

# Computational Improvements to Differential Side Channel Analysis

David Naccache<sup>a,b,1,2</sup>, Michael Tunstall<sup>b</sup> and Claire Whelan<sup>c,3</sup>

<sup>a</sup> *Gemplus Card International*

<sup>b</sup> *Royal Holloway, University of London*

<sup>c</sup> *Dublin City University*

**Abstract.** The process of performing a Side Channel Attack is generally a computationally intensive task. By employing a number of simple optimisations the data analysis phase of the attack can be greatly improved. In this paper we will describe some of these improvements and show in the context of DES when attacked using Kocher's classic DPA [1], that a 97% reduction in data processing can be achieved.

**Keywords.** Side Channel Attacks, Differential Power Analysis, The Data Encryption Standard (DES)

## 1. Introduction

Side Channel Analysis (SCA) is an active area of research in the security community and in particular, the smart card industry today. The interest in these attacks is due to the fact that they are highly effective and difficult to defend against. The side channel acquired can be the power consumption of the chip, the electromagnetic emanations or even execution time. Differential SCA (DSCA) is a specific type of SCA in which the attacker captures numerous executions of the target device performing an operation of interest<sup>1</sup>. After acquiring the signals a data analysis phase follows, where statistical tools are applied. This ultimately results in the retrieval of sensitive data.

Vast efforts have been made to develop next generation SCA attacks, catering to the cryptosystem [1,2,3], or the device [4,5] in question, or as a response to proposed countermeasures aiming to deter SCA [6,2,7,8,9,10]. However, to the authors' knowledge, the literature is devoid of addressing the computational complexity of these attacks. In actual fact the process of performing a differential attack is a lengthy process:

- Data capture can take a long period of time. This depends on a variety of practical factors, such as the duration of the monitored process, the oscilloscope→PC trans-

---

<sup>1</sup>David Naccache, Gemplus Card International, Applied Research and Security Centre, 34 rue Guynemer, Issy-les-Moulineaux, F-92447, France (phone: +33 616 598349, e-mail: david.naccache@gemplus.com).

<sup>2</sup>Michael Tunstall, Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK (phone: +44 1784 443093, e-mail: m.j.tunstall@rhul.ac.uk, david.naccache@rhul.ac.uk).

<sup>3</sup>Claire Whelan, School of Computing, Dublin City University, Ballymun, Dublin 9, Ireland (phone: +353-1-7005616, e-mail: cwhelan@computing.dcu.ie).

<sup>1</sup>This is limited to power consumption and electromagnetic acquisitions.

fer rate, the equipment's sampling frequency, the number of curves necessary to overpower the target's signal to noise ratio *etc.*

- The processing of the captured information requires that the data is manipulated in various ways to derive information on the key used.

This is a tedious and time consuming process, that the present work seeks to improve on without affecting the attack's precision or probability of success. In this paper we will present a number of computational improvements to DSCA. These shortcuts will be described in terms of the notorious Differential Power Analysis (DPA) [1], which is most popular in the literature. The methods described are applicable to other forms of DSCA, such as Differential Electromagnetic Analysis (DEMA) [11,12], where once the acquisition stage of the attack is complete, the data analysis phase is exactly the same as DPA. As an illustrative example, assuming that a classic DPA (*i.e* as in [1]) of a given device requires  $\simeq 64,000$  power curve operations. The techniques described in this paper would obtain exactly the same results by spending only 3% of the computational effort (namely  $\simeq 2100$  operations).

The paper is organised as follows: Section 2 establishes the notations used throughout the paper. Section 3 recalls the general principles of differential side channel attacks. We will then evaluate the number of calculations that this involves, so as to establish a benchmark against which we will measure our optimisations. Section 4 contains the differential attack optimisations. Section 5 presents the results, and we conclude in section 6.

## 2. Preliminaries and Notations

We introduce the following notation to track our plaintext segments through the S-boxes.

- The algorithm will be executed  $N$  times. This will produce  $N$  current consumption waveforms (also referred to as *power traces*)  $w_i$  and  $N$  plaintext-ciphertext pairs  $(P_i, C_i)$  where  $1 \leq i \leq N$ .
- The messages given to the algorithm are assumed to be random, unless otherwise stated.
- The algorithm is assumed to be a naive implementation that has no side-channel countermeasures, as a discussion on attack strategies is beyond the scope of this paper.
- We denote the S-boxes by  $\ell_j$  where  $1 \leq j \leq N_S$  and  $N_S$  is the number of S-boxes in the algorithm. For example,  $N_S = 8$  for DES and 10 for AES.
- Each S-box will have as input, both subsets of  $P_i$  and the key  $K$ . For an  $m$  bit S-box, we denote the partial plaintext and partial key by  $P_{i,j}$  and  $K_j$  where  $P_{i,j}$  corresponds to the subset of  $P_i$  which is entered into S-box  $j$  and  $K_j$  corresponds to the key hypothesis  $K_j = n$  for S-box  $j$ . Note that both  $P_{i,j}$  and  $K_j$  can have the value  $n$  where  $0 \leq n \leq 2^m - 1$ .
- For each partial key  $K_j$  a hypothesis will be tested. In order to test this hypothesis,  $2^m$  DPA waveforms will be created. We denote this set of waveforms by  $\Delta_n$  where the  $n^{\text{th}}$  key hypothesis produces the differential trace  $\Delta_n$ .
- We will construct 'representative curves'  $W_n$  where there are  $2^m$  possible curves of this form. We define  $W_n$  by

$$W_n = \sum_{i=1}^N \chi_i w_i \quad \text{where} \quad \chi_i = \begin{cases} 1 & \text{if } P_{i,j} = n \\ 0 & \text{otherwise} \end{cases}$$

$W_n$  will be described further in section 4.2.

- The operations on waveforms, such as the addition of two curves, are ordinary pointwise addition functions.

### 3. Generic Differential Power Analysis

DPA can be performed on any algorithm in which an intermediate operation of the form  $\beta = S(\alpha \oplus k)$  is calculated, where  $\alpha$  is known and  $k$  is the secret key (or some segment of the key). The function  $S()$  is a non-linear function, usually a substitution table (referred to as an S-box), which produces an intermediate output value  $\beta$ .

The process of performing the attack initially involves running the target device  $N$  times with  $N$  random plaintext values  $P_i$ , where  $1 \leq i \leq N$ . The encryption of  $P_i$  under the secret key  $K$  to produce the corresponding ciphertext  $C_i$ , will result in  $N$  current consumption waveforms  $w_i$ . These waveforms are captured with an oscilloscope, and sent to a PC for analysis and processing.

To find a given partial key  $K_j$ , the output produced from the S-box  $\ell_j$  when given  $K_j$  and all  $P_{i,j}$  will be used to categorise the current consumption waveforms. A single output bit  $b$  from  $\ell_j$  is used for this categorisation. For all possible hypotheses, *i.e.*  $K_j = 0 \dots K_j = 2^m - 1$ , and each partial message  $P_{i,j}$ ,  $b$  will classify whether waveform  $w_i$  is a member of one of two possible sets. The first set  $S_0$  will contain all the waveforms where  $b$  is equal to zero, and the second set  $S_1$  will contain all the remaining waveforms, *i.e.* where the output bit  $b$  is equal to one.

For each hypothesis, a differential trace  $\Delta_n$  is calculated by finding the average of each set and then subtracting the resulting values from each other.

$$\Delta_n = \frac{\sum_{w_i \in S_0} w_i}{|S_0|} - \frac{\sum_{w_i \in S_1} w_i}{|S_1|} \quad (1)$$

The DPA waveform with the highest peak will validate a hypothesis for  $K_j$ , *i.e.*  $K_j = n$  corresponds to the  $\Delta_n$  featuring a maximum amplitude. For a single DPA waveform this involves  $N$  additions (as there will be an average of  $\frac{N}{2}$  elements in each set), two divisions and one subtraction, to create the differential  $\Delta_n$ . Therefore, for all hypotheses the total number of operations to calculate all DPA waveforms for one S-box is  $2^m \times (N + 3)$ .

### 4. Fast Differential Power Analysis

By introducing a number of basic pre-calculations we can accelerate DPA. We will optimise DPA in terms of the calculations performed in the statistical analysis phase of the attack, *i.e.* the calculation of the DPA waveforms. To calculate one differential  $\Delta_n$ , the operations involved are basic pointwise addition, subtraction and division of curves. We target our optimisations in terms of minimising the number of additions of current con-

sumption curves, as this is the most utilised operation. We will treat the division (calculation of mean) and subtraction operations as constant, as these are fundamental operations in the calculation of the differential  $\Delta_n$ , and cannot be enhanced.

In the following section we will detail a number of optimisations which can greatly reduce the number of additions involved in performing a DPA. We will measure these optimisations in terms of the number of calculations required to calculate the differential traces  $\Delta_n$ , for all hypotheses.

#### 4.1. The Global Sum

The simplest optimisation involves the calculation of the global sum  $G$ . The global sum is nothing but the summation of all the current consumption waveforms that have been acquired.

$$G = \sum_{i=1}^N w_i$$

The calculation of the DPA waveform now only involves the summation of a single set, as opposed to two sets when using equation (1).

$$\Delta_n = \frac{G - S_{\text{least}}}{N - |S_{\text{least}}|} - \frac{S_{\text{least}}}{|S_{\text{least}}|} \quad (2)$$

$S_{\text{least}}$  represents the set with the least number of elements, i.e:

$$S_{\text{least}} = \begin{cases} S_0 & \text{when } |S_0| \leq |S_1| \\ S_1 & \text{when } |S_0| > |S_1| \end{cases}$$

The expected number of additions required to generate  $S_{\text{least}}$  can be calculated using:

$$\begin{aligned} E(X) &= \sum_{i=0}^N i \Pr[X = i] \\ &= \sum_{i=0}^{\frac{N}{2}} i \binom{N}{i} \left(\frac{1}{2}\right)^N + \sum_{i=\frac{N}{2}+1}^N (N-i) \binom{N}{i} \left(\frac{1}{2}\right)^N \end{aligned}$$

If, for example, 1000 acquisitions were taken this would result in an expected number of additions per hypothesis of 487. This is an improvement over the case where a set is chosen arbitrarily, when the expected number of additions will be 998.

The cost of precalculating  $G$  for a single hypothesis is obviously not worthwhile. However for  $2^m$  hypotheses significant savings are realised, as the maximum number of operations to calculate all DPA waveforms for a single S-box is  $(N-1) + 2^m \times (\frac{N}{2} + 3)$ .

Note, however, that separate pre-computation of  $G$  is not mandatory. The trick here consists in computing a first hypothesis just as in the original version of DPA and then summing the two resulting average curves to get  $G$ , thereby allowing the complexity of the next  $2^m - 1$  hypotheses calculations to be reduced calculations at no extra cost.

In the remainder of this paper,  $S_{\text{least}}$  will no longer be used. This is due to the optimisation described in the following section, which will change the way in which the raw data is distributed. In subsequent sections the differential trace  $\Delta_n$  will be calculated by subtracting either  $S_0$  or  $S_1$  from  $G$ , where the choice of  $S$  is completely arbitrary.

#### 4.2. Formation of Waveform Equivalence Classes

The input to each S-box will consist of partial bits  $P_{i,j}$  of the plaintext  $P_i$  and partial bits  $K_j$  of the key  $K$ . Concentrating on  $P_{i,j}$ , there are only  $2^m$  possible values which can enter  $\ell_j$ , yet we are dealing with  $N$  waveforms. Therefore, a number of the waveforms will have the same value for  $P_{i,j}$  and thus can be treated in the same manner (*i.e.* they form an equivalence class<sup>2</sup>). We define  $W_n$  for  $\ell_j$  as:

$$W_n = \sum_{i=1}^N \chi_i w_i \quad \text{where} \quad \chi_i = \begin{cases} 1 & \text{if } P_{i,j} = n \\ 0 & \text{otherwise} \end{cases}$$

This will produce  $2^m$  representative curves. This can be calculated *on-the-fly*. The partitioning of the power curves according to the key hypothesis will now result in  $2^{m-1}$  curves in each set. The differential trace will be calculated as

$$\Delta_n = \frac{G - S_0}{N - |S_0|} - \frac{S_0}{|S_0|}$$

where  $S_0$  will now contain exactly  $2^{m-1}$  elements<sup>3</sup>. To generate a single DPA waveform this will result in  $2^{m-1} - 1$  additions. For all hypotheses a total of  $2^m \times (2^{m-1} - 1)$  additions will be required to calculate all the DPA waveforms for one S-box. Pre-calculation involves the formation of the representative curves  $W_n$  and the global sum  $G$ . Since we can construct  $G$  as a function of  $W_n$ :

$$G = \sum_{n=1}^{2^m} W_n$$

The total number of operations that will be incurred in generating all DPA waveforms, is hence:

$$\begin{aligned} \text{Total Calculations} &= 2^m \left( \frac{N}{2^m} - 1 \right) + (2^m - 1) + 2^m (2^{m-1} - 1) \\ &= N - 1 + 2^m (2^{m-1} - 1) \end{aligned}$$

where for  $W_0 \dots W_{2^m}$ ,  $\approx \frac{N}{2^m} - 1$  additions will make up each  $W_n$ , additional  $2^m - 1$  summations will be required to form the global sum  $G$  and  $2^{m-1} - 1$  additions will be required to generate each DPA waveforms for  $2^m$  key hypotheses.

<sup>2</sup>Note the curves form an equivalence class for each  $\ell_j$ , *i.e.* for  $\ell_0$  the representative curves will be formed in a particular way, for  $\ell_1$  they will be formed in a different way, *etc.*

<sup>3</sup>If the case arises where a value for  $W_n$  does not occur, this may create a bias. This is the only situation where  $|S_0| \neq 2^{m-1}$ .

### 4.3. Curve Combining

Pre-calculating certain curve combinations enables groups of curves that occur in the same set for one hypothesis, to be recycled in subsequent hypothesis testing. Since we are dealing with  $2^m$  representative curves, precomputing all possible  $2^m!$  curve combinations is obviously infeasible. Therefore we propose to partition the curves into groups of size  $x$  and precompute the different possible combinations for each group.

For example, for  $x = 2$ , adjacent curves are summed. We define each pair as the value  $W_{2n,2n+1}$  where  $n = 0 \dots 2^{m-1}$  and  $n$  is incremented by two.

$$W_{2n,2n+1} = W_{2n} + W_{2n+1}$$

The use of the combined curves  $W_{2n,2n+1}$  in the evaluation of the set  $S_0$ , results in three possible scenarios for each pair:

1. The pair occurs *i.e.*  $W_{2n} + W_{2n+1}$  appears in  $S_0$ . The probability of this occurring is  $\frac{1}{4}$ . In this case, one additional summation must be performed.
2. The pair does not appear in  $S_0$ , *i.e.* the pair is in  $G$ . The probability of this happening is also  $\frac{1}{4}$ . In this case no action is performed.
3. The two curves  $W_{2n}$  and  $W_{2n+1}$  appear in two separate sets. There is a higher probability of this happening, *i.e.*  $\frac{1}{2}$ . In this case a single addition is required.

The expected number of additions per  $\Delta_n$  (assuming the global set already exists), is  $\frac{3}{4} \times 2^{m-1}$ . This is because there are  $2^{m-1}$  groups of three elements ( $W_{2n}, W_{2n+1}$  and  $W_{2n,2n+1}$ ), one of which will be used to add to  $S_0$  with probability  $\frac{3}{4}$ .

If  $x = 3$  then the following combinations of curves will be created from  $W_{3n}, W_{3n+1}$  and  $W_{3n+2}$ :

$$W_{3n,3n+1} = W_{3n} + W_{3n+1}$$

$$W_{3n+1,3n+2} = W_{3n+1} + W_{3n+2}$$

$$W_{3n,3n+2} = W_{3n} + W_{3n+2}$$

$$W_{3n,3n+1,3n+2} = W_{3n} + W_{3n+1} + W_{3n+2}$$

Following the same reasoning as above, the number of additions is  $\frac{7}{8} \times \frac{2^m}{3}$ . For all values of  $x$  this can be generalised to:

$$\text{Additions per Hypothesis} = \left( \frac{2^x - 1}{2^x} \right) \left( \left\lfloor \frac{2^m}{x} \right\rfloor - 1 \right) - \frac{2^{2^m \bmod x} - 1}{2^{2^m \bmod x}}$$

This comprises of two expressions as  $x$  will not always divide evenly into  $2^m$ , which will leave  $2^m \bmod x$  elements to be grouped together separately. Note that this is only an approximation of the amount of additions required for each set. In practice the number of additions involved in generating  $\Delta_n$  will depend on the contents of the set, and whether the precomputed combinations appear or not.

The amount of pre-calculation involved (incorporating  $G, W_n$  and the precomputed combinations) is:

$$\text{Pre-Additions} = N - 1 + \left( (2^x - x - 1) \left\lfloor \frac{2^m}{x} \right\rfloor \right) + (2^{2^m \bmod x} - (2^m \bmod x) - 1)$$

Therefore, total pre- and post-calculations, which will result in the production of  $2^m$  DPA waveforms is given by:

$$\text{Total Calculations} = 2^m \times \text{Additions per Hypothesis} + \text{Pre-Additions}$$

Memory requirements are obviously a vital factor, as the more pre-computed values, the more storage they will take up and the more time it will take to load these values into memory, which will effect the overall attack performance. In order to balance the time-memory tradeoff and achieve the optimal attack, we give the following formula to derive for a given value of  $x$ , the memory that is required. The formula allows an attacker to relate the number of precomputed values to their resources.

$$\text{Memory Required} = (2^x - 1) \left\lfloor \frac{2^m}{x} \right\rfloor + (2^{2^m \bmod x} - 1) \quad (3)$$

This value gives the number of representative curves  $W_n$  which need to be stored in memory. Each point in this curve will be stored in a 32-bit word so that no information is lost. The value generated in (3) will be multiplied by  $4 \times$  the size of one acquisition (assuming that the values acquired are byte sized).

#### 4.4. Chosen Plaintext Differential Power Attacks

The pre-calculations previously made for S-box  $\ell_j$ , unfortunately will be redundant for  $\ell_{j+1}$ . This is due to the fact that we will be focusing on a later section of the current consumption waveforms, which when classified according to the partial input  $P_{i,j}$  will fall into different equivalence classes  $W_n$  than before. Therefore for S-box  $\ell_{j+1}$ , the regeneration of the representative curves will be required.

In classical DPA the message given to the algorithm under attack is random. However if we can perform a chosen plaintext attack we can utilise the precalculated  $W_i$  for use in subsequent S-boxes. The simplest case is where input to S-box  $\ell_1$  is the same for  $\ell_1 \dots \ell_{N_S}$ . For example, construct the plaintext so that all plaintext bytes are equal, *i.e.*  $\text{byte}[1] = \text{byte}[2] = \dots = \text{byte}[16]$  in AES. This means that there are 256 possible values for the plaintext. Calculating the DPA waveform for the first S-box will calculate the DPA waveform for all others at the same time, giving sixteen peaks at the points in time in which the sixteen key bytes are being manipulated. Using this method may not always be advantageous as some confusion can arise as to which peak corresponds to which key byte.

Obviously the use of this technique is dependant on the algorithm and the eligibility checks that the plaintext undergoes before it is encrypted. A similar and valid attack is an attack on an implementation of DES, where the plaintext is generated such that the  $P_{i,j}$  entered into  $\ell_{j \bmod 2=0}$  is equal for all even S-boxes, and  $P_{i,j}$  entered into all odd S-boxes  $\ell_{j \bmod 2=1}$  is equal for all odd S-boxes. All the even numbered S-boxes can use the same set of data generated during the pre-calculation for the first S-box. The odd numbered S-boxes also use this data but with a permutation on the value associated with each representative curve. This does not affect the quality of the results produced as each

S-box uses a different permutation. The DPA peak will be at the same level as if a random plaintext was used.

Note that these optimisations are applicable when the attack is concentrating on the first round of a secret key algorithm. If the attack focuses on the last round where the DPA waveforms are related to the ciphertext, these optimisations will be useless as the data can not be controlled.

## 5. Results

We will describe our optimisations in terms of performing DPA on DES, where 1000 acquisitions current consumption acquisitions have been taken. This has been seen experimentally to often be sufficient to determine some relationship between the current consumption and the data manipulated. It may be necessary to use more acquisitions, or possible to use less, but 1000 has usually proven to be a good starting point.

Given  $N = 1000$  current consumption waveforms  $w_i$ , the following table details the number of operations that must be performed in the act of generating the differential traces for the key hypotheses.

	Precalculation	Additions per hypothesis	Additions per S-box	Curves Required	Memory Required
Theoretical DPA	-	998	63872	-	-
Optimisation 1: Global Sum	999	487	32167	1	4
Optimisation 2: Equivalence Classes	999	31	2983	65	260
Optimisation 3:					
2 bits	1031	23.3	2519	96	384
3 bits	1083	18	2235	148	592
4 bits	1175	14.1	2075	240	960
5 bits	1322	11.6	2064	387	1548
6 bits	1580	9.8	2207	645	2580
7 bits	2079	8.4	2619	1144	4576
8 bits	2975	7.0	3421	2040	8160
9 bits	4513	6.5	4928	3578	14312
10 bits	7088	5.9	7468	6153	24612

The more bits are grouped together, the more memory is required to conduct the attack. It has been assumed the the current consumption acquisitions consist of a million points where each point is one byte, therefore we will allow for the representative curve to take up four times as much memory as that of a raw data curve.

As shown in the table the best results are obtained when five bits are grouped together at a time. However, the memory requirement for this is 1.5 Gigabytes. The amount of additions when three bits are combined is slightly higher but requires a much more reasonable amount of memory.

The optimal location for the storage of the representative curves is obviously in the computers RAM. This is because the access times are significantly faster than those of

a hard drive, especially as the amount of data being processed is too large for the hard drive to store in its cache.

## 6. Conclusion

In this paper we presented a number of optimisations that can be used with Kocher's original algorithm to significantly reduce the computation time. These optimisations allow an attacker to search for data dependence in a short period of time. This can be used as a preparatory phase to other forms of SCA, such as CPA [13]. These types of treatment can help to reduce false positives by reducing the occurrence of "ghost peaks" as described in [13].

In the example given, the time taken for the processing of all the hypotheses for one six-bit section of the key is reduced by a factor of thirty. In actuality, this time will be further decreased as the acquisitions on the computer's hard drive are only accessed during the construction of the representative curves  $W_n$ . The rest of the processing takes place in RAM.

The ideas expressed in section 4.4 have not been discussed in the example, as the gain for the overall attack depends on how the message can be manipulated. The gain for an attack where the plaintext can be freely manipulated should reduce execution time by a factor of approximately sixty.

In our analysis we assume the resolution of the current consumption acquisitions is one million points. In practice this can vary depending on a number of factors, such as the storage capacity of the oscilloscope, the amount of time spent localising the S-boxes, and the algorithm being attacked. In the case of AES, larger S-boxes are used, and so there will be a greater number of key hypotheses, which will result in an increase in the number of precomputed values to be stored. The worst possible scenario is where the memory requirement becomes unmanageable and pre-computation actually inhibits the attack. There are two approaches that an attacker can employ to combat this situation. In the case where the acquisitions captured are large, one can split the acquisition into smaller sections and perform the respective operations on these sections, and subsequently concatenate the files to construct the full DPA waveform. Alternatively, (3) can be used to determine how much pre-computation is possible with the memory available, allowing an attacker to achieve a maximum benefit from the optimisations described.

## References

- [1] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology - CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–379. Springer Verlag, 1999.
- [2] T. Messerges. Securing the aes finalists against power analysis attacks. In B. Schneier, editor, *Fast Software Encryption - FSE 00*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer Verlag, 2000.
- [3] J. S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In C. K. Koc and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer Verlag, 1999.

- [4] T. Messerges, E. Dabbish, and R. Sloan. Investigations of power analysis attacks on smartcards. In *USENIX Workshop on Smartcard Technology*, 1999.
- [5] S. B. Ors, E. Oswald, and B. Preneel. Power analysis attacks on an fpga - first experimental results. In C. D. Walter, C. K. Koc, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 03*, volume 2779 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.
- [6] L. Goubin and J. Patarin. Des and differential power analysis, the duplication method. In C. K. Koc and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer Verlag, 1999.
- [7] M. L. Akkar and C. Giraud. An implementation of des and aes, secure against some attacks. In D. Naccache C. K. Koc and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 01*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer Verlag, 2001.
- [8] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology - CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer Verlag, 1999.
- [9] L. Goubin. A sound method for switching between boolean and arithmetic masking. In C. K. Koc D. Naccache and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 01*, volume 2162 of *Lecture Notes in Computer Science*, pages 3–15. Springer Verlag, 2001.
- [10] J. S. Coron and A. Tchoulkine. A new algorithm for switching from arithmetic and boolean masking. In C. K. Koc C. D. Walter and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 03*, volume 2779 of *Lecture Notes in Computer Science*, pages 89–97. Springer Verlag, 2003.
- [11] J.J. Quisquater and D. Samyde. A new tool for non-intrusive analysis of smartcards based on electromegnetic emissions, the sema and dema methods. 2001.
- [12] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In C. K. Koc, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 01*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer Verlag, 2001.
- [13] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 04*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Verlag, 2004.