SCIENCES SORBONNE UNIVERSITÉ

Side-Channel Attacks

DPA, CPA : Application on AES-128

January 19, 2020

Students :
Abdel Rahman TALEB
Ibrahim BAMBA
François GAGNARD

Supervisors :
Karine HEYDEMANN
Quentin MEUNIER

Contents

Introduction

Nowadays, cryptography is omnipresent on various embedded systems such as smart cards. The encryption algorithms on these components have to provide high security against malicious actions. Meanwhile, the proved "theoretical" security of such algorithms does not necessarily guarantee safety when implemented "in practice". Indeed, these algorithms are often vulnerable to attacks that intercept and analyze specific measures, such as electricity consumption, or execution time. These attacks are called Side-Channel Attacks (or SCA).
SCA are attacks of type "divide-and-conquer" which aim to reconstruct the full secret parameter of the algorithm (or the key) from smaller pieces called subkeys. The execution of such a procedure results in a set of possible values for each subkey. Finding the secret parameter thus amounts to recombining the sequence of subkeys that are most "probable" among the others, and which corresponds to that of the secret parameter for a normal execution of the algorithm.
A lot of encryption algorithms have been proven vulnerable to SCA attacks when implemented without countermeasures. During this project, we mainly focus on two widely-known side-channel attacks: Differential Power Analysis (DPA), and Correlation Power Analysis (CPA). In the rest of this report, we introduce the two attacks, and study their impact and performance on the AES-128 symmetric encryption algorithm.

I) AES-128 Encryption Algorithm

1) Description

The Advanced Encryption Standard (or AES) is a block cipher encryption algorithm based on the design of a substitution–permutation network. It processes the input in blocks of fixed size of 128 bits (16 bytes), and uses a key of size 128, 192, or 256 bits. For this project, we focus on keys of 16 bytes (AES-128). It consists of 4 transformations applied to the bytes of the input block, that is seen as a 4×4 array of bytes, and repeated for 10 rounds for the AES-128. Algorithm 1 offers a high-level description of the procedure.

Algorithm 1 AES-128 Encryption Algorithm

Data: Input Block $B = b_0 \dots b_{15}$, Key $K = k_0 \dots k_{15}$
Result: Cipher $C = c_0 \dots c_{15}$
$C \leftarrow$ **AddRoundKey**(B, K)
for $i = 1 \dots 9$ **do**
    $C \leftarrow$ **SubBytes**(C)
    $C \leftarrow$ **ShiftRows**(C)
    $C \leftarrow$ **MixColumns**(C)
    $C \leftarrow$ **AddRoundKey**(C, K)
end
$C \leftarrow$ **SubBytes**(C)
$C \leftarrow$ **ShiftRows**(C)
$C \leftarrow$ **AddRoundKey**(C, K)
return C

All of these operations provide **Confusion** and **Diffusion** properties in order for each input bit to be diffused across each output bit after enough rounds of the algorithm. The confusion is done through the substitution steps, while the diffusion is done through the permutation steps of the network. In fact :

- The confusion property is insured by **AddRoundKey**(C, K) which combines each byte of C with each byte of the key K, followed by **SubBytes**(C) which performs a non-linear substitution on each byte of C using a look-up table.
- The diffusion property is insured by **ShiftRows**(C) that performs cyclic shifts on the last three rows of the 4×4 array C, as well as **MixColumns**(C)

that also does linear mixing on each column of the 4×4 array C.

After enough rounds, the algorithm would have sufficiently mixed the message block with the secret key, and diffused it over all the bytes of the output cipher block. AES algorithm has been designed to be resistant against known attacks, and to be a simple and fast procedure to be designed on many CPUs. However, many works have shown that this "textbook" version of AES-128 is in fact vulnerable to side-channel attacks. In the rest of this report, we will discuss two well-known attacks (DPA & CPA) that manage to retrieve the secret key from a number of executions of the algorithm, and conclude with a comparison between both attacks regarding efficiency and complexity.

2) Attack Target

Side-channel attacks that target the AES-128 algorithm are able to retrieve the secret key by independently attacking each of the 16 key bytes. With that said, there is clearly a step in the algorithm which output bytes depend each on the corresponding input and key bytes independently. Indeed, the procedures **ShiftRows**(C) and **MixColumns**(C) are constructed to insure the diffusion of the input bytes on each of the output bytes, allowing a mix among all of the input bytes to all of the output bytes. So when looking for an attack target during the execution, it is clear that it should be before any **ShiftRows** or **MixColumns** is done. Looking more carefully at the very first **AddRoundKey**(B,K) and **SubBytes**(C) performed, we can rewrite these two operations as :

$$SBox[b_i \oplus k_i] = c_i \text{ , for } i = 0 \dots 15 \tag{1}$$

where $SBox[]$ refers to the aforementioned non-linear lookup table. With enough information on b_i and c_i, a reasonable dependency between the output and the key byte k_i can be extracted. Notice that each of the 16 substitutions is performed regardless of the others, using one key byte at a time. This property is the target that allows for SCA to work on the AES algorithm. From a total execution, one hopes that the power consumption during the operations from equation (1) will reveal some information on the corresponding key bytes.
In general, SCA attacks on the AES algorithm target either the first **AddRoundKey**(B,K) or **SubBytes**(C) of the first round. In this work, we focus on the SBox operation.

II) Differential Power Analysis (DPA) [2]

1) Description

In cryptanalysis of cryptographic hardware device, the Differential Power Analysis (or DPA) [2] is the study of the electric power and voltages entering and leaving a circuit in order to discover secret information such as the encryption key. Some operations, which are more costly, increase the electrical consumption of the circuit, in particular by the use of more components (analog or logic). When handling the secret, an analysis of these variations and peaks allows to extract valuable informations about the secret. Precisely, DPA attack relies on two main assumptions: power consumption at a given moment depends on the result of the operation executed at that moment, and that the consumption of a fixed computation follows a normal distribution (which is mostly the case in practice). Thus, the attack considers that the value of one bit (or more generally the hamming weight) of a given computation result affects the power consumption at that instant.
Like most side-channel attacks, DPA targets each byte of the secret key independently. It focuses on a step in the algorithm where the output depends on the subkey value and the input text, for the power consumption traces to reveal a certain amount of information on the correct key. The procedure consists mainly of four steps repeated for each subkey:

1. Locate a critical computation in the algorithm which output depends on the input text and the key byte used. At the moment of this computation, the power consumption should be biased according to the key value.

2. Acquire a certain number of power consumption traces, for different executions of the algorithm, using the same key but with different input texts.

3. For each possible value of the subkey, compute the output of the critical operation with respect to the acquired input text bytes, and partition the corresponding traces into two sets according to one chosen bit value (1 or 0) of the output. (One hopes that this partitions the traces into "low hamming weight" and "high hamming weight" output). Then, compute the difference between the two traces obtained from the averaged sets of

traces, and extract the maximum of these differences.

4. After obtaining a maximum difference for each possible value of the subkey, the one with the highest difference is the best guess for the considered byte. Repeating this process for all bytes reveals the full secret key.

Under the right hypothesis for the key byte, the traces would be partitioned according to a criterion that influences the power consumption, at the critical computation moment. Calculating the difference of the averaged sets reveals such an influence on the result, which is why the maximum difference is considered in the described procedure. Meanwhile, for other hypothesis of the key byte, the attack will create an arbitrary partition of the traces and both averages of the two sets will be almost identical, since they both correspond to an arbitrary partitioning of the traces. This modeling of the power consumption allows the attacker to infer the secret key by observing the difference between the two averaged traces.

2) Application to AES-128

To apply DPA on the AES-128 algorithm, one must find the critical moments of the execution that depend on the output and the secret key value. As explained in Section I) (Attack Target), the considered operation is the first round of **SubBytes**(C), specifically the output of the SBox given in equation (1), which input depends on the plain text and the key value, since the different bytes are not mixed yet and less hypotheses need to be tested. The attacker chooses one of the 8 bits of the SBox's output to partition the traces according to it. Under the right hypothesis of the considered key byte, the acquired consumption traces will be influenced by the hamming weight of the SBox's output at that moment, and more precisely by the value of the bit we use to partition. The key is composed of 16 bytes, and every subkey will have 256 possible values to test. The complete algorithm is described in Algorithm 2. An implementation in C of this algorithm is submitted along with this report.

3) Experimental Results

To perform the DPA attack, the AES-128 algorithm has been executed a number of times, and power consumption traces of the executions have been acquired

using a ChipWhisperer board. The traces were of 6000 sample points each, in order to be sure to completely cover the first round of SBox performed, where the attack should take place. As described earlier, at the right moment during the computation, and under the right key byte hypothesis, the power consumption should be influenced by the hamming weight (or the number of 1 bits) in the Sbox's output.

Algorithm 2 DPA Attack [2] on AES-128 Encryption Algorithm

Data: A set of consumption traces $T = \{t_i\}_{i=1..n}$ composed of m sample points and associated each to a different known 16-bytes plaintext $P = \{p_i\}_{i=1..n}$. A bit index ind_bit to partition the traces.

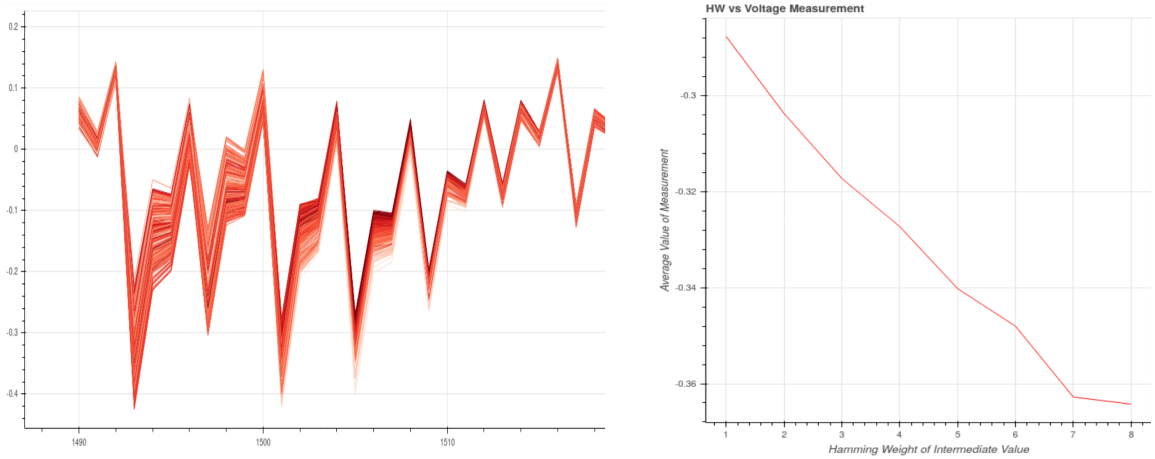
Result: A guess for the secret key K

$K \leftarrow []$
for $j = 0 \dots 15$ **do**
  $Diffs \leftarrow []$
  for $k = 0 \dots 255$ **do**
    $T_0 \leftarrow [],\ T_1 \leftarrow []$
    for $i = 0 \dots n-1$ **do**
      if $(SBox[p_{ij} \oplus k][ind\_bit] = 1)$ **then**
        $T_1.append(t_i)$
      else
        $T_0.append(t_i)$
      end
    end
    $T'_0 \leftarrow$ Average of all traces in T_0
    $T'_1 \leftarrow$ Average of all traces in T_1
    $Diffs.append(Max(|T'_0 - T'_1|))$
  end
  $K.append(ArgMax(Diffs))$
end
return K

An observation can be made to validate this theory. Out of the 6000 samples captured, we determined that the first round of Sbox corresponds to the sample points between 1300 and 2800 approximately. To test the relation between the power consumption and hamming weight of the Sbox's output for the first byte for example, we retrieved the value of the first key byte k_0 and computed

the hamming weight of $SBox[p_0 \oplus k_0]$, where p_0 is the first byte of the plaintext associated to each of the traces. Figure (1a) represents the power consumption, where each of the traces' curve have been colored according to the hamming weight of the Sbox's output (light red for lowest weight, dark red for highest weight). The sample points in these curves are sub-intervals that cover the first iteration of the $SubBytes$ procedure since we are testing the theory for the first byte. Since the key byte used is the right secret value, the correlation between power consumption and hamming weight can be observed specifically at the sample point 1505, where the power consumption is increasing with the hamming weight (color of the curve darkens in a degraded style). This is because at that point in the execution, we are under the right key hypothesis, and both values should correlate as predicted, while at other sample points, the coloring order corresponds to an arbitrary order. Precisely, the curve in Figure (1b) represents the variation of the power consumption for the traces, with respect to the hamming weight at the sample point 1505, and we can clearly see that it is a quasi-linear function where the voltage decreases (so consumption increases) with the hamming weight increase. This experiment also works for the other 15 bytes of the key, but we spare the experimental observations that are identical to the one studied.

After validating the DPA attack theory, the attack has been tested on the acquired traces. Figure 2 shows the number of correct key bytes retrieved by the attack, with respect to the number of traces used (the variable ind_bit to partition the traces have been set to maximize the number of correct bytes). It is clear that increasing the number of used traces allows the attack to have a more precise vision of the power consumption's evolution, and so to be able to break larger parts of the secret. Using 5000 traces, the attack finds 14 out of 16 of the secret key bytes, which is the maximum it was able to find. Using more traces (10000 for example) does not allow a complete success in attacking all of the 16 bytes. This is most probably due to the fact that DPA attack is very sensitive to the exact values of power consumption, and in our case, it is possible that the ChipWhisperer wasn't able to acquire error-free traces. This is also due to the fact that the traces' partitioning is done using only 1 bit of the SBox's output, which does not necessarely represent the hamming weight of the considered value. One may hope that a bit value '1' means higher hamming weight, but this does not hold in 100% of the cases.



(a) Power consumption traces colored according to the hamming weight of the Sbox's output at the first round using key byte k_0 (light red for lowest weight, dark red for highest weight).

(b) Evolution of the power consumption, with respect to the hamming weight of the SBox's output for each of the plaintexts' first byte p_0 xored with the key byte k_0.

Figure 1: Graphical image of the hamming weight effect's theory on the power consumption traces acquired for 1000 executions of AES-128 using the same secret key (illustrates the case of the first key byte k_0).

Figure 3 illustrates the success of the attack according to the choice of ind_bit variable to partition the traces, for three different configurations (1000, 3000 and 5000 traces). It is clear that as in Figure 2, increasing the number of traces increases the success rate of the attack. On another hand, and while this observation cannot be pointed out theoretically, we can see through the experiments that $ind\_bit = 3$ (4th bit of the SBox's output value) is the best choice that makes the attack break the most secret bytes. While choosing the 6th bit ($ind\_bit = 5$) results in a complete failure of the procedure, regardless of the number of traces used. This shows that one should test different configurations when executing the attack to be able to find the best case scenario.

Regarding the sample points interval that corresponds to the first round of the SBox and that made the attack work, an analytic experiment is done in the next section using the CPA attack (cf. Figure 6) to locate the sub-interval that measured the consumption exactly at the 16 iterations of the SBox's computation.

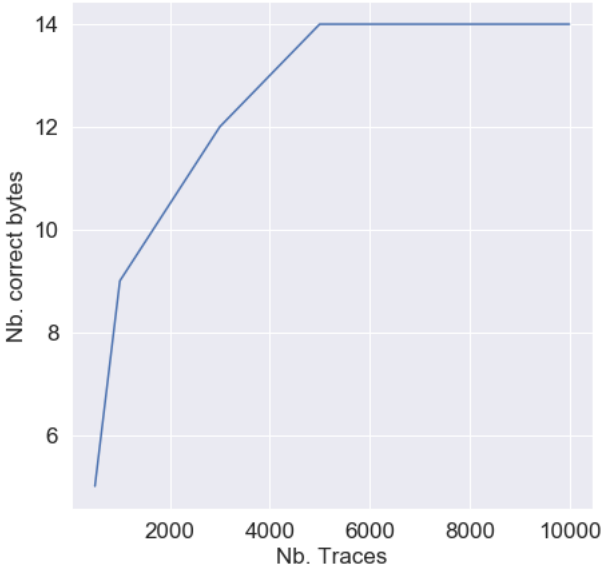


Figure 2: Number of correct key bytes guessed by the DPA attack, with respect to the number of traces used (of 6000 samples each).

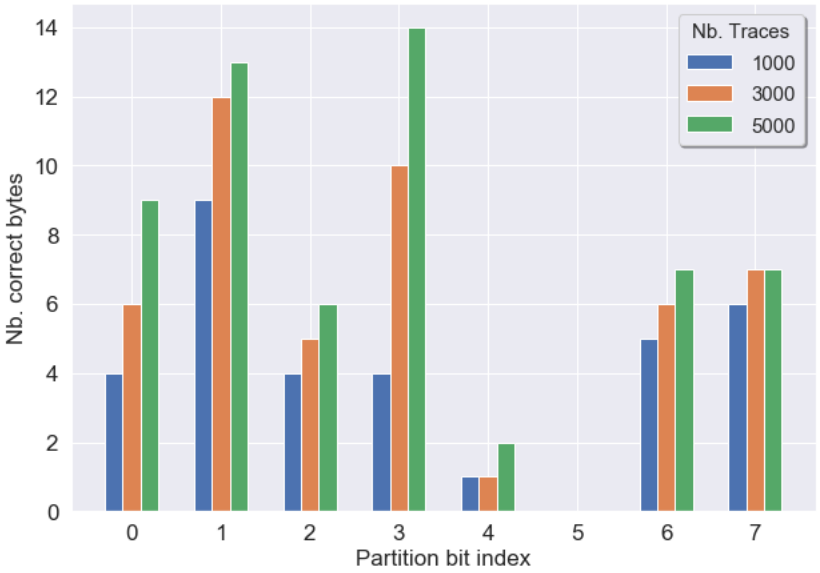


Figure 3: Number of correct key bytes guessed by the DPA attack, with respect to the index of the bit ind_bit used to partition the traces into two sets (from 0 to 7), using different number of traces of 6000 samples each.

III) Correlation Power Analysis (CPA) [1]

1) Description

Now we study another well-known SCA attack. Correlation Power Analysis (CPA) [1] is a side-channel attack that aims to retrieve the secret key based on the power or electrical consumption of an encryption computation using the key. It relies on the fact that a certain consumption at a given moment highly depends on the computed result at that moment and the value of the secret key. For these critical moments in the execution, CPA tries to find a correlation using the Pearson Correlation Coefficient, between the "real model" of power consumption measurements, and a "virtual model" constructed by the attacker, hoping to simulate a consumption that corresponds to the real model, for a correct guess of the key. Like most other SCA, CPA works on each subkey at a time, making the number of possibilities to test more reasonable. In general, the attack consists of four main steps repeated for each subkey:

1. Construct a virtual consumption model for the encryption algorithm, which at critical moments of the execution, will correlate with the real power consumption of the executed operation depending on the result and the secret key value.

2. Acquire a certain number of power consumption traces, for different executions of the algorithm, using the same key but with different input texts.

3. Consider every possible value of the key byte to compute the virtual consumption model values using the acquired traces and the known plain texts.

4. For every point in the traces, calculate the Pearson Correlation Coefficient between the modeled consumption and the actual value in the acquired traces. The correlation coefficient will have a maximum value when the guessed subkey correponds to the actual one in the secret key. This value is then most probably a correct guess for the right key.

2) Application to AES-128

As for the DPA attack, to apply CPA on the AES-128 algorithm, the considered operation is the first round of **SubBytes**(C), since the different bytes are not mixed yet and less hypotheses need to be tested. The output of the SBox depends only on the input text and the used key byte. The Hamming Weight of the Sbox's output value, or the Hamming Distance between the input and the output of the Sbox, are the most commonly used to construct the virtual consumption model. The key is composed of 16 bytes and each one is attacked independently, leaving 256 possible values to test for each subkey. The complete algorithm is described in Algorithm 3. An implementation in C of this algorithm is submitted along with this report.

Algorithm 3 CPA Attack [2] on AES-128 Encryption Algorithm

Data: A set of consumption traces $T = \{t_i\}_{i=1..n}$ composed of m sample points and associated each to a different known 16-bytes plaintext $P = \{p_i\}_{i=1..n}$

Result: A guess for the secret key K

$K \leftarrow []$
for $j = 0 \dots 15$ **do**
  $Coeffs \leftarrow []$
  for $k = 0 \dots 255$ **do**
    $HW \leftarrow \left[\text{ Hamming\_Weight}(\ SBox[p_{ij} \oplus k]) \ \right]_{i=1..n}$
    $L \leftarrow []$
    for $l = 0 \dots m-1$ **do**
      $L.append($ Pearson Correlation Coefficient between HW and the list of of sample points of index l from the n traces of the set T $)$
    end
    $Coeffs.append(\ Max(L) \ )$
  end
  $K.append(\ ArgMax(Coeffs) \ )$
end
return K

3) Experimental Results

As for the DPA attack experiments, the same power consumption traces of several executions of the AES-128 that have been captured using a ChipWhisperer

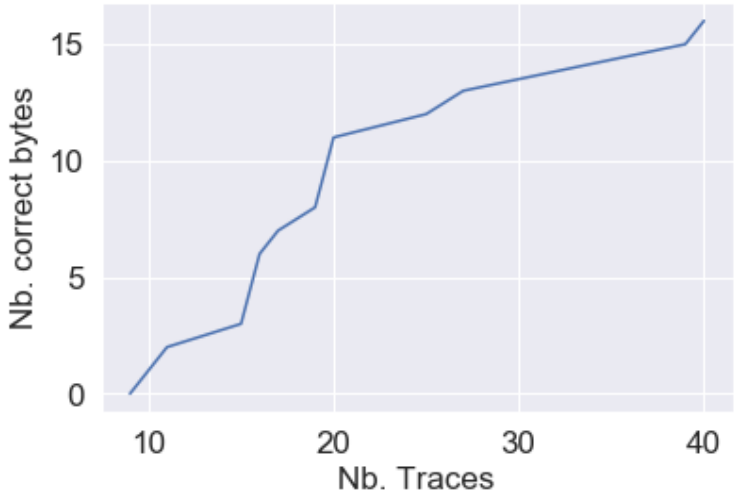


Figure 4: Number of correct key bytes guessed by the CPA attack, with respect to the number of traces used (of 6000 samples each).

board are used to test CPA. The traces are of 6000 sample points, to be sure to cover the complete first round of the SBox. For each of the key bytes, under the right value hypothesis, one would expect for the array of sample points corresponding to $SBox[b_i \oplus k_i]$ in each of the traces, to correlate with the virtual consumption model constructed using the considered hypothesis.

Experimental results of this attack are shown in Figure 4 for different number of traces. We can clearly see that 40 traces were enough in order to break the entire secret key. As expected, decreasing the number of captured traces degrades the attack's performance, where having only 9 traces is the worst case and none of the key bytes are retrieved. However, the fact that 40 traces are enough for the attack to work perfectly is a very strong result. In fact, this shows that CPA is not as sensitive as DPA to the values of the power consumption of the traces. It just needs a certain "pattern" or "flow" to be respected by the measured values, in order for them to have a relation with the hamming weight consumption model constructed with the right key hypothesis. The attack also uses all of the information contained in the hamming weight of the SBox's output, unlike DPA that partitions using 1 bit of the output. Statistically, Figure 5 shows the number of times each of the key bytes $(k_0, \ldots, k_{15})$ is guessed correctly, through all of the experiments with different number of traces. We can see that the power consumption at the moment of computation using the 12th key byte k_{11} is the best to correlate with the virtual consumption model.

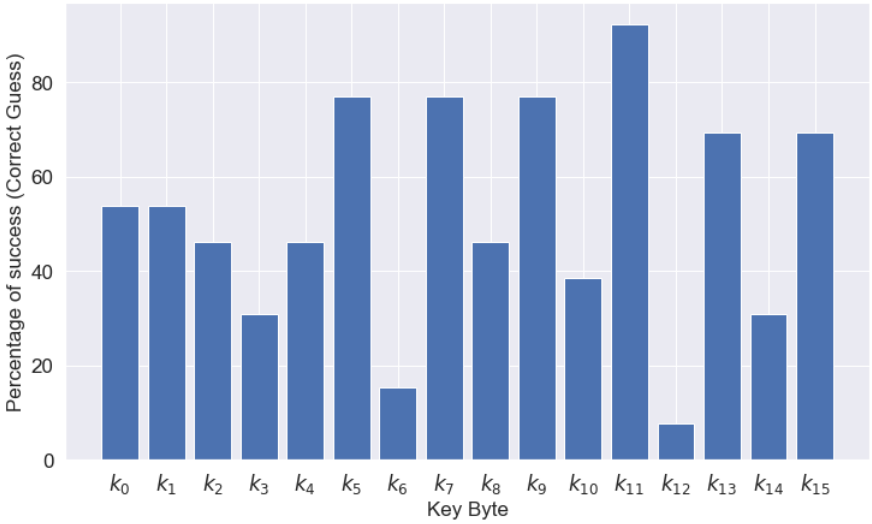


Figure 5: Percentage of correct guesses of each of the key bytes $(k_0, \ldots, k_{15})$ by the CPA attack, over 13 experiments in total done with an increasing number of traces (from 12 to 40).

It has been retrieved in 92% of the experiments, which means that even with a very small number of traces, the power consumption values are representative enough for the correlation to work. However, the hardest key byte to retrieve is the 13th k_{12}, which only worked in 7% of the experiments, meaning that it needed more traces for the correlation to work. It is the hardest byte to be guessed during the attack if the number of traces is not enough. We focus now on analyzing the interval of sample points among the 6000 that corresponds to the first round of SBox and that made the attack work. Figure 6 shows the number of correct guesses of key bytes using 40 traces as before, but modifying

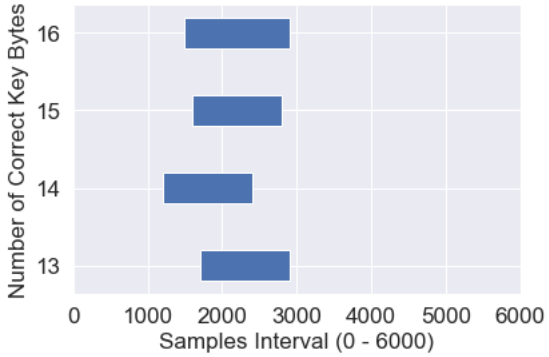


Figure 6: Number of correct key bytes guessed by the CPA attack, using 40 traces, with respect to the samples sub-interval used in the interval [0;6000[.

the number of samples and the interval used among the 6000 measured. We can see that using samples in the interval [1500;2900] results in a complete success of the attack with 16 bytes retrieved. While moving this interval to the left or to the right, or decreasing its size reduces the number of correct guesses. For example, only 13 bytes are guessed when considering samples in [1700;2900], and other experiments not plotted on this figure shows that further intervals that don't include sample points at least in [1200;3000] results in a complete failure of the attack. This shows that the first round of the SBox is captured in this interval by the ChipWhisperer. In general, when not sure about these exact values, the attack will be executed on all of the sample points and the algorithm will be able to correlate the consumption values at the correct sample points.

IV) DPA vs. CPA : Comparison & Results

After presenting both DPA and CPA attacks, and testing them on the AES-128 encryption algorithm, a comparison can be made on the efficiency and the hypotheses of these attacks.

First, the DPA attack as seen in Section II), proved to be very sensitive to the power values measured in the acquired traces. While the considered criterion (the hamming weight of the SBox's output) to partition the traces into two sets seems to affect the power consumption, this criterion is reduced to partitioning the traces according to the value of one bit of the output, which is not representative of its hamming weight in 100% of the cases. Plus, the computation of a "difference" between the "averaged" traces of these two sets, results in an important sensitivity to the values captured during the acquisition. Since the maximum difference value depends on each of the sample points, the slightest error in the traces measurement can very much affect the computed values of DPA. Conversely, the CPA attack is less sensitive to the exact power consumption values. In fact, it focuses on the evolution of the power consumption at a certain sample point, and its correlation with a virtual consumption model (Hamming weight model) under different key hypotheses. This correlation does not depend on the exact values of the traces, but on a larger view of the evolution of these values with respect to the hamming weight of the manipulated value at that point. The correlation coefficient is more tolerant to small error values in the measurement as long as the two consumption models correlate in the end, unlike the DPA attack, as seen in the experiments.

Second, the sensitivity level of both of the algorithms also affects the number of traces needed for the considered attack to work properly. As seen in the experimental results of Sections II) and III), Table 1 sums up the best configuration for each of the attacks and their success rate (See rows 1, 2, 3 and 4 of the table). The DPA attack needed 5000 traces to get its best accuracy, and even with that, it was only able to recover 14 bytes of the secret key. Using the same acquired traces, the CPA only needed 40 of these acquisitions to completely break the secret. The correlation under the right key hypothesis was clearly

observable using a small number of traces. Meanwhile, the measured values were not enough to allow a complete success of the DPA attack. This proves that the CPA attack, being simpler to analyze, is also more efficient on such encryption algorithms. In terms of speed, the CPA attack only took 1.7 seconds for 40 traces, while DPA took 99 seconds to recover only 14 key bytes.

This brings us to the last comparison of the two attacks regarding complexity and execution time. In fact, as shown in Table 1, the best configuration for CPA is with 40 traces, while with DPA is 5000 traces, so the execution time is measured for different configurations for each attack. We would like to test both algorithms with the same number of traces and samples. Row 5 of Table 1 shows execution times using 5000 traces for both CPA and DPA. We can see that the CPA algorithm is almost 12x slower than the DPA procedure. This is justified by the operation count of both algorithms. Actually, it is true that both CPA and DPA attacks have a total number of iterations in $\mathbf{O}(\#\mathbf{traces} \times \#\mathbf{samples} \times \#\mathbf{key\_hypotheses} \times \#\mathbf{key\_bytes})$. Meanwhile, they differ in the computation done to manipulate the traces. DPA attack does a simple addition of sample points along the columns of the traces, then executes a subtraction between two averaged traces and finds maximum values. However, the CPA attack, while also considering each column of the traces (each sample point), it computes a correlation coefficient between each of the columns with the hamming weight of the computed SBox output for each key hypothesis. This correlation coefficient adds multiplication and division operations after computing means for each of the arrays. Without including all the details, the cost of computing this coefficient is clearly higher than performing additions, subtractions and two averages for each column as for the DPA attack, which results in a slower execution time for CPA. Nevertheless, this experiment uses the same number of traces to evaluate both attacks, while we saw that CPA needed much fewer traces than DPA to obtain good results. So as far as the execution time is concerned, CPA is still in practice more efficient than DPA since it does need as much consumption acquisitions as the latter.

	DPA	CPA
Best configuration	5000 traces	40 traces
Number of correct bytes guesses for best config.	14	16
Execution time for best config.	99.5 sec	1.7 sec
Execution time with 5000 traces		1254.3 sec

Table 1: Execution times and best configurations for each of the CPA and DPA attacks.

Conclusion

In this report, we have studied Side-Channel attacks, specifically applied to AES-128 algorithm, that showed us the vulnerability of the latter against such attacks. Several countermeasures have been implemented in the industrial field to prevent such vulnerabilities (masking, ...). We closely studied two well known attacks DPA and CPA, applied them to AES-128, and compared their performances. Correlation Power Analysis attack proved to be more efficient to break the secret and simpler to analyse. While the CPA execution time is much higher than that of the DPA for the same number of traces used, CPA proved to need less traces for the secret information to be revealed through a correlation between the real power consumption model and a virtual consumption model, while DPA is very sensitive to the values measured during the acquisition of the traces. Simple errors resulted in a partial failure of the latter, even with a large number of provided traces. This sensitivity is avoided in the CPA attack since the algorithm looks for a larger view of the correlation between two particular consumption models.

We would like to thank Professor HEYDEMANN Karine, and Professor MEUNIEUR Quentin, for their supervision of this work and their constant help and support.

References

[1] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation power analysis with a leakage model”. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2004, pp. 16–29.

[2] Paul Kocher, Joshua Jaffe, and Benjamin Jun. “Differential power analysis”. In: *Annual International Cryptology Conference*. Springer. 1999, pp. 388–397.