Differential Power Analysis Enhancement with Statistical Preprocessing

Victor Lomné, Amine Dehbaoui, Philippe Maurine, Lionel Torres, Michel Robert

Differential Power Analysis Enhancement with Statistical Preprocessing

Victor Lomné, Amine Dehbaoui, Philippe Maurine, Lionel Torres, Michel Robert

LIRMM, UMR 5506, University Montpellier 2 - CNRS, 161, rue Ada, 34392 Montpellier, France

Email: {firstname.lastname}@lirmm.fr

Abstract—**Differential Power Analysis (DPA) is a powerful Side-Channel Attack (SCA) targeting as well symmetric as asymmetric ciphers. Its principle is based on a statistical treatment of power consumption measurements monitored on an Integrated Circuit (IC) computing cryptographic operations. A lot of works have proposed improvements of the attack, but no one focuses on ordering measurements. Our proposal consists in a statistical preprocessing which ranks measurements in a statistically optimized order to accelerate DPA and reduce the number of required measurements to disclose the key.**

I. Introduction

In past decades, cryptologists have mainly focused on the theoretical security of cryptographic algorithms. But since the 90's, more and more embedded systems, like smart-cards, are used to manage sensible data and therefore integrate cryptosystems. These embedded systems are generally built in Complementary Metal Oxide Semi-conductor (CMOS) technology, and this technology has an interesting property from a cryptanalytic point-of-view : physical leakages emanating from the device are correlated with processed data. Then researchers have focused on exploiting these leakages as a new source of information to disclose the secret key used to cipher sensible data.

As a result, different attacks exploiting the computational time (Timing Attack (TA) [1]), the power consumption (Differential Power Analysis (DPA) [2]) or electromagnetic radiations of the device (Differential ElectroMagnetic Analysis (DEMA) [3]) appeared. This class of attacks has been called the *Side-Channel Attacks* (SCA).

In this context, a lot of improvements have been proposed to enhance the original DPA algorithm, like multi-bits DPA [4] [5], Correlation Power Analysis (CPA) [6], Partition Power Analysis (PPA) [7], or Mutual Information Analysis (MIA) [8]. All these enhancements aim at improving the *selection function*, the backbone of the attack, describing the way to correlate physical leakages with processed data. When considering all the proposed improvements of the original DPA, no one focuses on ordering power consumption measurements (also called power consumption curves ($PCCs$)). Indeed, using, at first, $PCCs$ with the most useful informations on processed data should accelerate the attack and decrease the number of $PCCs$ required to guess the key. Then our proposal is a statistical technique which aims at selecting $PCCs$ with the greatest information leakage, and ordering them in a decreasing order of information leakage.

In the scope of the SCA, an international competition has been launched at CHES'08, the *dpacontest* [9], which consists in providing $PCCs$ monitored on a hardware Data Encryption Standard (DES) co-processor, and where the goal of the challengers is to propose the fastest attack to retrieve the secret key. The criterion used to compare the different implementations is the number of $PCCs$ required to retrieve the full key of the DES.

To validate our proposal, we have led multi-bits DPA and CPA on $PCCs$ provided by the dpacontest, first using a *random order*, and secondly using our algorithm which provides a *statistically optimized order*. Results show that our technique reduces significantly the number of $PCCs$ required to guess the key involved in a DES encryption.

The outline of the article is the following : our statistical preprocessing technique is presented in part 2, and the part 3 gives concrete results applied on measurements provided by the *dpacontest*. Finally, part 4 concludes the article.

II. Statistical Preprocessing Technique

A. Intuitive idea

In the rest of the article, the DES will be used as example, because of it is the well-known block cipher and principles of SCA stay the same on others cryptographic algorithms. Moreover, we consider, for convenience, that the adversary is in the case of a known-plaintext attack and tries to guess the round-key 1 of the DES (the remaining 8 bits could be found with a bruteforce attack).

Because of the set of all possible values for the round-key 1 is too big to test all of them, the adversary divides usually the round-key 1 in 8 parts of 6 bits (called here sub-key) and attacks each sub-key independently and sequentially.

In the classical DPA attack, the adversary computes, for each sub-key hypothesis, a differential curve following equation (1) [2]:

$$\Delta_{K_s}[j] = \frac{\sum_{i=1}^{N} D(PTI_i, K_s)T_i[j]}{\sum_{i=1}^{N} D(PTI_i, K_s)} \tag{1}$$

$$-\frac{\sum_{i=1}^{N}(1 - D(PTI_i, K_s))T_i[j]}{\sum_{i=1}^{N}(1 - D(PTI_i, K_s))} = \epsilon_1 - \epsilon_2$$

where K_s is the sub-key hypothesis, $\Delta_{K_s}[j]$ is the j-th sample of the differential curve, N is the number of $PCCs$ used, PTI_i is the i-th Plaintext Input, $T_i[j]$ is the j-th sample

of the PCC and D the decision function ranking $PCCs$ in sets A or B, also called *selection function*.

If the hypothesis of the sub-key is good, a spike will appear at the time index of the differential curve, where the Intermediate Value (IV) is computed.

The number of $PCCs$ necessary to perform the attack depends mainly on the measurements conditions. From (1) and for the spike to be identified

$$\epsilon_1 - \epsilon_2 > \sigma/\sqrt{N} \qquad (2)$$

must hold [10], where σ represents the noise and N the number of necessary $PCCs$. To decrease N, an adversary would like to amplify the difference $\epsilon_1 - \epsilon_2$ in (2).

Intuitively, an intermediate value where the four bits switch will induce the maximum of power consumption than these four bits can consume on the PCC at the time index where the IV is computed. Inversely, an intermediate value where no bit switch will induce the minimum of power consumption than these four bits can consume on the PCC at the same time index.

Thus, if the adversary could choose $PTIs$ in order to obtain only IVs where all the bits switch and others where no one switch, he should obtain $PCCs$ with greater spikes and others with smaller spikes than randomly, and maximize the difference in (2). But our improvement does not concern a chosen-plaintext attack.

Rather than optimizing number of switching bits of IVs, we aim at selecting $PCCs$ with greatest and smallest spikes during the attacked clock cycle. Then, begining the attack by processing first these selected $PCCs$ will increase the difference $\epsilon_1 - \epsilon_2$, and verify (2) with less $PCCs$ than in the case of random selection of $PCCs$.

The only requirement necessary for this idea is to know before launching the preprocessing where is the targeted clock cycle. But applying a Simple Power Analysis (SPA) on one of the measured $PCCs$ often allows to identify the clock cycle matching with the first or last round of the DES. Another way to identify this clock cycle is to perform a DPA with few $PCCs$, and select time indexes where spikes appear (even if these spikes do not match with the good key hypothesis).

Note that two cases have to be considered. The first one concerns implementations where the whole round is computed as the same time (i.e. the data path has a size of 64 bits in the case of the DES). This is the case of most hardware implementations, for instance the co-processor used for the dpacontest. The second case concerns software implementations, especially on 8 bits processors, where the data path is 8 bits sized. Our idea has been applied on co-processors of the first case, that means that the R register is fully updated during the same clock cycle (the 8 sboxes are computed at the same time). So PCC with the greatest spike at the attacked clock cycle will not mean that the whole 4 bits of the 8 parts of the R register switch. But statistically, this PCC has more bits switching than another one where the attacked spike has a value closed to the mean value of all considered spikes. Our idea could be applied on implementations with 8

Algorithm 1 PDF as maximum of the spike
***
Require: N $PCCs$, $tMin$, $tMax$
Ensure: N ranked $PCCs$
 1: initialize $tab1$ as array Nx2
 2: initialize $tab2$ as array Nx1
 3: **for** $i = 1$ to N **do**
 4:     $tab1[i,1] = \max_{t \in \{tMin, tMax\}}(PCC_i[t])$
 5:     $tab1[i,2] = i$
 6: **end for**
 7: sort($tab1$) following the first column
 8: **for** $i = 1$ to $N/2$ **do**
 9:     $tab2[2i - 1, 1] = tab1[i,2]$
10:     $tab2[2i, 1] = tab1[N - i + 1, 2]$
11: **end for**
12: **return** $tab2$
***

bits data path, but we should adapt the principle on software implementations. For instance we could sum the eight spikes where 4 bits values linked to sboxes are computed, and considering this sum rather than the value of one spike.

From this intuitive idea, we propose two techniques to select and rank $PCCs$ with greatest and smallest spikes at the attacked clock cycle. One can look for the maximum of the spike, or for the integral of the parabola drawn by the spike. Formally, considering the maximum or the integral as a random variable called X, we aim at evaluating the probability density function (PDF) of each observation of X. Thus, we select for half observations of X with the greatest PDF, and for half those with the smallest PDF.

B. Algorithm 1 : PDF as maximum of the spike

This first technique ranks $PCCs$ following the decreasing maximum amplitude of the targeted spike. Thus, the algorithm selects PCC with the greatest amplitude as first PCC in the new *statistically optimized order*, then PCC with the smallest amplitude as second PCC in the new order, PCC with the second greatest amplitude will be the third PCC in the new order, etc ... That gives the algorithm 1.

C. Algorithm 2 : PDF as integral of the parabola drawn by the spike

The second technique follows the same algorithm as above, but rather than ranking $PCCs$ following the decreasing maximum amplitude of the targeted spike, we compute the integral of the parabola drawn by the spike. More precisely, for all the considered $PCCs$, we compute the average value of the targeted spike. Then we compute the area of the spike above the horizontal line with ordonnate equal to the computed average. This horizontal line allows to evaluate the noise and static consumption threshold. This technique is described in algorithm 2.

III. CONCRETE RESULTS

The *dpacontest* provides $PCCs$ of a hardware Data Encryption Standard (DES) co-processor. We have used the campaign *secmatv1* composed of 81089 $PCCs$.

Algorithm 2 PDF as integral of the parabola drawn by the spike

---

Require: N $PCCs$, $tMin$, $tMax$
Ensure: N ranked $PCCs$

1: initialize $tab1$ as array Nx2
2: initialize $tab2$ as array Nx1
3: **for** $i = 1$ to N **do**
4:    $tab1[i,1] = mean_{t \in \{tMin,tMax\}}(PCC_i[t])$
5: **end for**
6: $\bar{m} = mean(tab1[:,1])$
7: **for** $i = 1$ to N **do**
8:    **for** $j = tMin$ to $tMax$ **do**
9:      **if** $(PCC_i[j] < \bar{m})\&(PCC_i[j+1] > \bar{m})$ **then**
10:        $t1 = j$
11:      **end if**
12:      **if** $(PCC_i[j] > \bar{m})\&(PCC_i[j+1] < \bar{m})$ **then**
13:        $t2 = j$
14:      **end if**
15:    **end for**
16:    $tab1[i,1] = \int_{tMin}^{tMax}(PCC_i[t])dt - \bar{m} * (PCC_i[t2] - PCC_i[t1])$
17:    $tab1[i,2] = i$
18: **end for**
19: sort($tab1$) following the first column
20: **for** $i = 1$ to $N/2$ **do**
21:    $tab2[2i-1,1] = tab1[i,2]$
22:    $tab2[2i,1] = tab1[N-i+1,2]$
23: **end for**
24: **return** $tab2$

Note that our attacks are performed under the following conditions :

- we apply multi-bit DPA or CPA.
- we guess 6 bits of sub-key at a time
- we regard the Hamming Distance between L15 and L16
- we attack the 16th round of the DES, and we focuse the key search on time indexes 14450 until 14550

To evaluate the efficiency of our proposal, we have led 2 kind of experiments. The first one evaluate the average success rate of the preprocessing methods in comparison with classical attacks on a fixed number of $PCCs$. The second shows that the efficiency of our techniques increases with the number of PCC preprocessed. Note that the computational time for the two techniques is very small, for instance ranking some hundreds of $PCCs$ needs less than 10 seconds in average.

A. *Evaluate the average success rate of the methods*

Following guidelines given in [11], a smart way to compare the effectiveness of different SCA methods is to evaluate the success rate of the attack. Thus we have applied the following experiment 100 times :

- create a random order of 500 $PCCs$ among the whole 81089 $PCCs$,
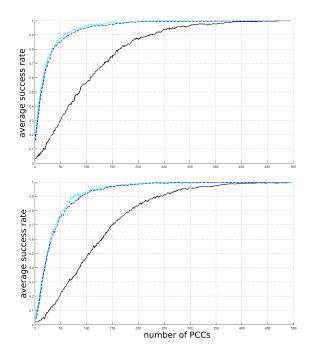- apply DPA and CPA on this random order,

Fig. 1. average success rate of DPA (upper) and CPA (lower), without preprocessing (dark solid line), with method 1 preprocessing (cyan solid line), with method 2 preprocessing (dashed line)

- apply algorithms 1 and 2 on the random order to generate two *statistically optimized orders*,
- apply DPA and CPA on the two new orders,

By this way we have computed the average success rate of each attack (DPA and CPA) and for each method (without preprocessing, applying method 1 before attacking and using method 2 before attacking). Results are drawn on figure 1, the upper graph is the evolution of the average success rate for a DPA attacks, whereas CPA attacks success rates are drawn in the lower graph. On the two graphs the dark and solid line corresponds to the results obtained with classical DPA and CPA (without preprocessing), the cyan and solid line is the evolution of the method 1, whereas the dashed line corresponds to the results of the method 2.

A classical DPA or CPA discloses the full round-key 16 in a bit less than 400 $PCCs$ in average, whereas our preprocessing techniques allow to reach an average success rate of 1 with about 250 $PCCs$. Note that success rates of DPA an CPA are closed, without and with preprocessing.

B. *Efficiency functions on number of $PCCs$*

We would also know, functions on N, the evolution of the effectiveness of our two ranking algorithms. Then we have led DPA and CPA on the naive order, corresponding to the database order as $PCCs$ are provided on the website of the *dpacontest*. The multi-bit DPA discloses the round-key 16 with 260 $PCCs$, whereas the CPA needs 213 $PCCs$ to guess the key.

Then, we have selected the 200 first $PCCs$ of the database order, and we have applied the two algorithms. With the
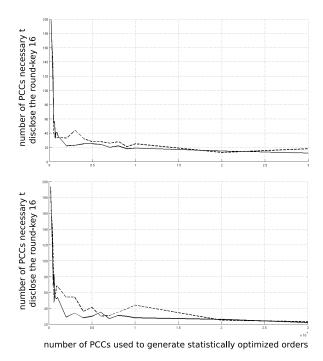
Fig. 2. evolution of the number of $PCCs$ necessary to disclose the key functions on the number of $PCCs$ used to generate the *statistically optimized orders*. DPA (upper) and CPA (lower). Solid lines correspond to method 1 whereas dashed lines to method 2

obtained two new orders, we have performed DPA and CPA. Then we have done the same experiment, but using the 300 first $PCCs$ of the database order, etc ... Figure 2 draws the evolution of the effectiveness of the two techniques functions on the number of $PCCs$ used to compute the *statistically optimized order*.

More precisely, figure 2 summarizes results, the abscissa axis represents the number of $PCCs$ used to generate the two *statistically optimized orders*, whereas the ordonnate axis represents the number of $PCCs$ necessary to disclose the round-key 16. The upper figure represents results for DPA, where the solid line correpsonds to method 1 and dashed line to method 2, whereas the lower figure draws results for CPA.

This experiment shows that the efficiency of our two preprocessing methods increases with the number of $PCCs$ used to generate the *statistically optimized orders*. For instance, whereas the DPA without preprocessing requires 260 $PCCs$ to disclose the key, method 1 applied on 30000 $PCCs$ allows to guess the key with only 12 $PCCs$ applying the same DPA.

IV. CONCLUSION

We have proposed a preprocessing technique for DPA attacks which ranks $PCCs$ in a *statistically optimized order*, and which allows to disclose key of a cryptosystem with less $PCCs$ than using a random order (or the acquisition order). From this idea we have given two methods, one focusing on the maximum of the attacked spike and another focusing on the integral of the parabola drawn by the spike. These techniques are straightforward to implement and need a small computational time. Experiments made on a hardware DES co-processor show that we reduce greatly the number of $PCCs$ necessary to guess the key, and show also that the efficiency of this preprocessing technique increases with the number of $PCCs$ preprocessed. This technique could also enhance attacks on protected implementations trying to balance the leakage, like Dual-Rail Precharge logic styles.

REFERENCES

[1] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *CRYPTO*, 1996, pp. 104–113.
[2] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proc. 19th International Conference on Cryptology (CRYPTO)*, 1999, pp. 388–397.
[3] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Proc. 3rd Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2001, pp. 251–261.
[4] T. Messerges, E. Dabbish, and R. Sloan, "Investigations of power analysis attacks on smartcards," in *Proc. of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology (WOST)*, 1999, pp. 17–17.
[5] R. Bevan and E. Knudsen, "Ways to enhance differential power analysis," in *Proc. 5th International Conference on Information Security and Cryptology (ICISC)*, 2002, pp. 327–342.
[6] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. 7th Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2004, pp. 16–29.
[7] T. Le, C. Canovas, and J. Clédière, "An overview of side channel analysis attacks," in *Proc. of the 2008 ACM symposium on Information, computer and communications security (ASIACCS)*, 2008, pp. 33–43.
[8] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Proc. of the 10th international workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2008, pp. 426–442.
[9] "dpacontest 2008/2009, http://www.dpacontest.org."
[10] C. Clavier, J. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *Proc. of the Second International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2000, pp. 252–263.
[11] F. Standaert, T. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Proc. of the 28th Annual International Conference on Advances in Cryptology (EUROCRYPT)*, 2009, pp. 443–461.