A Generic Protection against High-Order Differential Power Analysis

Mehdi-Laurent Akkar (makkar@slb.com),
Louis Goubin (lgoubin@slb.com)

Cryptography Research, Schlumberger Smart Cards
36-38 rue de la Princesse, BP 45, F-78430 Louveciennes Cedex, France

Abstract. Differential Power Analysis (DPA) on smart-cards was introduced by Paul Kocher [11] in 1998. Since, many countermeasures have been introduced to protect cryptographic algorithms from DPA attacks. Unfortunately these features are known not to be efficient against high order DPA (even of second order). In these paper we will first describe new specialized first order attack and remind how are working high order DPA attacks. Then we will show how these attacks can be applied to two usual actual countermeasures. Eventually we will present a method of protection (and apply it to the DES) which seems to be secure against any order DPA type attacks. The figures of a real implementation of this method will be given too.

Keywords: Smart-cards, DES, Power analysis, High-Order DPA

1 Introduction

The framework of Differential Power Analysis (also known as DPA) was introduced by P. Kocher, B. Jun and J. Jaffe in 1998 ([11]) and subsequently published in 1999 ([12]). The initial focus was on symmetrical cryptosystems such as DES (see [11, 14, 1]) and the AES candidates (see [3, 4, 7]), but public key cryptosystems have since also been shown to be also vulnerable to the DPA attacks (see [15, 6, 9, 10, 16]).

Two main families of countermeasures against DPA are known:

- In [9, 10], 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 [4] and [5].
- In [2], M.-L. Akkar and C. Giraud introduced the *transformed masking method*, 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. However, they do not take into consideration more elaborated attacks called "High-order DPA". These attacks, as described in [11] by P. Kocher or in [13] by T. Messerges, consist in studying correlations between the secret data and *several* points of the electric consumption curves (instead of *single* points for the basic DPA attack).
In what follows, we study the impact of the High-order DPA attacks on both countermeasures mentioned above. Moreover, we describe new secure ways of implementing a whole class of algorithms (including DES) against these new attacks.

The paper is organized as follows:

- In section 2, we recall three basic notions: the (high-order) differential power analysis, the duplication method and the transformed masking method.
- In Section 3, we study "duplication method" and show that an implementation of DES (or AES), which splits all the variables into n sub-variables is still vulnerable to an n-th order DPA attack. Section 3.1 gives the general principle of the attack and section 3.2 discusses practical aspects.
- Section 4 is devoted to the analysis of the "transformed masking" (see [2]). For such an implementation of DES, section 4.1 describes how a "second order" DPA can work. A new variant we call the "superposition attack" is also presented. In section 4.2, we show that an AES (=Rijndael) implementation protected with the "transformed masking" method can also be attacked, either by second order DPA, or by the "Zero problem" attack.
- Section 5 presents our new generic countermeasure: the "unique masking method". We illustrate it on the particular case of DES. In 5.1, we explain the main idea of "unique mask". In 5.2, we apply it to the full protection of a DES implementation. The security of this implementation against n-th order DPA attacks is investigated in sections 5.3 and 5.4.
- Section 6 focuses on the problem of securely constructing the modified S-Boxes used in our new countermeasure. The details of the algorithm are presented, together with practical impacts on the amount of time and memory needed.

– In Section 7, we give our conclusions.

2 Background

2.1 The (High-Order) Differential Power Analysis

In basic DPA attack (see [11, 12], or [8]), also known as first-order DPA (or DPA when there is no risk of confusion), the attacker records the power consumption signals and compute statistical properties of the signal for each individual instant 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 [10]):

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.*

In this paper, we consider the so-called *High-Order Differential Power Analysis* attacks (HODPA), which generalize the first-order DPA: the attacker now compute statistical correlations between the electrical consumptions considered at several instants. More precisely, an "n-th order" DPA attack takes into account n values of the consumption signal, which correspond to n intermediate values occurring during the computation.
These attacks now rely on the following fundamental hypothesis (in the spirit of [10]):
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 us to decide whether two inputs (respectively two outputs) give or not the same value for a known function of these n variables.*

2.2 The "Duplication" Method

The "duplication method" was initially suggested by L. Goubin and J. Patarin in [9], and studied further in [4, 10, 5]. It basically consists in splitting the data (manipulated during the computation) into several parts, using a secret sharing scheme, and computing a modified algorithm on each part to recombine the final result at the end. For example, a way of splitting X into two parts can consist in choosing a random R and splitting X into $(X \oplus R)$ and R.

2.3 The "Transformed Masking" Method

The "Transformed Masking" Method was introduced in [2] by M.-L. Akkar and C. Giraud. The basic idea is to perform all the computation that all the data are XORed with a random mask. By using suitably modified tables (for instance S-Boxes in the case of DES), it is possible to have the output of a round masked by exactly the same mask that masked the input. The computation is thus divided into two main steps: the first one consists in generating the modified tables and the second one consists in applying the usual computation using these modified tables (the initial input being masked before starting the computation and the final output being unmasked after the computation).

3 Attack on the Duplication Method

3.1 Example: Second Order DPA on DES

In what follows, we suppose that two bits b_1 and b_2, appearing during the computation, are such that $b_1 \oplus b_2$ equals the value b of the first output of the first S-Box in the first DES round. The attacker performs the following steps:

1. Record the consumption curves C_i corresponding to N different inputs E_i ($1 \leq i \leq N$). For instance $N = 1000$.
2. The attacker guesses the interval δ between the instant corresponding to the treatment of b_1 and the instant corresponding to the treatment of b_2. Each curve C_i is then replaced by $C_{i,\delta}$, which is the difference between C_i and (C_i translated by δ). He then computes the mean curve CM_δ of the N curves $C_{i,\delta}$.
3. The attacker guesses the 6 bits of the key on which the value of b depends. From these 6 key bits, he computes for each E_i the expected value for b. he then computes the mean curve $\mathrm{CM'}_\delta$ of all the $C_{i,\delta}$ such that the expected b equals 0, and $\mathrm{CM''}_\delta$ the mean curve of all the other $C_{i,\delta}$
4. If $\mathrm{CM'}_\delta$ and $\mathrm{CM''}_\delta$ do not show any appreciable difference, go back to 3 with another choice for the 6 key bits.
5. If no choice for the 6 key bits was satisfactory, go back to 2, with another choice for δ.
6. Iterate the steps 2, 3, 4, 5 with two bits whose "exclusive-or" comes from the second S-Box, the third S-Box, ..., until the eighth S-Box.
7. Find the 8 remaining key bits by exhaustive search.

3.2 The Attack in Practice

As specified in the original paper [10], it is clear that the n-th duplication is vulnerable to an n-th order DPA attack. An important point is to notice that if the method is not carefully implemented, it will be easily detected on the consumption curve, just by identifying n repetitive parts in the calculus. In this case, it would be easy for the attacker to just superpose the different parts of the curves (in a constant, or proportional to $\log(n)$, time, but not exponential in n).
Moreover, in certain scenarios, the attacker has full access to the very details of the implementation. In particular, for high-level security certifications (ITSEC, Common Criteria), it is assumed that the attacker knows the contents of the smartcard ROM.

4 Attack on the Transformed Masking Method

4.1 DES: Second Order DPA

4.1.1 Usual Second Order DPA: For the DES algorithm, the input of a round is masked with a 64 bits value $R = R_{0-31}||R_{32-63}$ divided in two independent masks of 32 bits each. The modified S-boxes S' are the following (where S are the original ones).

$$S'(X) = S(X \oplus EP(R_{32-63})) \oplus P^{-1}(R_{0-31} \oplus R_{32-64})$$

Where EP represents the Expansion Permutation, and P^{-1} the inverse of the P permutation after the S-Boxes. We can see that using this formula the output mask of the value at the end of a DES round is nearly R. To get exactly the R masked value, the left part of the value has to be remasked with $R_{0-31} \oplus R_{32-64}$.
It is clear, like noticed in the article, than this countermeasure is subject to a second-order DPA attack. Indeed, the real output of the S-boxes is correlated to the masked value and the value R ; so getting the electrical trace of these two values one can combine them and get a trace on which will work a classical DPA attack. In order to perform efficiently such an attack, without need of n^2 point like in the general attack, the attacker should get precise information about the implementation of the algorithm: he should know precisely where the interesting values are manipulated.

4.1.2 Superposition Attack: In this section we will present a new kind of DPA attack. In theory it is a second-order DPA attack; but in practice it is nearly as simple as an usual DPA attack. The idea is the following: in a second order DPA the most difficult thing is to localize the time where the precise needed values are manipulated. On the contrary localizing a whole DES round is often quite easy. So instead of correlating precise part of the consumption traces we will just correlate the whole trace of the first and the last round. With these method one can notices than at one moment we will have the trace consumption T of the following value which is the output S-Boxes values:

$$\begin{aligned} T &= (S'(E(R_{15}) \oplus K_{16}) \oplus R') \oplus (S'(E(R_1) \oplus K_1) \oplus R') \\ &= \quad S'(E(R_{15}) \oplus K_{16}) \quad \oplus \quad S'(E(R_1) \oplus K_1) \end{aligned}$$

Where R' is the right part of the mask permutated by the expansion permutation. One can notice that the T value does not depend of the random masking value and than R_1 and R_{15}[1] are often known. Considering this, it is easy to sea that performing a guess on the 2×6 bits of the subkey of the first and last round, it is possible to guess the XORed value of the output of the S-Boxes of the first and last round. After that once can perform an usual DPA-type attack attacker and find the values of the different sub-keys of K_1 and K_{16}. Due to redundancy of the key-bits one can moreover check the coherency of the results: indeed with such an attack one will find $2 \times 6 \times 8 = 96 >> 56$ bits for the key. The detailed algorithm is the following:

- Correlate (usually an addition or subtraction of the curves) the first and last round traces.
- For All the messages M, For the S-box $j = 1..8$
- For k=0 to 63, For l=0 to 63
- Separate the Messages , considering one bit of the XOR values of the output of the j^{th} Sbox (round 1 and 16) for the message M considering that the subkey of the S-Box j of the first round is k, and the subkey of the S-Box j of the last round is l.
- Average and subtract the separated curves.
- Choose the value k, l where the greatest peak appear.
- Check the coherency of the keybits found.

A cautionary look of the attack could convince the reader that any error of one bit on the guess of K_1 or K_{16} eliminate all the correlation. Comparing to an usual second order DPA attack, even if this attack require the analyze of $2^{12} = 4096$ possibilities, it has the advantage not to need

[1] R_{15} can be deduced from the output applying the inverse of the final permutation

a precise knowing of the code. And from a complexity point of view it increases by a constant factor ($2^6 = 64$) the amount of time and memory needed for the attacker and not by a linear factor.

4.1.3 Conclusion:

The superposition attack, even if it is a theoretical second order attack is very efficient in practice. Therefore to use transformed masking method, one must use different masks at each step of the algorithm. This idea have been developed and adapted to produce the protection described in this article.

4.2 AES

For the AES, the countermeasure is nearly the same than in DES. The only difference is that no transformed tables are used for the non-linear part of the AES (the inversion in the field GF(256)) but the same table with a multiplicative mask. The distributivity of the multiplication over XOR (addition in the field) is used. So from an additive mask it is easy, without unmasking the value, to switch to a multiplicative one, to go through the Sboxes and to get back to an the mask.

4.2.1 Usual Second Order DPA:

For AES it is exactly the same than in the DES transformed masking method. Correlating the masked value and the mask allow an effective attack against this method.

4.2.2 The "Zero" problem:

Because a multiplicative mask is used during the inversion, one can see that if the inverted value is zero -and this value just depend of 8 bits of the key in the first and last round- then whatever is the masking value, the inverted value will be unmasked. Therefore if someone is able to detect in the consumption trace that the value is zero instead of a random masked value, one will be able to break such an implementation. Of course probabilistic tools such as variance analysis are devoted to such analysis.

4.2.3 Superposition Method:

As in the DES, one can say that using the same superposition method it would be possible to find the key 16 bits by 16 bits superposing the first and last round of AES because these are using the same mask. Unfortunately after the last round a last subkey is added to the output of the round. So the attacker need at least to guess 8 more bits of the key. It increase the attacker amount of work to 24 bits for each Sbox. In theory it is not a quadratic attack in the number of samples but in practice it is not so easy to perform more than 16 billions manipulation of the curve for each tables and each message.

4.2.4 Conclusion: Judging by these attacks we can consider that the adaptive mask countermeasure on AES is not efficient even against some simpler attack than second order ones.

5 Unique Masking Method Principle

We have seen that the actual countermeasure against DPA are intrinsically vulnerable to high order DPA. Often the order of vulnerability is two, and even when it is theoretically more; practically it is one or two. In the next section we will present a method to protect the DES that seem to be efficient against any order DPA attacks. We will first describe the elementary steps of the method for after see how to construct a complete secure DES and why it seems to be secure.

5.1 Masked Rounds

Given any 32 bits value α we will define two new functions $\tilde{S}_1$ and $\tilde{S}_2$ based on the Sboxes function S.

$$\begin{cases} \forall x \in \{0,1\}^{48}\ \tilde{S}_1(x) = S(x \oplus E(\alpha)) \\ \forall x \in \{0,1\}^{48}\ \tilde{S}_2(x) = S(x) \oplus P^{-1}(\alpha) \end{cases}$$

where E is the expansion permutation and P^{-1} is the inverse of the permutation after the Sboxes.
We define f_{K_i} to be the composition of E, the XOR of the i^{th} round subkey K_i the Sboxes and the permutation P. We then define $\tilde{f}_{1,K_i}$ and $\tilde{f}_{2,K_i}$ by replacing S by $\tilde{S}_1$ and $\tilde{S}_2$ in f.

Remark We can see that $\tilde{f}_1$ gives an unmasked value from a α-masked value and that, $\tilde{f}_2$ gives a α-masked result from an unmasked one.
Using the function f, $\tilde{f}_1$ and $\tilde{f}_2$ one can obtain 5 different rounds using masked/unmasked values. The figure 1 represents these five different rounds. The plain fill represents the unmasked value and the dashed fill represents masked values.
The following automata (cf fig. 2) shows how these rounds are compatible with each other. The input states are the rounds where the input is unmasked (A and B) and the output states are the one where the output of the rounds are unmasked (A and E).

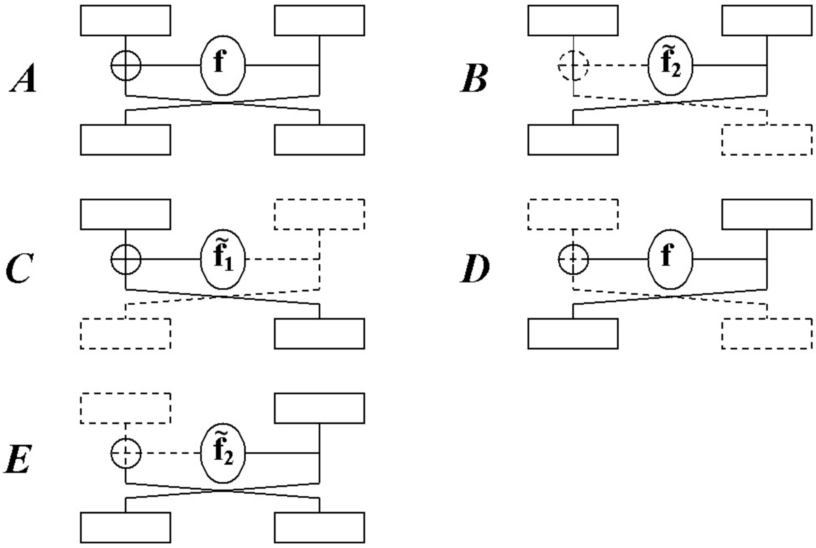

Fig. 1. Masked rounds of DES

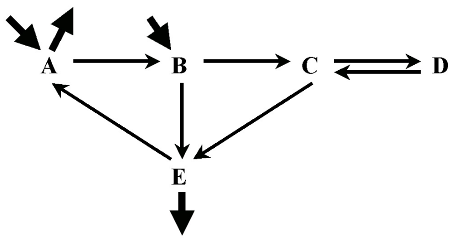

Fig. 2. Combination of the rounds

5.2 Complete DES with Masked Rounds

It is easy to see that one could obtain a 16 round complete DES with these requirements. $IP - BCDCDCEBCDCDCDCE - FP$ represents a correct example (IP represents the initial permutation of DES and FP the final one).

5.3 Security Requirements

In all this section we will consider that the modified Sboxes are already constructed and that the mask α changes at each DES computation.

The first step is to analyze in the DES of how many key bits depends the bits of the data at each round. This simple analyze is summarized in the figure 3. We have also considered that the clear and the cipher were known, explaining the symmetry of the figure.

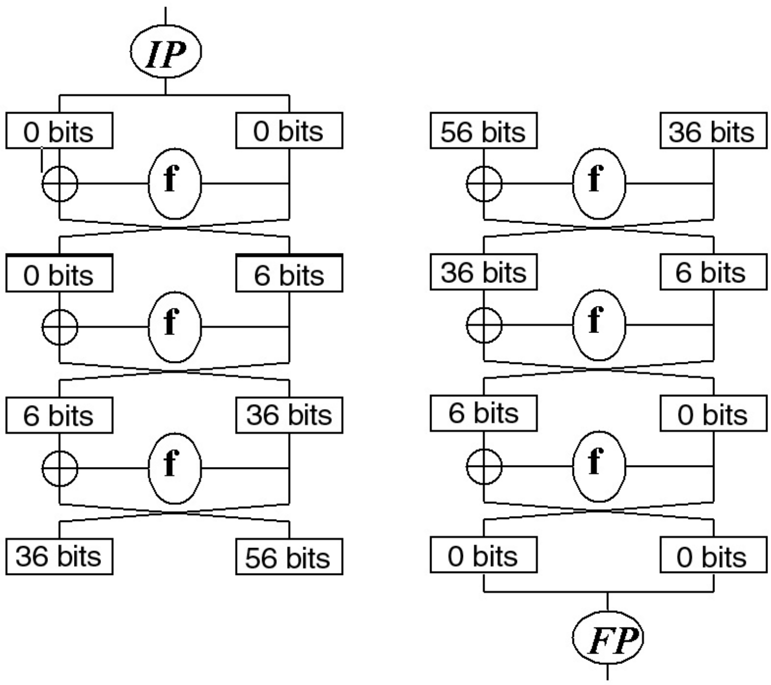

Fig. 3. Number of key bits / bits of data

To get a correct security we have considered that the critical data are the one where the bits are dependant of less than 36[2] bits of the key. So we can see that only two parts have to be protected: the one connecting R_2 and L_3 and the one connecting R_{15} and L_{16}. We define as usual L_i (respectively R_i) as the left part (respectively the right part) of the message at the end of the i^{th} round. Of course the one depending of none bits of the keys have not to be protected.
Therefore these values must be masked and oblige the first three rounds to be of the form:

$$BCD \text{ or } BCE$$

The last three rounds must be of the form:

$$BCE \text{ or } DCE$$

Taking in account these imperatives

$$IP - BCDCDCEBCDCDCDCE - FP$$

is -for example- a good combination.

5.4 Resistance to DPA

5.4.1 Classical DPA: This countermeasure clearly protect the DES against DPA of order one. Indeed all the value depending of less than 36 bits of the key are masked by a random mask which is used only once.

5.4.2 Enhanced Attacks: First we have to notice that this countermeasure is vulnerable against the superposition method guessing 12 bits of the key. Indeed the same mask is used in the first and last round of the DES. So to counteract this attack we will from know consider that there's two different masks α_1 and α_2 which are used in the first and last round of DES. It is easy to see that the proposed combination of round permit at the 7^{th} and 8^{th} round to switch from α_1 to α_2 because of the structure of E-round/B-round which leave their output/input unmasked. With evident notations we can get the following example of DES:

$$IP - B_{\alpha_1}C_{\alpha_1}D_{\alpha_1}C_{\alpha_1}D_{\alpha_1}C_{\alpha_1}E_{\alpha_1}B_{\alpha_2}C_{\alpha_2}D_{\alpha_2}C_{\alpha_2}D_{\alpha_2}C_{\alpha_2}D_{\alpha_2}C_{\alpha_2}E_{\alpha_2} - FP$$

Let now consider n-th order DPA attack. The idea is to correlate several value to get the consumption of an important value. For us an important

[2] If we consider that a curve contains 128 8 bits-samples, 36 bits represents an amount of 2 Tb of memory needed

value is consider to be a value which could be guessed with less than 36 bit of the key. But we have seen that all these value are masked. Moreover the mask appear only once in all the calculus[3], so even with high order correlation it is impossible to get any information about the masked value

5.5 Variation

- If we want the mask never to appear several times (even on values depending on more than 36 bits of the key) one can use the following combination instead of the proposed one:

$$IP - B_{\alpha_1} C_{\alpha_1} E_{\alpha_1} AAAAAAAAAAB_{\alpha_2} C_{\alpha_2} E_{\alpha_2} - FP$$

- For paranoid people it is even possible to add two new masks and to mask every values depending on less than 56 bits of the key.
- This method is modular: if one uses a protocol where the input or the output are not known, one can eliminate the associated mask.

6 Effective Construction of the Modified S-Boxes

In this section algorithms will be described using pseudo c-code.

6.1 Principle

It is easy to see that the following operation must be performed securely in order to construct the Sboxes $\tilde{S}_1$.

- Generate a random α.
- Perform a permutation on α (permutation P^{-1}).
- XOR a value ($P^{-1}(\alpha)$) to a table.

For the construction of $\tilde{S}_2$, we need to:

- Recuperate α because it is the same than in $\tilde{S}_1$.
- Permutate it ($E(\alpha)$).
- XOR to a table containing (1..63).

Of course securely means that all these operations must be done without giving any information about the consumption of α at any order (1,2 ...).

[3] We remind the reader that we have considered that the tables are already constructed. This part will be analyzed in the next section

6.2 Generation of a Random Number: for example 64 bits

We consider that we have access to a 64 bytes array t and to a random generator (for example a generator of bytes). We can proceed like the following:

- for(i=0..63) { t[i]=rand()%2 }
- for(i=0..63) { swap(t[i],t[rand%64]) }

With this this method one can see that we get in memory a 64 bits random value and that an attacker just know the hamming weight of α (if he can perform an SPA attack). For this we have considered that the attacker could not in one shot determinate what is the array entry addressed when we swap the entries ; hypothesis which looks quite reasonable.

Variant 1: To save time and memory we can imagine the following method which is much faster and does not look too weak. We will get 16 4-bits values in a 16 bytes array:

- for(i=0..16) { t[i]=rand() }
- for(i=0..16) { swap(t[i] AND 7,t[rand%16] AND 7) }

Indeed we can consider that the 4 bits of high weight will strongly influence the consumption.

Variant 2: This other method produces and 8 bytes random array. It is faster but less secure.

- for(i=0..8) { t[i]=rand() }
- for(i=0..16) { t[rand()%8] XOR= rand() }

6.3 Permutation

Classically it can be done bit per bit randomly. Against it only allow the attacker to get the hamming weight of the permuted value.
To speed up and have a memory gain, one could perform randomly the permutation quartet per quartet or even byte per byte. An idea could be to add some dummy values and perform the permutation. The dummy values would just not be considered after the permutation time.

6.4 XOR

Here a general method could be to XOR the value bit per bit in a random order to the table. Once again many compromise are possible to perform the XOR: do it byte per byte, add dummy values ...

6.5 Practical Considerations

The usual Sboxes are using 256 bytes. We need them but they could be stored in ROM. For the additional tables we need to store them in RAM. In the normal security method (two masks α_1 and α_2) we need to store 4 new tables. So the total requirement in RAM is of 1024 bytes.
We have seen that the construction of the Sboxes could be performed quite securely. Of course the most secure method is very slow and will really slow down the DES execution and use a lot of memory. The idea was just to show that it was theoretically possible to build the table without filtering any information[4] with a reasonable model of security[5] But we have also seen that it is possible to increase the speed and decrease the memory without loosing too much security.

Lets now have a look at how could be applied our countermeasure to the AES algorithm. Due to the higher number of tables (more than 16 instead of 8) and because they are bigger (8→8 bits instead of 6→4) compared to DES, our countermeasure would require about 8 Kb (or 16 Kb for a high security level) of RAM, a size which is too big for usual smartcards. Some simplifications -which would unfortunately decrease the level of security- are therefore necessary to apply our countermeasure to AES implementation.

7 Real implementation on the DES algorithm

A real implementation of this method have been completed on an ST19 component. It includes the following features described in the last sections:

- SPA protections: Randomization and masking method for the permutations and the manipulation of the key (permutations, Sboxes access...).
- DPA protection: HO-DPA Protection of the first and last three rounds of the DES.
- S-Boxes constructions is done bit per bit with bit per bit randomization while computing the masking value.
- DFA Protection: multiple computation, coherence checking ...

With all this features we get an implementation with:

- 3 KB of ROM code.

[4] But the hamming weight of the value
[5] The attacker is not able to read the exact memory access in one shot.

- 81 bytes of RAM and 668 bytes of extended RAM
- An execution time of 38 ms at 10 Mhz.

This implementation have been submitted to our internal SPA/DPA/DFA laboratory which have tried to attack it without success.

8 Conclusion

Opposed to other proposed countermeasures, the unique masking method presents the following advantages:

- It is actually the only protection known against high-order DPA.
- The core of the DES is exactly the same than ordinary; so one can use with very light modification its implementation just adding the Sbox generation routine.
- The important values are masked with a unique mask which never appear in the DES computation. For example with the transformed masking method the mask were appearing often (for a first mask at the whole beginning and at each rounds). Here one do not even have to mask the entry or unmask the output.
- The only part where the mask is appearing (but it could be randomly and bit per bit) does not depend neither of the key and neither of the message. Therefore the security is totally focused at this point.
- This method is very flexible and modular without important changes in the code: it could even be a compilation parameter to determine which level of security one wants.
- A real implementation have been performed proving the feasibility of this countermeasure in reasonable time (less than 40ms with full protections).

References

1. 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.
2. 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.
3. 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`

4. 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`
5. 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.
6. 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.
7. 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`
8. J. Daemen, M. Peters, G. Van Assche, *Bitslice Ciphers and Power Analysis Attacks*. In Proceedings of FSE'2000, LNCS 1978, Springer-Verlag, 2000.
9. L. Goubin, J. Patarin, *Procédé de sécurisation d'un ensemble électronique de cryptographie à clé secrète contre les attaques par analyse physique*. European Patent, SchlumbergerSema, February 4th, 1999, Publication Number: 2789535.
10. 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.
11. 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`
12. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*. In Proceedings of CRYPTO'99, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.
13. 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.
14. 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`
15. 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.
16. 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.