Journal of ELECTRICAL ENGINEERING, VOL. 63, NO. 2, 2012, 125–129

COMMUNICATIONS

ON PRACTICAL RESULTS OF THE DIFFERENTIAL POWER ANALYSIS

Jakub Breier [*] — **Marcel Kleja** [**]

This paper describes practical differential power analysis attacks. There are presented successful and unsuccessful attack attempts with the description of the attack methodology. It provides relevant information about oscilloscope settings, optimization possibilities and fundamental attack principles, which are important when realizing this type of attack.

The attack was conducted on the PIC18F2420 microcontroller, using the AES cryptographic algorithm in the ECB mode with the 128-bit key length. We used two implementations of this algorithm - in the C programming language and in the assembler.

K e y w o r d s: differential power analysis, side channel attacks, cryptanalysis

1 INTRODUCTION

There are many devices, which can be used for cryptographic computations. Special purpose microprocessors or smartcards are widely used because of the size and sufficient computing power. For example, smartcards can store the cryptographic keys or they can even do the encryption or the digital signature, so the key never leaves the device. Besides the intended input and output these devices have additional outputs and often additional inputs, which can be used by the attacker to reveal the secret key used by the device. The information leaked unintentionally by the device is called the side-channel information and the attacks which exploit this information are called the side channel attacks. They are categorized as passive, non-invasive attacks, because the device works in a standard way and only interfaces are used to obtain the necessary information.

Cryptographic devices, like any other computing devices, consume the electric power. The consumed power depends on the operations performed by the device. It was shown that the information about the consumed power can be misused by an attacker. The power analysis attack, if correctly performed, can be very strong in comparison to the classical cryptanalysis attack.

Differential power analysis [3] (DPA) attack is the most popular type of the side channel attacks. It is because it does not need the detailed knowledge of an attacked device and it can reveal the secret key even if the captured samples contain a greater amount of noise. These attacks require a bigger amount of samples in comparison to the simple power analysis, so it is usually necessary to have a physical control over the device for a certain period of time. When performing a DPA attack, only the knowledge of the used encryption algorithm is usually needed. Statistical analysis of the samples enables to reveal the secret key.

This paper presents practical aspects of the DPA attack performed on a universal microcontroller, with the software implementation of the AES algorithm. There are described both successful and unsuccessful attack attempts along with the detailed description of the performed attacks and techniques to improve the effectiveness of the DPA [1].

The rest of the paper is organized as follows. Section 2 presents the attack methodology and used devices and Section 3 provides an overview of the used attack strategy. Next three sections summarize attack test, in Section 4 are plaintext dependency tests, Section 5 presents the attack on the implementation in the C language and Section 6 presents the attack on the assembler implementation of the AES algorithm. The last section provides conclusions of our work.

2 USED DEVICES AND ATTACK METHODOLOGY

As a platform for the cryptographic algorithm implementation we choosed the microcontroller PIC18F2420 by Microchip Technology Inc. The parameters of the microcontroller are listed Tab. 1.

The communication between the computer and the microcontroller is realized by the RS-232 interface. The algorithm AES [2] in ECB mode with the key size of 128 bits is software-implemented in the microcontroller. The encryption is executed in a following manner: a plaintext data block is sent to the microcontroller, there it is encrypted and sent back. For sending challenges and receiving replies we used a personal computer. The mi-

[*] Institute of Applied Informatics, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, breier@fiit.stuba.sk; [**] Slovak National Security Authority, Bratislava, marcel.kleja@nbusr.sk

DOI: 10.2478/v10187-012-0019-5, ISSN 1335-3632 © 2012 FEI STU

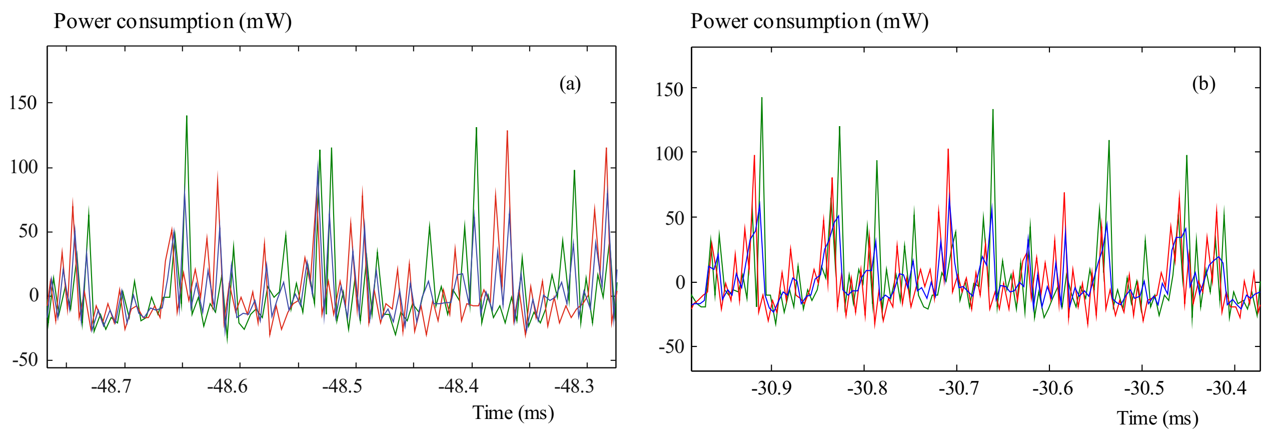

Fig. 1. Sample alignment

Table 1. Microcontroller parameters

PIC18F2420	
Architecture	8-bit
Program Memory Type	Flash
Program Memory Size	16 kB
EEPROM	256 B
RAM	768 B
CPU Frequency	40 MHz

Table 2. Chosen oscilloscope parameters

LeCroy WavePro 7200A	
Analog Bandwidth	2 GHz
Rise Time	225 ps
Input Channels	4
Vertical Resolution	8 b
Sensitivity	50 Ω
Single-shot Sampling Rate / Channel	10 GS/s
Maximum Sampling Rate	20 GS/s
Trigger Sensitivity	2 div < 2 GHz

crocontroller communicates with the computer through the RS-232 interface.

In side channel analysis attacks, the side channel is detected by a power measurement circuit or by the electromagnetic probe and the output voltage is captured by the digital sampling oscilloscope. The oscilloscope samples the analog input voltage and stores the digital samples in its memory. The analog-digital conversion is characterized by three basic parameters: input bandwidth, sampling frequency and resolution. The power consumption was measured on a 10 Ω serial resistor inserted in the power line of the microcontroller. The output signal of this test circuit was recorded by a LeCroy WavePro 7200A oscilloscope, its characteristics are listed in Tab. 2. The microcontroller generates a trigger signal to help the oscilloscope recognize the start of the encryption. First, we loaded an AES implementation in the C language into the microcontroller. A few blocks were sent through RS-232 interface from computer so that computation corectness could be proven. Then we could indentify the time sector, in which an encryption was computed. Each sample was stored in a distinct file for further analysis in MATLAB.

In a first phase were always compared two samples from a set of 1000 samples. This comparison was made to test the oscilloscope settings, if they are eligible for the attack. The sampling frequency was set to 200 kS/s, the file size was 10 kB, the secret key and the plaintext were always the same. The result of this test was that the drift of the internal clock signal of the microcontroller is considerable very unpredictable. Sometimes one tact lasts longer, sometimes shorter, so the samples cannot be aligned to some position, in which they fit together with the time axis.

In Fig. 1a we can see that the green sample is a few time segments before the red sample. In Fig. 1b it is inverted and also in other time segments it is impossible to say how the traces will be aligned. The blue trace is an average of these two samples. In a case of a thousand or even one hundred samples, it is practically a line with no higher peaks.

We used the external oscillator to eliminate these unwanted characteristics. A silicon crystal served well for a precise tact length and accurate starting time. The next test proved that the usage of internal oscillator was responsible for the bad traces alignment.

3 ATTACK STRATEGY

When performing the differential power analysis, there exists a general strategy, which can be used for the attack. After the successful connection of the oscilloscope to the cryptographic device, adjusting the optimal settings and selecting the correct part of the sample for recording, there is a sequence of steps, which should be followed in order to make the attack successful.

In the first step, we needed to prepare the sufficient amount of plain-text blocks, which were processed by the device. The oscilloscope was triggered so that it started recording in the beginning of the encryption. When the

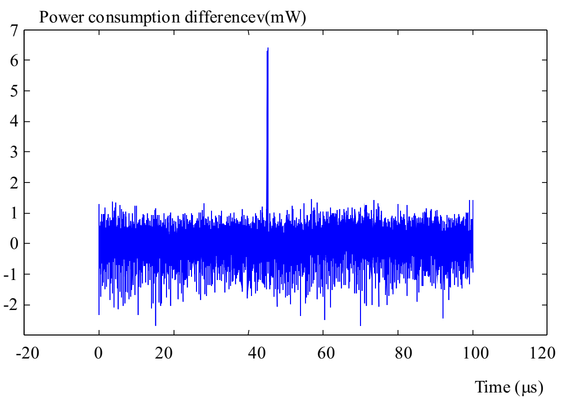

Fig. 2. Dependency on the first bit of the plain-text

encryption was finished, the recording stopped and the captured trace was saved into a file. The next sample was processed in the same way.

In the second step we analyzed the recorded samples in prepared programs. We wrote a program in Java for easy division of samples. It takes the plain-text blocks and captured samples as an input and it computes the hypothetical intermediate value of the targeted bit by using all 256 key hypotheses. The targeted bit in this case is always the first bit of the first byte. The intermediate value is a point in the algorithm, which depends on the plain-text and also on a concrete byte of the secret key. Then it makes the list which contains the name of the sample file and an attribute, whether the computed bit value is one or zero for every key hypothesis and for every sample file.

In the third step we needed to process the samples divided into groups by the statistical program in a way that it showed us the wanted result. So we wrote the program in MATLAB environment which took files with the division lists and the captured samples as an input. Then it computed the average of each group of the samples, so we got two average power traces. One shows us the power consumption in a case the intermediate bit value was one and the other shows us the case in which it was zero. Then it subtracted these two traces and we could see, whether there are significant peaks in the subtracted graph or not. It created the subtracted graphs for every 256 key byte hypotheses and it saved the maximal value of each graph.

In the last step we had maximal values for every key byte and we could determine the secret key by the visual inspection of the graphs with these values. If the maximal value is separate and no other key byte hypothesis has the same or similar value, then the result is clear. If there are more maximal values or the maximal value is not very significant, it means that we were unable to discover the key byte with these settings.

4 PLAIN–TEXT DEPENDENCY TESTS

We made one thousand of samples containing different plain-text and using the same secret key for the first test, so the dependence of the consumed power on the plain-text should be proven. The sampling frequency was set to 100 MS/s, because it provided optimal results — the accuracy was sufficient for the attack and the sample size was 100 kB, so it was comfortable to work with them.

The ideal way to prove the dependency on the plain-text is to focus on some bit of the message. So the first test aimed at the first bit of the first byte of the plain-text. The samples were divided into two groups. In the first group were samples, which contained the bit value one on this position in the plain-text and in the second group there were the others, with the bit value zero. The other values in the plain-text were neglected, so they randomly consisted of ones and zeroes and they should not affect the consequential power graphs. This test helped to visualize if it is possible to execute a differential power analysis with the chosen settings on given devices. Figure 2 shows that that the dependency on the plain-text is clear. The x-axis displays the time division, 100 μs in this case, and the y-axis shows the current in volts.

There is a maximal value approximately between 40 and 50 μs, which shows us the dependency of the consumed power on input values sent to the microcontroller. Other tests were also performed for the validation of these results. The dependency on the other bits of the plain-text was proven and also the random division of the samples was made and there were no significant maximal values.

5 ATTACK TESTS WITH THE AES IMPLEMENTATION IN THE C LANGUAGE

Considering the good results of the previous tests it was possible to continue with the attack. In this phase it was necessary to deliberate the concrete algorithm used in the encryption process and to choose appropriate intermediate values, which will depend on the used secret key.

The first operation in the AES algorithm is the AddRoundKey (plaintext is binary xor-ed with the first round key). The operation is carried on sequentially, so firstly the first byte of plaintext is xor-ed with the first byte of the key to another byte, which is then further transformed by the encryption process. After this operation is applied the non-linear byte substitution with a use of the substitution table (S-box), cyclic shift of matrix rows and a mix of the matrix columns. After these operations is applied the round key and then is this process applied again, nine more times (the last round does not include the mix columns operation). So the best way to determine the dependency on the secret key is to choose the state after the S-box substitution [4]. From the higher level we can view on the substitution as on the navigation in a substitution table, in which the first hexadecimal value of the byte indicates the table row and the second one indicates the table column. This mechanism is showed on Fig. 3.

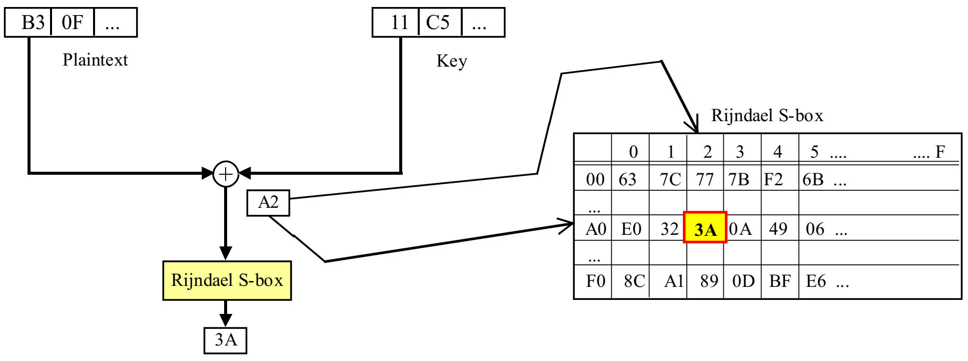

Fig. 3. Xor operation and the S-box substitution

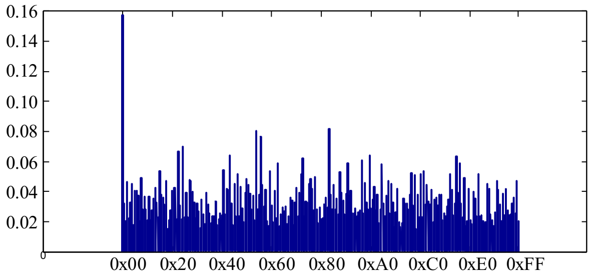

Fig. 4. Maximal values of each of the 256 key hypotheses

Table 3. Attacks with the various number of samples

Number of samples	Wrong guesses	Success rate	Time eduction
700	16	0%	30%
800	7	56.25%	20%
850	0	100%	15%

Our program written in MATLAB environment used for the purpose of the attack text files containing the specification on how to divide the samples as the first input and binary files with the samples as the second input. The analysis was carried on an Intel Core i7 920 computer with 6GB of operating memory. The time necessary to analyze 256 key hypotheses was approximately 4.5 hours, that means the whole 128b key was possible to discover after 3 days.

6 ATTACK TESTS WITH THE AES IMPLEMENTATION IN THE ASSEMBLER

The next tests were performed on the AES implementation written in the assembler. Programming in assembler gives much more space for the optimization. The encryption speed got higher, the whole process took about 300 μs, so it was possible to capture a bigger part of encryption using the same sample rate. The attack methodology was the same, so the samples were also divided in accordance with the most significant byte of the S-box output.

After the analysis for every key hypothesis of the 16-byte secret key, we however discovered that the random noise was in the most cases higher than the right key guess. So for the next tests we choosed a sampling frequency 1 GS/s and to keep the same sample size, the captured time period was reduced to 100 μs. With these settings it was possible to reveal 12 of 16 bytes of the secret key, every fourth byte was guessed incorrectly.

The problem was eliminated by narrowing the time interval. The analysis of the particular samples showed that the S-box substitution is done between 3 and 12.5 μs, so by analyzing only this interval it is possible to reduce the time needed to compute the maximal values of the key guesses and also to eliminate the random noise from other parts of the sample.

The attack optimized by choosing the time interval was done in 16 minutes for one key byte, so the whole key was discovered in 4 hours and 16 minutes.

The next optimization was to determine the minimal number of samples needed for the attack. All successful previous attacks were done with 1000 samples. The first attack with reduced number of samples included 700 samples. It was unsuccessful on every tested byte, so the 800 samples was chosen for the next test. In this case, the majority of key bytes was identified correctly, however 7 bytes from the 16 byte key were still guessed wrongly. The next test, which was the last, analyzed 850 samples and it showed that this number is enough to reveal the entire key when the time interval is reduced to the region of S-box substitution. The results are summarized in Tab. 3.

7 CONCLUSIONS

In this paper we presented results of practical tests of the differential power analysis on the AES algorithm, implemented on the particular microcontroller in the C language and in the assembler.

The results were positive, the attack was performed in laboratory conditions and it revealed all 16 bytes of the secret key. The time needed for the attack was reduced to 3 hours and 37 minutes with a use of a personal computer, which is a negligible time in comparison with a brute force attack on this algorithm.

For the further work it would be valuable to try the attack without the auxiliary trigger signal, however the time shift of the samples in this case is a nontrivial problem. There are some works describing an attack on misaligned samples [4], or trying to find the method which will help to align them [5].

References

[1] BREIER, J. : Differential Power Analysis of Rijndael Operations on a Selected Microcontroller, FI MUNI, Brno, 2010.

[2] DAEMEN, J.—RIJMEN, V. : The Design of Rijndael: AES – The Advanced Encryption Standard, Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[3] KOCHER, P.—JAFFE, J.—JUN, B. : Differential Power Analysis, Springer-Verlag, 1999, pp. 388–397.

[4] MANGARD, S.—OSWALD, E.—POPP, T. : Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security), Springer-Verlag, New York, Seceacus, 2007.

[5] van WOUDENBERG, J. G. J.—WITTEMAN, M. F.—BAKKER, B. : Improving Differential Power Analysis by Elastic Alignment, research paper (2008).

Received 19 January 2011

Jakub Breier received his Degree in Information Security from the Masaryk University, Brno in 2010. Currently he is doctoral student at Faculty of Informatics and Information Technologies, STU in Bratislava. His main fields of interest are risk analysis and security mechanisms.

Marcel Kleja received his Degree in Electrical Engineering from the Slovak University of Technology, Bratislava in 1996. Currently he works for the National Security Authority. His main fields of interest include implementation of cryptographic mechanisms.