'2001, requiring *two* distinct masks. At CHES'2002, Trichina, De Seta and Germani suggested the use of a *single* mask to improve the performances of the protected implementation. We show here that their countermeasure can still be defeated by usual first-order DPA techniques.
In another direction, Akkar and Goubin introduced at FSE'2003 a new countermeasure for protecting secret-key cryptographic algorithms against high-order differential power analysis (HO-DPA). As particular case, the "*Unique Masking Method*" is particularly well suited to the protection of DES implementations. However, we prove in this paper that this method is not sufficient, by exhibiting a (first-order) enhanced differential power analysis attack. We also show how to avoid this new attack.

Keywords: Tamper-resistant devices, Side-Channel attacks, Power Analysis, DPA, Transformed Masking Method, Unique Masking Method, DES, AES.

1 Introduction

The framework of Differential Power Analysis (also known as DPA) was introduced by P. Kocher, J. Jaffe and B. Jun in 1998 ([13]) and subsequently published in 1999 ([14]). The initial focus was on symmetrical cryptosystems such as DES (see [13, 16, 2]) and the AES candidates (see

[4, 5, 8]), but public key cryptosystems have since also been shown to be vulnerable to the DPA attacks (see [17, 7, 11, 12, 19]).

In software two main families of countermeasures against DPA are known:

- In [11, 12], L. Goubin and J. Patarin described a generic countermeasure consisting in splitting all the intermediate variables, using the secret sharing principle. This *duplication method* was also proposed shortly after by S. Chari *et al.* in [5] and [6].
- In [3], M.-L. Akkar and C. Giraud introduced the *transformed masking method* (TMM), an alternative countermeasure to the DPA. The basic idea is to perform all the computation such that all the data are XORed with a random mask. Moreover, the tables (*e.g.* the DES S-Boxes) are modified such that the output of a round is masked by the same mask as the input.

Both these methods have been proven secure against the initial DPA attacks, and are now widely used in real life implementations of many algorithms.

The TMM method can be used to protect AES implementations against DPA. Two masking values are then required to cope with the (non-linear) ByteSub operation. In a recent paper E. Trichina, D. De Seta, L. Germani [20] proposed the "*Simplified Adaptive Multiplicative Masking*" (SAMM), a variation of TMM with a single masking value, thus providing simpler and faster implementations for AES. Unfortunately, we will show in this paper that this method can be broken by usual DPA attacks.

Also suggested by P. Kocher, J. Jaffe and B. Jun [13, 14], and formalized by T. Messerges [15], Higher-Order Differential Power Analysis (HO-DPA) consists in studying correlations between the secret data and *several* points of the electric consumption curves (instead of *single* points for the basic DPA attack). To protect secret-key algorithms against this new class of attacks, M.-L. Akkar and L. Goubin recently proposed [1] a new countermeasure: the so-called "*Unique Masking Method*" (UMM).

In this paper, we describe an unexpected power-analysis attack, which can be applied to implementations of secret-key algorithms using the UMM method. More precisely, in the chosen-plaintext model, the attacker can recover the secret key by successively applying two classical DPA attacks on the second round.

The paper is organized as follows:

- In section 2, we recall basic notions about Differential Power Analysis (DPA), the Transformed Masking Method (TMM) and about the Simplified Adaptive Multiplicative Masking (SAMM).
- In section 3, we analyze the mathematical hypotheses on which the security of SAMM relies and point out a flaw in the design of the countermeasure.
- In section 4, we show how this flaw can be exploited by studying the power consumption of a real component.
- In section 5, we recall basic notions about Higher-Order Differential Power Analysis (HO-DPA) and about the Unique Masking Method (UMM).
- In section 6, we theoretically study the security of the UMM applied to DES and show how it could be cryptanalysed.
- In section 7, we give our perspectives and conclusions about the attacks presented in this paper.

2 Background

2.1 Differential Power Analysis

Differential Power Analysis (DPA) was introduced by Kocher, Jaffe and Jun in 1998 [13] and published in 1999 [14]. The basic idea is to make use of potential *correlations* between the data handled by the micro-controller and the electric consumption measured values. Since these correlations are often very low, *statistical* methods must be applied to deduce sufficient information from them.
The principle of DPA attacks consists in comparing consumption values measured on the *real* physical device (for instance a GSM chip or a smart card) with values computed in an *hypothetical model* of this device (the hypotheses being made among others on the nature of the implementation, and chiefly on a part of the secret key). By comparing these two sets of values, the attacker tries to recover all or part of the secret key.
The initial target of DPA attacks was limited to symmetric algorithms. Vulnerability of DES – first shown by Kocher, Jaffe and Jun [13, 14] – was further studied by Goubin and Patarin [11, 12], Messerges, Dabbish, Sloan [16] and Akkar, Bévan, Dischamp, Moyart [2]. Applications of these attacks were also largely taken into account during the AES selection process, notably by Biham, Shamir [4], Chari, Jutla, Rao, Rohatgi [5] and Daemen, Rijmen [8].
However public-key algorithms were also shown to be threatened: Goubin, Patarin [11, 12] and Messerges, Dabbish, Sloan [17] showed how to apply

DPA against RSA, and the case of elliptic curve cryptosystems was analyzed by Coron [7], Okeya, Sakurai [19] and many others (see for instance [10] for a detailed bibliography).
In the basic DPA attack (see [13, 14] or [9]), also known as first-order DPA (or just DPA), the attacker records the power consumption signals and computes statistical properties of the signal for each individual moment in time of the computation. This attack does not require any knowledge about the individual electric consumption of each instruction, nor about the position in time of each of these instructions. It only relies on the following fundamental hypothesis (quoted from [12]):

Fundamental hypothesis (order 1): *There exists an intermediate variable, that appears during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows us to decide whether two inputs (respectively two outputs) give or not the same value for this variable.*

2.2 The Transformed Masking Method for AES

More details about this technique can be found in [3].
The idea is to mask the message at the beginning of the AES algorithm, and to recover the same mask at the end of each round. An important step for the AES is to securely perform the inversion step. For this, one need to compute $A_{i,j}^{-1} \oplus X_{i,j}$ from $A_{i,j} \oplus X_{i,j}$ where $A_{i,j}$ is the block (i, j) in an AES computation and $X_{i,j}$ the corresponding masking value. To perform this securely, Akkar and Giraud proposed to use the following operations (see Fig. 1):

1. Multiply the masked value A by a non zero random value Y to get $AY \oplus XY$
2. XOR with XY to get AY
3. Perform the inversion to get $A^{-1}Y^{-1}$
4. XOR with XY^{-1} to get $A^{-1}Y^{-1} \oplus XY^{-1}$
5. Multiply by Y to get $A^{-1} \oplus X$

2.3 The Simplified Adaptive Multiplicative Masking for AES

More details about this technique can be found in [20].
The general idea is not to use a different Y masking value to switch from an additive mask to a multiplicative one, but to use the same masking value X instead. The algorithm is the following.

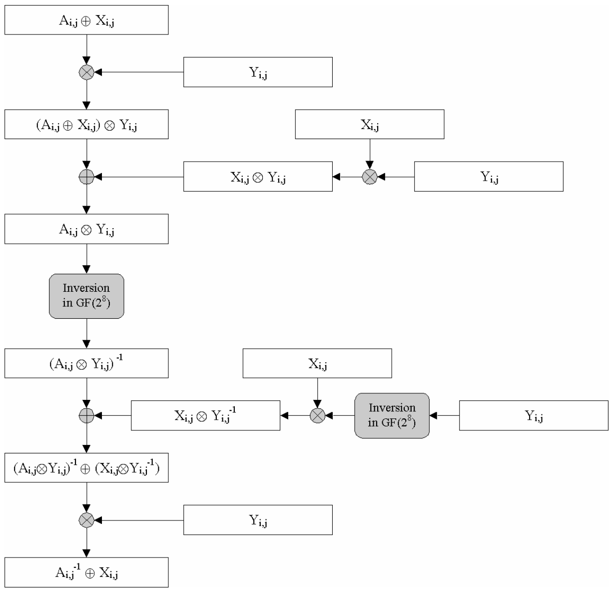

Fig. 1. Modified inversion in GF(2^8) with masking countermeasure.

1. Multiply the masked value A by X to get $AX \oplus X^2$
2. XOR with X^2 to get AX
3. Perform the inversion to get $A^{-1}X^{-1}$
4. XOR with 1 to get $A^{-1}X^{-1} \oplus 1$
5. Multiply by X to get $A^{-1} \oplus X$

One can notice that the particular value $AX \oplus X^2$ appears during the computation. The authors of [20] suggest that even if $AX \oplus X^2$ is not fully random, it is sufficiently random to serve the purpose. In the next section, we will precisely study the function $AX \oplus X^2$ and show that it introduces a weakness in the method.

3 Theoretical Analysis of the SAMM

3.1 Study of the repartition of $AX + X^2$

In the following part we will denote:

$$\mathbb{K} = \mathbb{F}_2 \;\; and \;\; \mathbb{K}_8 = \mathbb{F}_{2^8}$$

We will also define:

$$\begin{aligned} f_A\colon \mathbb{K}_8 &\longrightarrow \mathbb{K}_8 \\ X &\longrightarrow AX + X^2 \end{aligned}$$

and

$$\begin{aligned} F\colon \mathbb{K}_8 &\longrightarrow \mathcal{P}(\mathbb{K}_8) \\ A &\longrightarrow \{f_A(X),\ X \in \mathbb{K}_8\} \end{aligned}$$

where $\mathcal{P}(\mathbb{K}_8)$ represents the power set of $\mathbb{K}_8$.

Remark: In what follows, we will denote by $\#(A)$ the cardinality of A.

With these definitions we have the following result:

Theorem 1. *If $A \in \mathbb{K}_8$ then we have:*

$$\begin{aligned} \#F(A) &= 128 \text{ if } A \neq 0 \\ &= 256 \text{ if } A = 0 \end{aligned}$$

Proof.
- Case $A = 0$: $f_0(X) = X^2$ and is bijective on $\mathbb{K}_8$. Therefore we obviously have $\#F(0) = 256$.
- Case $A \neq 0$: $f_{A\neq 0}(X) = AX + X^2$, if given a Y, we want to solve the equation $f_A(X) = Y$, by defining $Z = X/A$ (because $A \neq 0$), we obtain the following equation to solve:

$$Z^2 + Z = Y/A^2$$

and it is well known that this equation has no solution if $Trace(Y/A^2) = 1$ and two solutions (if one gets one solution W the other is $W + 1$) if $Trace(Y/A^2) = 0$. Therefore it is easy to see that $\#F(A) = 128$.
This theorem shows that the simplified mask covers only one half of the 256 possible values on $\mathbb{K}_8$. This was already noticed in the article [20].
However, let us now study the distribution of $F(A_1)$ and $F(A_2)$, for two distinct values A_1 and A_2. The following proposition gives us a more precise result:

Theorem 2. *If $(A_1, A_2) \in (\mathbb{K}_8 \setminus \{0\})^2$ and $A_1 \neq A_2$ then:*

$$\#(F(A_1) \cap F(A_2)) = 64$$

Proof.
Let A_1 and A_2 be two elements of $\mathbb{K}_8 \setminus \{0\}$. We are looking for the values Y such as the two equations

$$\begin{cases} Y = A_1X + X^2 \\ Y = A_2X' + X'^2 \end{cases}$$

are simultaneously solvable or unsolvable. This is equivalent to:

$$Trace(Y/A_1) = Trace(Y/A_2)$$

By linearity of the trace operator we obtain:

$$Trace(Y/A_1 - Y/A_2) = 0$$

Let now consider $Z \in \mathbb{K}_8$. To Z corresponds a unique value Y such as $Y/A_1 - Y/A_2 = Z$, which is $Y = Z/(1/A_1 - 1/A_2)$. Since there exist 128 elements Z of trace 0, there exist 128 elements Y such that the previous system is simultaneously solvable or unsolvable.
Let us now consider:

$$\begin{cases} n_1 = \{Y \in \mathbb{K}_8 \text{ such that } Trace(Y/A_1) = 0 \text{ and } Trace(Y/A_2) = 1\} \\ n_2 = \{Y \in \mathbb{K}_8 \text{ such that } Trace(Y/A_2) = 0 \text{ and } Trace(Y/A_1) = 1\} \\ n_3 = \{Y \in \mathbb{K}_8 \text{ such that } Trace(Y/A_1) = 0 \text{ and } Trace(Y/A_2) = 0\} \\ n_4 = \{Y \in \mathbb{K}_8 \text{ such that } Trace(Y/A_2) = 1 \text{ and } Trace(Y/A_1) = 1\} \end{cases}$$

The last result gives us the following equation: $n_3 + n_4 = 128$
This equation, together with obvious trace considerations, gives the following system:

$$\begin{cases} n_3 + n_4 = 128 \\ n_1 + n_3 = 128 \\ n_2 + n_3 = 128 \\ n_2 + n_4 = 128 \end{cases}$$

Solving this system, we obtain: $n_1 = n_2 = n_3 = n_4 = 64$, thus achieving the proof of Theorem 2.

3.2 Consequences

Theorem 2 is really important because it proves that two distinct values (during the computation) are "projected" onto two sets of 128 values when the SAMM is implemented and that these two sets have 64 common values and 64 different ones. Therefore when an attacker performs a DPA on an implementation including the SAMM he will on average record the consumption of $F(A)$ instead of A but for two distinct values A and B, $F(A)$ and $F(B)$ are also distinct, allowing the attacker to distinguish the two cases. This give strong evidence that the attack is very likely to work. Moreover even if the attacker does not know the fact that SAMM is implemented, he will be able to recover the key because the attack is exactly the same.

4 Semi-Real and Real Analysis of the SAMM

We have seen in the previous section that there is a theoretical flaw in the SAMM method. To go further, we have to check which results are obtained when using different models of power consumption. We also have to experiment on a real embedded device. More work about the consumption model of embedded device can be found in many papers: see for example [6, 2, 16].

4.1 Linear Model

The "linear model" considers that the power consumption of the card is proportional to the value of each bit of the manipulated value. For example the consumption of an 8-bit value

$$X = b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7$$

will be equal to

$$c_0 * b_0 + c_1 * b_1 + c_2 * b_2 + c_3 * b_3 + c_4 * b_4 + c_5 * b_5 + c_6 * b_6 + c_7 * b_7$$

where $\{c_i\}_{i \leq 8}$ is the average consumption of bit i. For example the Hamming weight (denoted by HW) model is a linear one with $c_i = c_j$ for all (i, j).

We have performed some experimentations in the following way. We have computed for every $A = 0..255$ the whole subset $F(A)$ and we have computed the average weight of each bit of the values in $F(A)$ to check if there was any bias in the results. The results are as follows:

- For 8 values of A (0x03,0x15,0x87,0x8C,0xCE,0xEB,0xED and 0xF6), one specific bit of the eight bits of $f_A(X)$ always vanishes, whatever the value of X is!
- For the 248 other values A, the probability that "the i-th bit of $f_A(X)$ is equal to zero" is equal to $\frac{1}{2}$. So an attacker is unable to predict one bit in order to compute a classical DPA attack.

If we now analyze the repartition of the HW of the values $F(A)$ we get the nine following sets:

Subset	#	HW 0	HW 1	HW 2	HW 3	HW 4	HW 5	HW 6	HW 7	HW 8
1	1	1	8	28	56	70	56	28	8	1
2	28	2	12	32	52	60	52	32	12	2
3	56	2	10	26	50	70	62	30	6	0
4	8	2	14	42	70	70	42	14	2	0
5	70	2	8	24	56	76	56	24	8	2
6	28	2	4	32	60	60	60	32	4	2
7	8	2	2	42	42	70	70	14	14	0
8	56	2	6	26	62	70	50	30	10	0
9	1	2	0	56	0	140	0	56	0	2

For example, line 8 in the table means that there are 56 values of A such as, there exists 2 values X for which $HW(f_A(X)) = 0$, 6 values X for which $HW(f_A(X)) = 1$, ..., and no element X such as $HW(f_A(X)) = 8$. Moreover, one can notice that the average Hamming weight of all the subsets is equal to 4 except for the subset 4 (which contains the 8 special values detailed in the previous paragraph), whose mean is equal to 3.5. The explicit values A of the nine subsets can be found in the appendix.
The conclusion is that even if only a small bias exists, if a component respects a linear model of consumption (such as the HW model), the masking method would probably be quite efficient in practice. However, one has to check what happens with a real embedded device: some experiments on an 8051-based 8-bit CPU are discussed in the following section.

4.2 Real Device Analysis of the SAMM

To obtain concrete results, we have made a comparison between the power consumption of a card manipulating a value A and a card manipulating the value $AX \oplus X^2$ with random X. For this we have recorded the consumption of a load operation on an 8 bits CPU based on an 8051 core.
The following curves (see Fig. 2 and Fig. 3) have been obtained by using the average consumption of 1024 power consumption traces. For the unmasked value we have used the record of 512 times the same value A. For the SAMM we have used the average of twice the 256 values $AX \oplus X^2$ with $X \in [0, 255]$. Then we have ordered the value per consumption to get an idea of the variance of the value. As seen before there are eight special values (the ones for which one bit is always 0) that we have excluded from

the curves, indeed the consumption of these values was really different from the others[1].

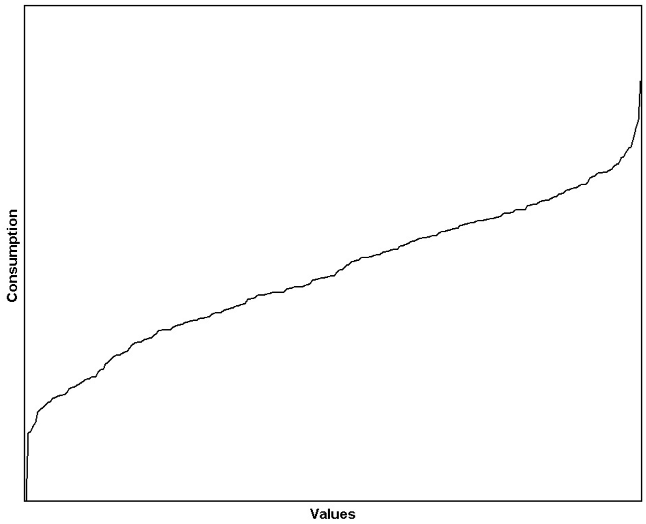

Fig. 2. Repartition of the consumption with unmasked values

As can be seen on these two curves, masked or unmasked, there is a quite important difference between different consumption values. That proves that, in a real embedded device, even if it gets somewhat slower (the variance in the SAMM curve is smaller), a successful DPA attack on a classical AES implementation will also succeed on an AES implementation using the SAMM method.

If the attacker knows that the SAMM method is used, he can use an adapted DPA attack to retrieve the key with less measurements than the usual first order DPA. For every hypothesis K_j on the key byte, one computes the difference of the means of the following sets:

$$S_0 = \{\text{measurements for the messages } M_k | M_k \oplus K_j \in \text{subset } 4\}$$
$$S_1 = \{\text{measurements for the messages } M_k | M_k \oplus K_j \notin \text{subset } 4\}$$

Then the correct hypothesis is the one with the highest difference of means.

[1] That shows that probably a SPA attack may be quite easy by focusing on these particular values !

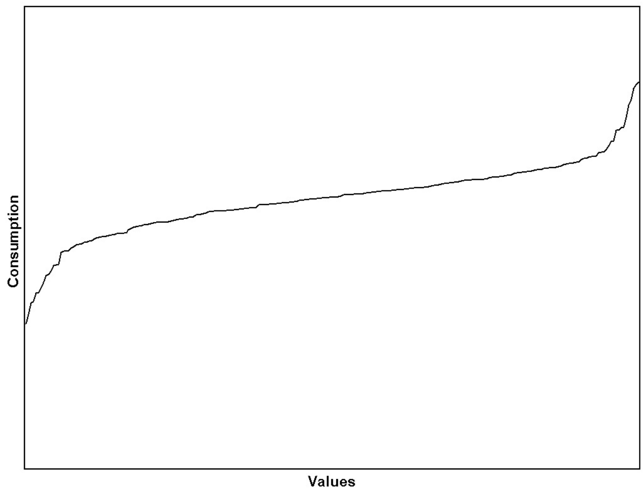

Fig. 3. Repartition of the consumption with SAMM values

Remark. Due to obvious confidentiality reasons, details about the chip we used have to remain undisclosed.

5 The Unique Masking Method

5.1 High-Order Differential Power Analysis

As mentioned in section 2, Differential Power Analysis implies the use of a hypothetical model of the physical device which performs the cryptographic computations. If this model is able to predict a *single* value, for instance the electric consumption of the device for a single instant t, the differential power analysis is said to be of *first order*. If the model is able to predict *several* such values, the differential power analysis is said to be of *higher order*.
High-order differential power analysis (HO-DPA), suggested by Kocher, Jaffe and Jun [13, 14] (see also [9]), was formalized by Messerges in [15]. In the spirit of [12], Akkar and Goubin (see [1]) gave a necessary and sufficient condition for a DPA attack of order n to be applicable:
Fundamental hypothesis (order n): *There exists a set of n intermediate variables, that appear during the computation of the algorithm, such that knowing a few key bits (in practice less than 32 bits) allows to decide whether two inputs (respectively two outputs) give or not the same value for a known function of these n variables.*

In [1], Akkar and Goubin studied the impacts of HO-DPA attacks on implementations of cryptographic algorithms, and showed that usual countermeasures against DPA are not sufficient to avoid this new class of attacks. Moreover, they proposed a generic protection against higher-order attacks, illustrated in details for the DES case.

5.2 A countermeasure against HO-DPA

The so-called "*Unique Masking Method*" (UMM) aims at providing a generic protection against differential power analysis of order n, whatever the value n may be. The two principles of this method is first to mask only the values that depend on less than 32 bits of the key in order to prevent DPA and secondly intermediate independent variables depending on less than 32 bits of the key must not be masked by the same value in order to thwart HO-DPA.
Let us describe the basic idea for the DES example. The unique mask is a random 32-bit value α. From this value, two sets of 8 S-boxes, denoted by $\tilde{S}_1$ and $\tilde{S}_2$, are defined by

$$\begin{cases} \forall x \in (\mathbb{F}_2)^{48},\ \tilde{S}_1(x) = S(x \oplus E(\alpha)) \\ \forall x \in (\mathbb{F}_2)^{48},\ \tilde{S}_2(x) = S(x) \oplus P^{-1}(\alpha) \end{cases}$$

where S denotes the 8 usual DES S-boxes, E is the expansion function and P is the classical DES permutation just after the S-boxes.

Let f_{K_i} $(1 \leq i \leq 16)$ be the functions involved in the Feistel scheme of DES (with the usual S-boxes), and $\tilde{f}_{1,K_i}$ (resp. $\tilde{f}_{2,K_i}$) the analogous function with the $\tilde{S}_1$ (resp. $\tilde{S}_2$) S-boxes. DES rounds can then be built from five possible frames, given in Figure 4 (solid lines correspond to unmasked data, dashed lines to masked data).

To be consistent with the DES computation, the sequence of rounds has to follow some rules, which can be summarized by a finite automaton, as shown in Figure 5. Initial states correspond to non-masked inputs (A or B), final states correspond to non-masked outputs (A or E). As an example, $BCDCDCEBCDCDCDCE$ is a valid sequence.

To provide a protection against differential power analysis attacks, all the values depending on less than 36 key bits are masked by α. This gives further constraints on the sequence of rounds: the three first ones have to be of the form BCD or BCE, and symmetrically the three last ones must be of the form BCE or DCE. For instance, the sequence $BCDCDCEBCDCDCDCE$ fulfills these conditions.

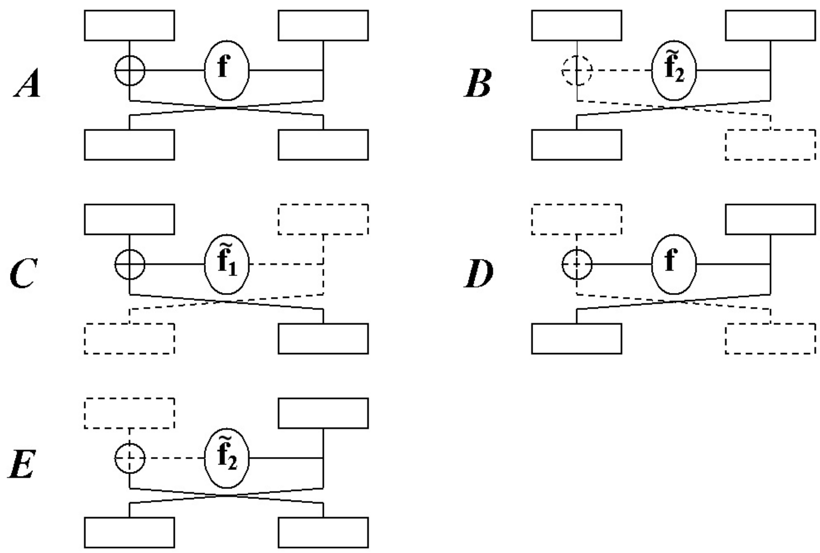

Fig. 4. The five possibilities for DES rounds

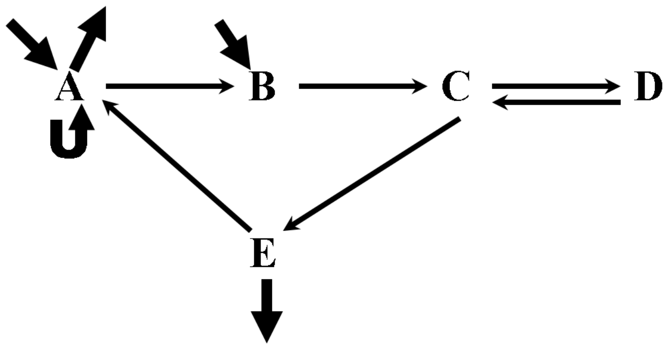

Fig. 5. Valid sequences for the rounds

The unique masking method has several advantages: the structure of classical implementations can be kept unchanged, the only difference is the generation of random tables (see [1] for several practical methods for securely generating S-boxes from the mask α). Moreover performances remain acceptable. For instance, [1] reports on a DES implementation, including the unique masking method (together with SPA and DFA protections), which runs in 38 ms (at 10 MHz) on a ST19 chip.

6 Enhanced DPA of the Unique Masking Method

6.1 Basic idea

For all the proposed sequences of rounds, the second round is always a "C"-type round. The output of the S-Boxes is unmasked and stay unmasked after being XORed with the left part of the message. After this XOR, the value goes through an "E"-type or "D"-type round for which the computation of the beginning of the f function (expansion function) is unmasked. For the second case (where the third round is of "D" type) even the S-Boxes output and the P permutation are unmasked.
The fact that the output of second round S-Boxes is unmasked will be the base of our attack.

6.2 Attacking the UMM

The attack is a chosen message attack. The main idea of the attack is to retrieve two intermediate values which are not protected against DPA, and then to get the key bits by solving an equation involving the two intermediate values. Before describing the attack, let us introduce some notations:

- IP denotes the initial permutation.
- E denotes the expansion function (from 32 to 48 bits).
- S denotes the S-Boxes.
- P denotes the P permutation after the S-Boxes
- $\oplus$ denotes the XOR operation
- K_i denotes the 48 bits subkey of round i.
- Finally, for a message M, L_i and R_i will denote the left part and the right part (32 bits each) of the output of the round i.

The attack can be described as follows:

- **First Part:**

- We perform a DES computation with some chosen messages M_i for which $R_{0,i}$ (the right part of the message M_i after IP) will be set to an arbitrary constant R_0. The left part $L_{0,i}$ will be random.
- We then perform a first-order DPA attack on the input of each S-Box of the second round. Because the output of the S-Boxes is unmasked we will guess the value of the second round key XORed with the – unknown but constant – output of the S-Boxes of the first round. The found value will be:

$$\delta = K_2 \oplus E(P(S(K_1 \oplus E(R_0))))$$

– **Second Part:**
- We will perform a second first-order DPA with other messages with a different known constant value R'_0, which will provide:

$$\delta' = K_2 \oplus E(P(S(K_1 \oplus E(R'_0))))$$

– **Final Part:**
- By taking XOR of the two values found at last steps, we obtain the value:

$$\delta \oplus \delta' = (K_2 \oplus E(P(S(K_1 \oplus E(R_0))))) \oplus (K_2 \oplus E(P(S(K_1 \oplus E(R'_0)))))$$

The value K_2 vanishes and the linearity of functions E and P gives us the equation:

$$S(K_1 \oplus E(R_0)) \oplus S(K_1 \oplus E(R'_0)) = g(\delta \oplus \delta')$$

- Because we know R_0 and R'_0, doing a exhaustive search on each 6-bits subkeys of K_1, will give us all the possible values for the subkey K_1.

On average, the differential properties of S will give us about 4 possibilities for each subkeys. Since there are 8 subkeys and we need to find the 8 bits which are not in K_1, this gives us $4^8 \times 2^8 = 2^{24}$ possibilities on the key. So an exhaustive search with one known plaintext/ciphertext pair will take a few seconds on a PC.

If one does not have access to such a pair, another attack with a R''_0 constant value will decrease the possibilities for K_1 (practically, K_1 is completely known if R''_0 is not badly chosen). Then it is possible to perform classical DPA against K_2, once K_1 is known, to get directly the 56 bits of the DES key.

The reader has to notice that even if the attack exploits the correlation of two results, the attack consists just in applying twice a really usual first-order DPA attack. So the number of traces and the processing time is just twice those needed for a classical DPA against an unprotected DES.

6.3 Attack Scenarios and Countermeasures

- Our attack is feasible only if the attacker is able to keep constant the right part of the message after the initial permutation. If it is not possible the attacker has to "wait" for messages having the same R_0 part (1 message every 2^{32} messages on average). So the attack becomes quite long to perform.
- If we consider a scenario for which only the output is known, the attack becomes as difficult as the one for which the message could not be controlled. A chosen cipher text attack mainly applies against an authentication scheme, in which the device has to cipher a challenge from the outside.
- A solution consists in masking the output of the second round, which seems to make this attack unfeasible. One can use a different mask but the use of α_1 is not forbidden since the bits of R_1 and R_2 that are masked by the same value depends on 42 bits of the key. So we need one more function $\tilde{f}_{3,K_i}$ with the modified S-Boxes $\tilde{S}_3(x)$ such that $\forall x \in (\mathbb{F}_2)^{48}$, $\tilde{S}_3(x \oplus E(\alpha)) = S(x) \oplus P^{-1}(\alpha)$.

7 Conclusion

In this paper we presented two new power analysis attacks.

The first one applies to the Simplified Adaptive Multiplicative Masking (SAMM). Even if, in some models (HW Model or Linear Model), the SAMM seems to be quite secure, the SAMM is (in practice) vulnerable to usual (first-order) DPA attacks. Moreover we have seen that, from a theoretical point of view, this countermeasure is not correct because of the distribution of the values $AX \oplus X^2$, which is quite unbalanced. We thus recommend, to obtain DPA-resistant implementations of AES, using the original TMM method with two masks (correctly implemented due to the zero problem as described in [20, 1]) or, to use a dynamic inversion of the S-Box if the 256 bytes needed in RAM are available.

The second attack applies to the Unique Masking Method for DES, showing that in the chosen-message scenario, an enhanced first-order DPA attack is still possible. This is not due to the method itself but to a different model of the attacker, allowing her to have full access to the inputs of the algorithm. In the case of the DES, we have then shown that unique masks have to be extended to at least two rounds to protect the implementation against a chosen text attack. The drawbacks are an overhead on performances and an increase of required memory by one third (corresponding to the computation of one more modified S-box).

References

1. M.-L. Akkar, L. Goubin, *A Generic Protection against High-Order Differential Power Analysis*. In Proceedings of FSE'2003, LNCS 2887, Springer-Verlag, 2003.
2. M.-L. Akkar, R. Bevan, P. Dischamp, D. Moyart, *Power Analysis: What is now Possible*. In Proceedings of ASIACRYPT'2000, LNCS 1976, pp. 489-502, Springer-Verlag, 2000.
3. M.-L. Akkar, C. Giraud, *An Implementation of DES and AES Secure against Some Attacks*. In Proceedings of CHES'2001, LNCS 2162, pp. 309-318, Springer-Verlag, 2001.
4. E. Biham, A. Shamir, *Power Analysis of the Key Scheduling of the AES Candidates*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`
5. S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`
6. S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power-Analysis Attacks*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 398-412, Springer-Verlag, 1999.
7. J.-S. Coron, *Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems*. In Proceedings of CHES'99, LNCS 1717, pp. 292-302, Springer-Verlag, 1999.
8. J. Daemen, V. Rijmen, *Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals*. In Proceedings of the Second Advanced Encryption Standard (AES) Candidate Conference, March 1999. Available from `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`
9. J. Daemen, M. Peters, G. Van Assche, *Bitslice Ciphers and Power Analysis Attacks*. In Proceedings of FSE'2000, LNCS 1978, Springer-Verlag, 2000.
10. L. Goubin, *A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems*. In Proceedings of PKC'2003, LNCS 2567, pp. 199-211, Springer-Verlag, 2003.
11. L. Goubin, J. Patarin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. European Patent, Bull CP8, February 4th, 1999, Publication Number: 2789535.
12. L. Goubin, J. Patarin, *DES and Differential Power Analysis – The Duplication Method*. In Proceedings of CHES'99, LNCS 1717, pp. 158-172, Springer-Verlag, 1999.
13. P. Kocher, J. Jaffe, B. Jun, *Introduction to Differential Power Analysis and Related Attacks*. Technical Report, Cryptography Research Inc., 1998. Available from `http://www.cryptography.com/dpa/technical/index.html`
14. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.
15. T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant software*. In Proceedings of CHES'2000, LNCS 1965, pp. 238-251, Springer-Verlag, 2000.
16. T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Investigations of Power Analysis Attacks on Smartcards*. In Proceedings of the USENIX Workshop on Smartcard Technology, pp. 151-161, May 1999. Available from `http://www.eecs.uic.edu/~tmesserg/papers.html`

17. T.S. Messerges, E.A. Dabbish, R.H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*. In Proceedings of CHES'99, LNCS 1717, pp. 144-157, Springer-Verlag, 1999.
18. I. Mironov, *(Not So) Random Shuffles of RC4*. In Proceedings of CRYPTO'2002, LNCS 2442, pp. 304-319, Springer-Verlag, 2002.
19. K. Okeya, K. Sakurai, *Power Analysis Breaks Elliptic Curve Cryptosystem even Secure against the Timing Attack*. In Proceedings of INDOCRYPT'2000, LNCS 1977, pp. 178-190, Springer-Verlag, 2000.
20. E. Trichina, D. De Seta, L. Germani, *Simplified Adaptive Multiplicative Masking for AES*. In Proceedings of CHES'2002, LNCS 2523, pp. 187-197, Springer-Verlag, 2002.

Appendix. Explicit subsets of values A, depending of the repartition of the Hamming weight of $f_A(X)$.

- Subset 1: 0x00
- Subset 2: 0x01, 0x06, 0x09, 0x22, 0x24, 0x2B, 0x46, 0x48, 0x56, 0x62, 0x65, 0x68, 0x75, 0x7B, 0x7C, 0x7D, 0x99, 0x9D, 0xA9, 0xB1, 0xC1, 0xCA, 0xCD, 0xEF, 0xF0, 0xF8, 0xFA, 0xFB
- Subset 3: 0x02, 0x04, 0x07, 0x0C, 0x0E, 0x12, 0x18, 0x19, 0x1C, 0x23, 0x29, 0x2A, 0x2D, 0x2E, 0x31, 0x37, 0x3E, 0x3F, 0x44, 0x49, 0x52, 0x54, 0x59, 0x5B, 0x5C, 0x5F, 0x60, 0x67, 0x6B, 0x78, 0x79, 0x81, 0x89, 0x8D, 0x8E, 0x8F, 0x90, 0x95, 0x96, 0x98, 0x9C, 0xA5, 0xB0, 0xB2, 0xB6, 0xB7, 0xB8, 0xBF, 0xC0, 0xC3, 0xD6, 0xD9, 0xE1, 0xE7, 0xEA, 0xF7
- Subset 4: 0x03, 0x15, 0x87, 0x8C, 0xCE, 0xEB, 0xED, 0xF6
- Subset 5: 0x05, 0x08, 0x0A, 0x0D, 0x11, 0x1A, 0x1D, 0x1E, 0x1F, 0x21, 0x27, 0x32, 0x34, 0x38, 0x3A, 0x3B, 0x3D, 0x43, 0x47, 0x4B, 0x4C, 0x4E, 0x4F, 0x51, 0x64, 0x69, 0x6D, 0x6E, 0x76, 0x77, 0x7E, 0x85, 0x88, 0x92, 0x97, 0x9B, 0x9E, 0xA0, 0xA1, 0xA2, 0xA4, 0xA8, 0xAA, 0xAC, 0xAE, 0xB3, 0xB9, 0xBB, 0xBE, 0xC4, 0xC5, 0xC7, 0xC8, 0xC9, 0xCF, 0xD0, 0xD1, 0xD2, 0xD5, 0xDC, 0xE5, 0xE8, 0xE9, 0xEC, 0xF3, 0xF4, 0xF5, 0xFD, 0xFE, 0xFF
- Subset 6: 0x0B, 0x0F, 0x10, 0x20, 0x26, 0x28, 0x36, 0x39, 0x40, 0x5E, 0x63, 0x6C, 0x6F, 0x83, 0x8A, 0x9F, 0xA7, 0xAD, 0xB5, 0xBC, 0xBD, 0xC2, 0xCC, 0xD8, 0xDB, 0xE4, 0xE6, 0xF9
- Subset 7: 0x13, 0x1B, 0x2C, 0x66, 0x72, 0x80, 0x84, 0xD3
- Subset 8: 0x14, 0x17, 0x25, 0x2F, 0x30, 0x33, 0x35, 0x3C, 0x41, 0x42, 0x45, 0x4A, 0x4D, 0x50, 0x53, 0x55, 0x57, 0x58, 0x5A, 0x5D, 0x61, 0x6A, 0x70, 0x71, 0x73, 0x74, 0x7A, 0x7F, 0x82, 0x86, 0x8B, 0x91, 0x93, 0x94, 0x9A, 0xA3, 0xA6, 0xAB, 0xAF, 0xB4, 0xBA, 0xC6, 0xCB, 0xD4, 0xD7, 0xDA, 0xDD, 0xDE, 0xDF, 0xE0, 0xE2, 0xE3, 0xEE, 0xF1, 0xF2, 0xFC
- Subset 9: 0x16