An Efficient Side Channel Attack Technique with Improved Correlation Power Analysis

Ngoc-Tuan Do and Van-Phuc Hoang(✉)

Le Quy Don Technical University, 236 Hoang Quoc Viet, Ha Noi, Viet Nam
phuchv@lqdtu.edu.vn

Abstract. Correlation Power Analysis (CPA) is an efficient way to recover the secret key of the target device. CPA technique exploits the linear relationship between the power model and the real power consumption of an encryption device. In theory, we only need fewer power traces to recover secret key bytes successfully. However, due to the impact of noise, we need a larger number of power traces in order to extract the secret key. Therefore, the computation time becomes a serious problem for performing this attack. This paper introduces a new method to reduce the computation time for CPA method with the technique of finding points of interest which was used for template attack. The experimental results have clarified the efficiency of the proposed method.

Keywords: Correlation power analysis · Side channel attack · AES

1 Introduction

Electronic cryptographic devices are widely used today for securing the secret information. However, there are emerging issues about side channel attack. In the scope of the statistical power analysis attack on cryptographic systems, two efficient techniques were proposed. The first one is well known Difference Power Analysis (DPA) introduced by Paul Kocher [1, 2] and formalized by Thomas Messerges et al. [3]. It uses statistical tools to find out the information correlates to confidential key. The second one is Correlation Power Analysis (CPA) that proposed by Brier et al. [4]. It exploits the correlation between the power model and real power consumption, in order to leak the secret key. These techniques and measurements are carefully taken into account by embedded system designers to know an attacker can measure the power consumption or electromagnetic emanations, which are two of the main physical quantities used for non-invasive attack.

In this paper, we focus on CPA in order to evaluate the security of the AES cryptography because it is an efficient technique to recover the secret key with a simple power consumption model. In fact, we realize that when the cryptographic devices run in low frequency without or with very small noise, CPA only needs a hundred of power traces to successfully extract all secret key. However, for the device running in high frequency and high noise, CPA needs a huge amount of power traces to recover the whole key.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2020
Published by Springer Nature Switzerland AG 2020. All Rights Reserved
N.-S. Vo and V.-P. Hoang (Eds.): INISCOM 2020, LNICST 334, pp. 291–300, 2020.
https://doi.org/10.1007/978-3-030-63083-6_22

Besides, to evaluate the protection method for a secure device, the designers need to consider to the real scenarios. Side channel analysis with real devices usually leads to the high level of noise, then the efficient attacks may require the large numbers of power traces. Therefore, finding an efficient way to analyze those power traces is critical.

The rest of this paper is organized as follows. The original CPA and some related works are described in Sect. 2. In Sect. 3, we will introduce a new CPA, which is an efficient CPA technique that allows to recover secret key byte at least 2× faster than the original CPA. Then, in Sect. 4, we show and compare the analysis results on cryptographic devices, which is an implementation of AES running on XMEGA MCU (on ChipWhisperer board). Finally, we conclude the paper in Sect. 5.

2 Original CPA and Related Works

2.1 Original CPA

CPA was first proposed by Brier et al. in [4]. It exploits the correlation between the real power consumption and the power consumption model. It is based on a power consumption model of the running device at some certain points of time which must depend on the fixed secret key and the plain text changed for each trace. For the Advanced Encryption Standard (AES) attack, attackers mostly used the performing after the first Sbox function to analyze because it has strongly correlation between the real power consumption and power consumption model. The most widely used consumption model is the Hamming distance between two relevant values in the same register [4], or simply the Hamming weight of the particular value [5, 6]. The correlation between the power model and actual power consumption is calculated through Pearson's correlation coefficient by evaluating the linear relationship. The formula of this correlation coefficient between a_i and b_i is given by:

$$\rho = \frac{\text{cov}(A, B)}{\sigma_A \sigma_B} \tag{1}$$

Next, we will discuss the analysis technique for AES algorithm. AES is a cryptographic algorithm executing on one byte separately. Therefore, this paper will focus on analyzing the key bytes independently. Let's denote that p is Pearson's correlation coefficient, N power traces of length L, $t_{n,i}$ is the consumption value of point i in trace n (with $1 \leq i \leq L$, $1 \leq n \leq N$). We have K possible values of a subkey (in this paper $K = 256$), we note $h_{n,k}$ the power consumption model of key k, in trace n and calculated by following formula:

$$h_{n,k} = HW(Sbox(Plaintext_n \oplus k)) \tag{2}$$

where HW denotes the Hamming weight of the Sbox output. With this data, we can see how well the power model and actual power consumption correlation is for each guess

key byte k at each time i, by the following formula:

$$\rho_{k,i} = \frac{\sum_{n=1}^{N}(h_{n,k} - \bar{h}_k)(t_{n,i} - \bar{t}_i)}{\sqrt{\sum_{n=1}^{N}(h_{n,k} - \bar{h}_k)^2 \sum_{n=1}^{N}(t_{n,i} - \bar{t}_i)^2}} \tag{3}$$

where $\bar{h}_k$ and $\bar{t}_i$ are the average of the power consumption model and real power consumption respectively at instant i, respectively.

It can be seen that, by taking the maximum of $\rho_{k,i}$ among all values for i and k, we can decide which power consumption model of the key is most correlated with the actual power consumption. In (3), i is the value of number of samples that we acquired in a trace. Fortunately, we don't need to use all of samples in a power trace, we only focus on a byte that has a leakage position as illustrated in Fig. 1. We use N random plaintexts corresponding to N traces in which each trace has L samples. Note that $t_{i,j}$ is the value of j^{th} sample in the trace number i^{th} $(1 < j < L, 1 < i < N)$, $d_{i,B}$ is the byte value of byte B $(B \in (1; 16))$ in the plaintext number i^{th}. In order to compute the correlation coefficient based on (3), Algorithm 1 as shown below is used. From this algorithm, it is clear that we can easily calculate the mean value of power consumption model and the real power consumption. However, the problem is the iteration of algorithm, it can be computed by $16 \times 256 \times L \times N$. It leads to a huge number of iterations that directly impact on the execution time. Therefore, the values of L and N need to be reduced for improving computing effectiveness. Hereafter, some related works will be discussed.

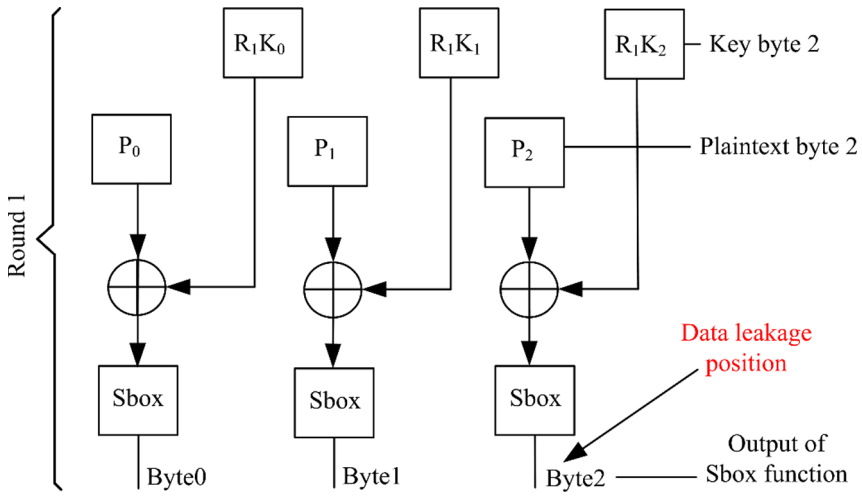

Fig. 1. Data leakage position of AES.

Algorithm 1: Algorithm for computing the correlation coefficient based on the conventional CPA technique.

Input: $d_{i,B}$, $t_{i,j\ (1\le j\le L)}$, L, N
for $B \in (1;16)$ **do**
  for $KEY \in (0;255)$ **do**
    $h_{1<i<N} = HW(SubByte(d_{i,B} \oplus Key))$
    $mean_h = \sum_{i=1}^{N} h_i, mean_{t_j} = \sum_{i=1}^{N} t_{(i,j)}$
    for $j \in (1;N)$ **do**
      $hdiff = h_i - mean_h$
      $tdiff = t_{i,j} - mean_t$
      $sum = sum + hdiff * tdiff$
      $sum1 = sum1 + hdiff * hdiff$
      $sum2 = sum2 + tdiff .* tdiff$
    end for
    $\rho_{KEY,L} = \text{abs}(sum ./ sqrt(sum1 .* sum2));$
  end for
    $\rho_{KEY} = \arg\max(\rho_{KEY,L})$
    $Subbyte_B = KEY$
end for

2.2 Related Works

Recently, CPA technique focuses on the computing effectiveness. In [7], the authors introduced a selection method that can be used to reduce the number of power traces (it means reducing N) to reveal the keys. This is a pre-processing technique which consists selecting a biased subset of the power traces have more information than average traces. Their method improves the performance of the original analysis but the authors does not show the computation analysis in terms of execution time. The work in [8] proposed a technique for CPA enhancement. The authors presented the Partitioning Power Analysis (PPA) technique to combine the techniques of DPA, multi-bit DPA and CPA in a single form. The notion of class was introduced by grouping the power consumption. Each class contains the messages which have the same Hamming distance as the reference state. After that, the correlation is calculated on the classes which use difference coefficients for the difference classes. According to [8], CPA is a particular case of PPA. Despite enhancement of CPA, the computation time is not presented in this method. Another technique is presented in [9], the authors show a new method to recover secret keys with less power consumption traces than the standard CPA. This improvement is done by selecting appropriate plaintexts, namely non-adaptive and adaptive. The authors choose

a set of messages with the most pairwise decorrelated subkeys consumption model. This technique reduces the number of power traces for CPA, but the author does not give any information of computation time.

Most recently, in [10], the authors proposed an algorithm that is similar to the original CPA. This technique is based on the idea of creating, at each point in time, a vector of consumption values indexed by the plaintext byte value. This method also refers to L and N and has two phases. In phase 1, the vector consumption is created in a single pass, requiring $(N \times L)$ iterations. Phase 2 uses these correlated vectors and corresponding power model with Hamming weight. This phase takes $L \times K^2$ $(K = 256)$ iterations. Consequently, this technique uses $(N \times L + L \times K^2)$ iterations for computing one byte key. Despite reducing execution time significantly, this method only works well if the number of power traces is large enough.

This work also employs the idea to reduce the iterations. As mentioned previously, the original CPA requires many power traces and the number of iterations is $16 \times 256 \times L \times N$. Therefore, to enhance the effectiveness of CPA, especially the computation time, in this paper, we aim to propose a technique that reduces the value of L for calculating the final result in CPA technique.

3 Proposed CPA Technique

In this Section, we describe the process of our proposed method and explain how to reduce the number of samples of a power trace in computing. CPA exploits the correlation of power consumption model. In this work, we use the Hamming weight with the data leakage positions as describing in Sect. 2.

In Algorithm 1, we need to calculate the mean of modeled power consumption and the real power consumption. The mean value of power trace needs to use all of the samples of all power traces. Therefore, the number of iterations in computing is large number. If we use a small amount of power traces, it means that we can reduce the number of iterations. Moreover, not every points is important in calculating the correlation, the subkeys only influence the power consumption at a few critical times, and corresponding samples are called Points of interest (POI). If we can pick POI, we can ignore most of the samples. Therefore, using the POI will reduce the number of iterations dramatically. For these reasons, we propose a method that can take the POI base on conventional CPA, and then they are used to calculate Pearson correlation coefficient for all of power traces with whole possible key.

The POI mentioned above is similar to the template attack, a simple technique was used is sum of differences method. In our proposed, the plaintext is random, and the secret key is fixed. We cannot use sum of differences, so the Pearson correlation coefficient is chosen instead. Since a power trace has L samples, we will have L values of correlation, after each iteration, if i^{th} sample has the linear relationship, it will steadily increase or reduce, by contrast it will flexible like noise. Figure 2 illustrates the correlation of 5,000 samples after computing 50 power traces. It is clear that some samples of power trace are much higher than the rest. One of these samples, which is correlated to correct byte key, then constantly goes up. A question is posed, what happen if we only compute a part

of power traces instead of all traces. In this case, we suppose that all correlation values will reduce but these POI still higher than the rest. This is an important point to extract the POI from the large sample. To implement this hypothesis, we propose Algorithm 2.

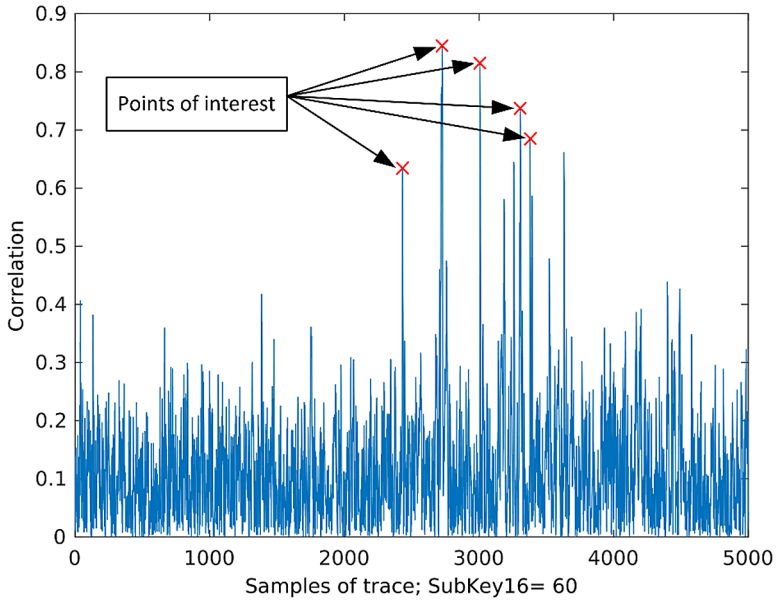

Fig. 2. Points of interest for one byte of secret key.

Inspired by the conventional CPA, our proposed algorithm is also based on the correlation coefficients to find out the highest value. However, the novelty of our proposed approach is the two-phase CPA technique with optimal chosen parameters. In the first phase, a smaller number of power traces is used and denoted as N_1 ($N_1 \leq \frac{1}{2}N$). After that, the Eq. (3) is applied to calculate α highest samples from all guess keys. The value of α is much smaller than L. In the last phase, this equation is used again to compute the correlation for all hypothesis keys again. However, only α samples are calculated instead of using L samples as the first phase. Next, we present the proposed algorithm in detail (Algorithm 2). The notation N, L, $d_{i,B}$ of Algorithm 1 are reused and N_1 denotes for the power traces using in phase 1.

Algorithm 2: Proposed algorithm for computing the correlation coefficient in two phases

Input: $d_{i,B}$, $t_{i,j\ (1\le i\le N,\ 1\le j\le L)}$, L, N_1, N ($N_1 \le \frac{1}{2}N$)

for $B \in (1;16)$ **do**

//Phase 1: taking α sample which have highest correlation value

for $KEY \in (0;255)$ **do**

$h_{1<i<N} = HW(SubByte(d_{i,B} \oplus KEY))$

$mean_h = \sum_{i=1}^{N_1} h_i, mean_{t_j} = \sum_{i=1}^{N_1} t_{i,j}$

for $i \in (1; N_1)$ **do**

$hdif1 = h_i - mean_h$

$tdif1 = t_{i,j} - mean_t$

$sum1\_1 = sum1\_1 + hdif1 * tdif1$

$sum1\_2 = sum1\_2 + hdif1 * hdif1$

$sum1\_3 = sum1\_3 + tdif1.*tdif1$

end for

$\rho_{KEY,\hat{L}} = \text{abs}(sum1\_1 ./ sqrt(sum1\_2.*sum1\_3));$

end for

for $i \in (1;\alpha)$ **do**

$\rho_{\hat{L}'} = \arg\max(\rho_{KEY,\hat{L}} - \rho_{\hat{L}'-1})$

$\bar{N}_{i,\bar{L}'} = N_{i,\bar{L}'}$

end for

//Phase 2: Compute the correlation values with α samples found in phase 1

for $KEY \in (0;255)$ **do**

$mean_h = \sum_{i=1}^{\bar{N}} h_i, mean_{t_{\hat{L}'}} = \sum_{i=1}^{\bar{N}} t_{i,\hat{L}'}$

for $i \in (1:\bar{N})$ **do**

$hdif2 = h_i - mean_h$

$tdif2 = t_{i,\hat{L}'} - mean_{t_{\hat{L}'}}$

$sum2\_1 = sum2\_1 + hdif2 * tdif2$

$sum2\_2 = sumden2\_2 + hdif2 * hdif2$

$sum2\_3 = sumden2\_3 + tdif2.*tdif2$

end for

$\rho_{KEY,\hat{L}'} = \text{abs}(sum2\_1 ./ sqrt(sum2\_2.*sum2\_3));$

end for

$\rho_{KEY} = \arg\max(\rho_{KEY,\hat{L}'})$

$Subyte_B = KEY$

end for

4 Experimental Results

In this section, we evaluate the proposed method on unprotected AES implementation compared to the conventional correlation coefficient used for CPA. We will show the impact of N_1 in computing time, then present two criteria including the accuracy and the execution time for recovering whole key. To prove the practical effectiveness of the proposed technique, we use the synchronized and low noise power traces of low frequency AES implementation acquired on ChipWhisperer board.

To illustrate the accuracy of algorithm, an AES implementation on XMEGA MCU is used. The number of correct key bytes will be used as the criteria to assess. In this work, 10,000 power traces are acquired from ChipWhisperer board and then added Gaussian noise ranging from 0.1 to 0.2 with steps of 0.02. As mentioned above, we will show the impact of N_1 in the proposed algorithm. In this case, the fixed value of $\alpha = 50$ is chosen. All experiments are implemented in MATLAB software.

In Fig. 3, it is clear that the more noise is added, the higher of N_1 is needed for extracting secret key successfully. However, the value of N_1 directly affects the computation time, therefore we cannot choose the too large value of N_1. In the experiments, with the value of $N_1 = [1/3\ N,\ 1/4\ N,\ 1/5\ N]$, we find that the value $N_1 = 1/3\ N$ will leads to the better results than other ones. Hence, this value is used for evaluating the execution time with the detail results in the next section.

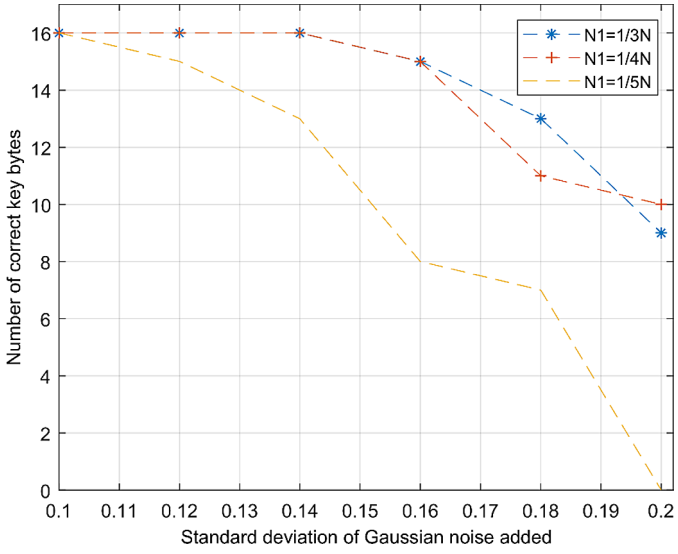

Fig. 3. The average of the correct number of key bytes corresponds to $N1$ and the standard deviation increases.

To demonstrate the effectiveness of timing execution, the proposed algorithm is implemented with the same configurations of the conventional CPA algorithm. In this experiment, we used a number of power traces in the range from 1,000 to 10,000 corresponding to the standard deviation of noise ranging from 0.06 to 0.15 and with $N_1 = 1/3\ N$. These parameters are listed in Table 1.

Table 1. Parameters of CPA attack on the whole AES key for computation time measurements.

Experiment no.	1	2	3	4	5	6	7	8	9	10
N	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
N_1	330	667	1000	1333	1667	2000	2333	2667	3000	3300
Noise deviation	0.06	0.07	0.08	0.09	0.10	0.11	0.12	0.13	0.14	0.15

Figure 4 shows the results of experiment. It can be seen that the proposed technique is much faster than the conventional CPA. As expected, the value of N_1 directly affects the calculation time and it is effective with the broad range of number of power of trace, not only for the large number as in [10]. When the number of power traces is larger, with the appropriately chosen value of N_1, the higher efficiency of the execution time can be achieved. The smaller N_1 is chosen, the lower computation time is achieved. In this case, we only use the value of $N_1 = 1/3\ N$, however, depending on the standard deviation value of noise, N_1 could be chosen smaller. According to Fig. 3, if the standard deviation is 0.14, the value of $N_1 = 1/4\ N$ can be applied. Hence, the efficiency of execution time will be higher.

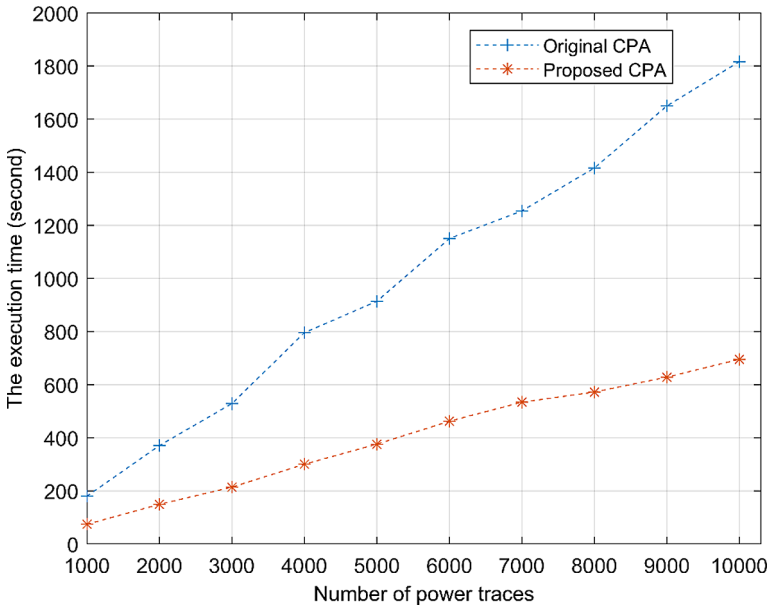

Fig. 4. Results of computation time for conventional and proposed CPA techniques, for power traces containing 5000 samples.

5 Conclusion and Future Work

In this paper, we have proposed a new CPA method in order to recover the secret key. This technique contains two phases, in which the first phase calculate the number of points of interest of all possible key bytes (very small compare to number of power trace samples). Then, the points of interest are used to compute the highest correlation value of all guess key bytes. The effectiveness of execution time is demonstrated in experiments with various number of power traces which have clarified the efficiency of the proposed method. In the future work, a mathematical representation and analysis will be investigated in detail.

Acknowledgment. This work is funded by Ministry of Science and Technology (MOST), Vietnam, under the grant number HNQT/TKCG/04.20.

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
2. Kocher, P., Jaffe, J., Jun, B.: Introduction to differential power analysis and related attacks (1998). http://www.cryptography.com
3. Messerges, T., Dabbish, E., Sloan, R.: Investigation of power analysis attacks on smartcards. In: Usenix Workshop on Smartcard Technology (1999). http://www.usenix.org
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
5. Coron, J.-S., Kocher, P., Naccache, D.: Statistics and secret leakage. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 157–173. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45472-1_12
6. Mayer-Sommer, R.: Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In: Koç, Çetin K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 78–92. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44499-8_6
7. Kim, Y., Sugawara, T., Homma, N., Aoki, T.: Akashi Satoh: Biasing power traces to improve correlation in power analysis attacks. In: First International Workshop on Constructive Side-Channel Analysis and Secure Design, Citeseer, pp. 77–80 (2010)
8. Le, T.-H., Clédière, J., Canovas, C., Robisson, B., Servière, C., Lacoume, J.-L.: A proposition for correlation power analysis enhancement. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 174–186. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_14
9. Ouladj, M., Guillot, P., Mokrane, F.: Chosen message strategy to improve the correlation power analysis. IET Inf. Secur. **13**(4), 304–310 (2019)
10. Quentin L. Meunier. FastCPA: Efficient Correlation Power Analysis Computation with a Large Number of Traces. In: 6th Cryptography and Security in Computing Systems (CS2 2019), https://doi.org/10.1145/3304080.3304082. hal-02172200 (2019)