2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA) | 978-1-6654-0127-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICAACCA51523.2021.9465337

Side Channel Attack Countermeasure for Low Power Devices with AES Encryption

Ruminot-Ahumada, Nicolás; Valencia-Cordero, Claudio; Abarzúa-Ortiz, Rodrigo

Abstract—**The advancement of Internet of Things produces a massive increase in the use of low-power devices, which can contain sensitive information. Most of these devices do not have the necessary security to protect their information. Therefore, the work aims is analyze and compare countermeasures for SCA in low-power devices with 128-AES encryption. In this scenario, we analyze existing countermeasures and conclude that algorithm-based countermeasures are more suitable for low-power devices. Subsequently, we run a set of tests to understand the scope of CPA attacks and thus establish a countermeasure that is tailored to these devices. Finally, we propose a countermeasure based on byte logic, comparing it with traditional countermeasures of the same type.**

Keywords—**Cryptography, Internet of Things, Side-channel attacks.**

I. Introduction

THE exponential increase in technologies has occurred steadily in recent decades. Technologies such as the Internet of Things (IoT) have emerged, one of which is expected to grow exponentially in the coming years [1], [2]. The use of Smart Cards and other embedded technologies have also been part of the exponential growth. Most of the devices found in IoT and embedded systems use microcontrollers. These devices handle all kinds of information, from irrelevant information to images, health records, physical access to places, temperature control, among others.

Due to the aforementioned there is also an increase in security problems for these devices (microcontrollers), which is why the need to keep this information safe arises. For many years, the traditional way to protect information is by encrypting it. Today, encryption offers a more than sufficient level of security against analytical attacks. However, a little over twenty years ago the cryptographic community encountered a type of attack that under certain conditions is capable of breaking cryptographic secrets in very short times. These types of attacks are known as implementation attacks.

Among the implementation attacks we find a type of attack called Side Channel Analysis (SCA), these attacks make use of a physical “leak” of information (acoustic changes, execution time, energy consumption, electromagnetic radiation [3], among others [4]). This information leak occurs when encrypting or decrypting and is related to the cryptographic secret. The first SCA attack was introduced by P. Kocher in [5] where a technique called as Simple Power Analysis (SPA) is applied. The SPA takes energy consumption samples of a device while it performs the encryption process, these energy samples are then analyzed in order to establish the secret key. Several useful SPA attack methods have been developed depending on the device, encryption and the information that is available as explained in [6]. Two years later, another SCA technique is developed that uses statistical analysis in energy consumption measurements, this technique is called Differential Power Analysis (DPA) [7]. This one manages to find the secret key in a very short time as shown in [8] where it is possible to recover the key in 2 hours and 30 minutes (with the advancement of technology these times have decreased even more). Finally, one of the most popular and powerful SCA techniques are the well-known Correlation Power Analysis (CPA) [9]. This technique uses Pearson’s correlation coefficient to obtain the degree to which hypothetical keys are related to the energy consumption samples, obtaining the correct secret key in less than one minute.

In this article an analysis of countermeasures for SCA in low-power devices with AES 128 encryption is performed. In addition, a countermeasure based in byte logic never used before is presented in order to homogenize energy consumption, making it more difficult to recover the secret key.

In section II some concepts for the understanding of the work are presented. In section III an analysis of countermeasures is performed on low-power devices limited to 8 and 16-bit architectures. Also, an analysis on correlation attacks is done. Later, in section IV the design and implementation of the proposed countermeasure is explained using a set of test. Finally, in section V some conclusions are presened based on the analysis and test results.

II. Background

A. Energy Consumption Model

Power consumption in transistor with Complementary Metal-Oxide-Semiconductor technology (CMOS) has two main components, a static component and a dynamic component [10], [11]. The static component, called static power or leakage power, is produced by the current flowing through the transistor while it should not conduct and its formula is as follows, where V_{dc} is the continuous voltage of the source and I_{leak} is the leakage current:

$$P_{leak} = I_{leak} \cdot V_{dc} \tag{1}$$

Ruminot Nicolas and Valencia Claudio is with the University of Santiago of Chile (USACH), Electrical Engineering Department, Chile (e-mail: nicolas.ruminot@usach.cl, claudio.valenciac@usach.cl).

Abarzúa Rodrigo, is with the University of Santiago of Chile (USACH), Mathematics and Computer Science Department, Chile (e-mail: rodrigo.abarzua@usach.cl).

On the other hand, the dynamic component, known as dynamic power, is generated by the change of state of the transistor. Dynamic power has two components, the power released by capacitance and the short-circuit power. The power released by the capacitance occurs because each node in a CMOS circuit is associated with a capacitance C. When the transistor changes state, an amount of energy is dissipated by charging the capacitance of the respective node. This power is represented by the following equation, where α is the activity factor, f is the switching frequency, C is the load capacitance and V_{dc} is the continuous voltage of the source:

$$P_{cap} = \alpha \cdot f \cdot C \cdot {V_{dc}}^2 \tag{2}$$

Finally, the short-circuit power comes from the change of state of the transistor. For a short time, current flows directly from the source to the ground. This power is represented by the following equation, where I_{max} is the maximum current flowing during the "short circuit"" and t_{sc} is the time interval during which the "short circuit" occurs:

$$P_{sc} = \alpha \cdot f \cdot V_{dc} \cdot I_{max} \cdot t_{sc} \tag{3}$$

The sum of all these powers results in the total power of a CMOS transistor.

$$P_{total} = I_{leak} \cdot V_{dc} + \alpha \cdot f \cdot C \cdot {V_{dc}}^2 + \alpha \cdot f \cdot V_{dc} \cdot I_{max} \cdot t_{sc} \tag{4}$$

B. Advanced Encryption Standard

Modern ciphers can be divided into two categories, symmetric and asymmetric ciphers. Of the symmetric ciphers, the most widely used cipher today is Advanced Encryption Standard (AES). This encryption processes plain texts of 128-bits and uses secret keys of 128, 192 and 256 bits depending on the level of security desired. Depending on the number of bits of the secret key, a part of the encryption is executed cyclically a certain number of times. For a 128-bit secret key the encryption will perform 10 iterations. This encryption works byte by byte, this means that it can modify the bits of one byte at the time. [12]. Fig. 1 shows the flow diagram for an AES 128 encryption.

It can be seen that AES 128 encryption has four sub-functions, *AddRoundKey()*, *SubBytes()*, *MixColumns()* y *ShiftRows()*. Next, a brief explanation of each sub-function is given below:

- *AddRoundKey():* Performs an XOR operation between the bytes of the state matrix and the bytes of the secret key corresponding to the iteration in which they are.
- *SubBytes():* This function adds non-linearity to the encryption and consists of a substitution of each of the 16 bytes of the state matrix through a look-up table. Look-up tables are fixed data tables, used to replace one value with another. In the case of AES, a $16x16$ table is used where each box has 1 byte of data in hexadecimal.
- *ShiftRows():* This function generates the rotation of the rows (the bytes pass from left to right) of the state array. The first row does not rotate, the second rotates one byte, the third rotates two bytes, and the fourth rotates three bytes.
- *MixColumns():* It consists of multiplying each of the columns of the state matrix by a fixed polynomial.

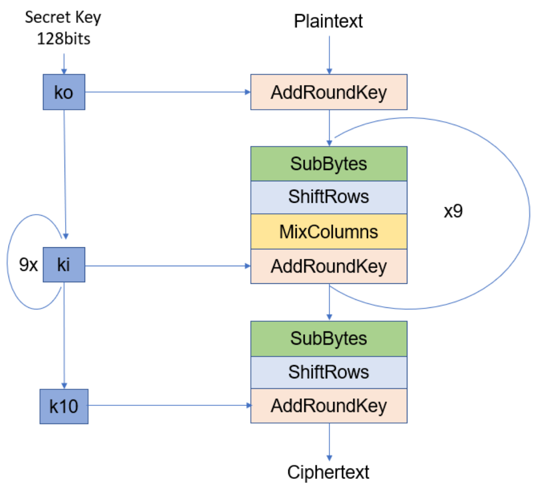


Fig. 1. Flowchart for AES 128 encryption.

C. Correlation Power Analysis

Among the implementation attacks we find several types of SCA, these differ between them by how they use the information leak to attack a device. To carry out any type of SCA based on energy consumption, you must have samples of the energy consumption of the device that is being attacked while it is encrypting or decrypting. Within the SCA attacks, there is one of the most powerful attacks against symmetric ciphers. These attacks are known as correlation attacks or CPA [9]. This attack makes use of statistical analysis to break a cryptographic secret and supports two hypotheses, first that logical changes from 0 to 1 or from 1 to 0 consume energy, and second that such changes require the same amount of energy.

To carry out a CPA, you must choose a place in the encryption to attack and around this generate a model of how energy consumption should vary with different inputs. Then, it is observed how closely related the hypothetical consumption patterns are with the measured consumption traces. This relationship is quantified through Pearson's correlation coefficient. Pearson's correlation coefficient is particularly useful as it tells us how closely related two variables x e y are. The function used is:

$$\rho_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sqrt{E[(x-\mu_x)^2]E[(y-\mu_y)^2]}} \tag{5}$$

Where, $cov(x,y)$ is the covariance between x and y, σ_x and σ_y are the standard deviation of x and y respectively, E is the expected value and μ_x, μ_y is the mean of x and y respectively [9]. Once it has been determined which hypothetical consumption model is more related to consumption traces (the one with a higher coefficient), the most probable secret key is obtained.

D. Countermeasures

Countermeasures for SCA can be classified into three types, hardware countermeasures, algorithm countermeasures, and protocol countermeasures.
Protocol countermeasures are not analyzed in this investigation. Hardware countermeasures use hardware modifications to reduce or eliminate information leaks. Algorithm countermeasures do not require modifications to the hardware, instead a modification is applied to the encryption algorithm without altering its output.

III. Countermeasures Analysis

To understand countermeasures, you must first understand how information leaks occur. Every time the value of a memory register is manipulated, its value is handled through the data buses. These data buses are basically a group of transistors that are responsible for keeping an output high or low (logical 1 or 0 respectively). As mentioned above, each transistor has an associated energy consumption, where the majority of energy consumption is due to the switching of the device. Every time the data bus changes its state, the transistors are changing their state. This is the reason why devices that use CMOS technology are prone to SCA attacks based on power consumption, since observing the power consumption and analyzing it can determine how many transistors have changed and when.
Next, an analysis and comparison of the most common hardware and algorithm countermeasures was performed. The main characteristics considered in the countermeasures are that it does not affect the functionality of the device and that it is capable of protecting the cryptographic secret in low-power devices. The functionality of low consumption devices is to operate correctly under normal conditions, that is, to have low energy consumption, to execute the processes in the expected times, to be able to execute the processes and to maintain a reduced size of the device.

A. Hardware Countermeasures

Among the Hardware countermeasures are very varied countermeasures, some make use of complemented logics [13], [14], [15], semi independent circuits to generate noise [16], partitioned processors and memories, clocks with irregular frequency [17], [18], reduction in signal power [19] among many others. All these modifications imply an improvement in the security of the devices against the majority of SCAs regardless of their encryption. This is relevant as not all types of countermeasures protect against more than one type of SCA. Furthermore, hardware countermeasures include most of the countermeasures with the best security performance. Despite this, all of these are characterized by increasing the size of the device, increasing energy consumption and execution time. Compared to an unprotected device, all the mentioned parameters increase at least two or three times as concluded in chapter eleven of [4].

B. Algorithmic Countermeasures

These countermeasures involve a modification to the encryption algorithm in order to reduce, eliminate or hide the relationship between the energy consumption samples and the processed data. These can be classified into three types, independent of the algorithm, semi-dependent of the algorithm and dependent on the algorithm. Algorithm independent countermeasures usually perform random operations in random places during the encryption process to make the trace of energy consumption more difficult to analyze, all these countermeasures can be applied on devices regardless of what encryption they use. Semi-dependent countermeasures are characterized by modifying parts of the encryption, such as the order in which some operations are executed or repeating operations among others [20], [21]. Some of them can be applied in various types of encryption, always with prior knowledge of the algorithm. Lastly, countermeasures dependent on the encryption algorithm require a good understanding and management of encryption since larger modifications are made or sometimes focused on a specific part of the encryption, which if not implemented correctly will harm the result of the encryption process [15].
The main losses that occur when implementing an algorithmic countermeasure is an increase in execution time and in the size of the code. Another drawback that these countermeasures have is that very few are effective against CPA attacks.

C. Countermeasures for low-power devices

Low-power devices are characterized by low energy consumption, limited execution times, and a small size. This set of features is the functionality of these devices. Furthermore, in this research low-power devices are limited to devices with 8 and 16 bit architectures (so countermeasures like Double-width Algorithmic Balancing [15] or TBox presented in chapter eleven of [4] could not be implemented as it requires a 32 bit data bus).
In summary, considering the main characteristics of hardware countermeasures, these are considered inappropriate for low power devices. Determining then that in these devices an algorithm countermeasure would be more appropriate.

IV. Design and Implementation of Countermeasures

This section presents the characteristics of the implementation of the CPA attack, we use as example an attack on a low-power device without countermeasure. Futhermore, an analysis of the attack is performed in order to understand which countermeasures are useful. Later, the design of an algorithm countermeasure for low power devices is explained. Finally, the designed countermeasure is submitted to tests, determining how practical its implementation is with respect to other algorithm countermeasures.

A. CPA attack on a device protected with AES 128

As mentioned, in CPA attacks models are generated of how, hypothetically, energy consumption should vary with different

inputs. This hypothetical model is generated with respect to 8 bits, because the AES encryption works byte by byte, a large part of the energy consumption that occurs due to the changes in the state of the transistors is directly related to the changes that are made in the bytes during the encryption process and therefore to the cryptographic secret.
The implementation of the attack was carried out on a microcontroller ATXMega128A1U-AU [22] protected with AES encryption algorithm with 128 bits of security arranged on one card CW303 [23], and a ChipWhisperer Lite device [24] for taking energy samples. The programs used for the capture and subsequent analysis of the samples are *CWCapture* and *CWAnalyzer* respectively. Finally, the parameters for the sampling were the following: $1,250$ offset points, 4000 points per trace, 50 traces and the results of the attack are observed in Fig. 2. In the top row the bytes with the highest correlation magnitude after analysis are shown. For each byte there is a column of the succeeding bytes in correlation magnitude. The bytes of the secret key will always be in red, while the darker the green of the boxes means that there is more difference in correlation with the byte that precedes it.

In summary, the results obtained are: The time required to take samples (T1) was 5 seconds, the time required to analyze the samples (T2) was 10 seconds, the total attack time (Tt) was of 15 seconds, the number of secret key bytes found was 16. In addition, it was observed that the weight of the encryption code (without decryption) is 2979 bytes and that encrypting 16 bytes of plaintext requires 43120 clock cycles.

B. Analysis of CPA

The CPA attack can be carried out at various points of the AES encryption, but without a doubt the most vulnerable point in this encryption is the output of the function *SubBytes()* that as explained in the section II corresponds to a substitution of values through a look-up table. In general, hypothetical consumption models try to predict the values that energy consumption can take at a certain point in the encryption process. In order to check how powerful the attack is, the following test was carried out: The look-up table was modified, leaving only the four least significant bits of each byte and replacing the four most significant bits with 0's, that is, 4 bits remain of original information. Subsequently, the replacement table is modified again, leaving only 3 bits of information, then 2 and finally 1. All these ciphers with modified replacement tables were attacked by a CPA. The results obtained reveal that with a correlation analysis it is possible to break a cryptographic secret even with 1 bit of information per byte. In the Table I the results obtained are found, where the number of traces corresponds to the number of traces necessary to find the 16 bytes of the secret key.

Additionally, another similar experiment was carried out where two CPAs were executed. This time two bits of original information per byte were kept in the replacement table, first two adjacent bits were used and then two interleaved bits. The results are in Table II and show that two adjacent information bits produce more information leakage than two interleaved bits.

TABLE I
CORRELATION TEST RESULTS.

Number of bits	Number of Traces
4	120
3	210
2	330
1	4100

TABLE II
RESULTS OF CORRELATION ANALYSIS IN ADJACENT AND INTERLEAVED BITS.

Two Bits of Information	Number of Traces
Adjacent	330
Interspersed	510

C. Countermeasure Design

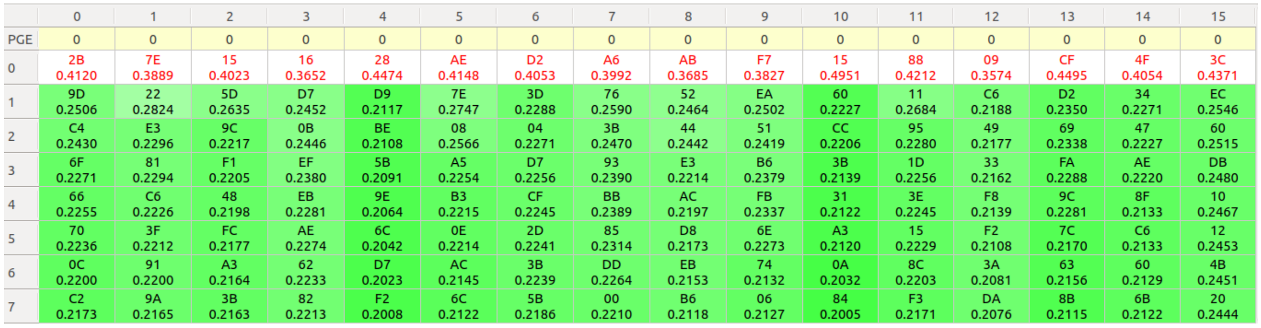
Taking into consideration the correlation tests carried out previously, we proceeded to design an algorithm countermeasure programmed in C language. This countermeasure separates the information processed by the encryption and works on it almost independently. In addition, it makes use of complemented and repeated logic with the intention of homogenizing energy consumption.
The idea is to separate both the plaintext and the secret key, leaving the information of each byte of plaintext in two complemented bytes and the information of each byte of the secret key in two bytes of repeated logic (this process can be observed in the Tables III and IV respectively). Once the information is separated, we proceed to work as long as possible with the "left" bytes separated from the "right" ones. Also, the replacement table used in the function *SubBytes()* is replaced by four equivalent look-up tables in which two of them have bytes with complemented logic and the other two with repeated logic (between the two tables of complemented logic they contain the same information as the original replacement table, the same happens with the tables of repeated logic). In Fig. 3 the flow chart of the designed countermeasure is shown. In the following subsections, the logics used and their properties are detailed.

1) Logical Complement: This logic was designed in order to take advantage of the reduction of information leakage by having bits interleaved. A byte of complemented logic is a byte where every two adjacent bits we find a 1 and a 0 logic. Also, only 4 of the 8 bits of each byte contain normal byte information. Table III shows an example of how each complemented byte stores half the information of a normal byte. It can also be inferred that the complemented bytes always have a Hamming Weight equal to four and that there are only 16 different complemented byte.

TABLE III
LOGICAL COMPLEMENT EXAMPLE.

Byte Type	Bits								HW	Hex
Normal	1	1	0	1	0	1	1	0	5	D6
Com. Left	1	0	0	1	0	1	1	0	4	96
Com. Right	0	1	0	1	0	1	1	0	4	56

2) Logical Repeated: The Logical Repeated consists in that every two bits of information will always be found either two



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PGE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2B 0.4120	7E 0.3889	15 0.4023	16 0.3652	28 0.4474	AE 0.4148	D2 0.4053	A6 0.3992	AB 0.3685	F7 0.3827	15 0.4951	88 0.4212	09 0.3574	CF 0.4495	4F 0.4054	3C 0.4371
1	9D 0.2506	22 0.2824	5D 0.2635	D7 0.2452	D9 0.2117	7E 0.2747	3D 0.2288	76 0.2590	52 0.2464	EA 0.2502	60 0.2227	11 0.2684	C6 0.2188	D2 0.2350	34 0.2271	EC 0.2546
2	C4 0.2430	E3 0.2296	9C 0.2217	0B 0.2446	BE 0.2108	08 0.2566	04 0.2271	3B 0.2470	44 0.2442	51 0.2419	CC 0.2206	95 0.2280	49 0.2177	69 0.2338	47 0.2227	60 0.2515
3	6F 0.2271	81 0.2294	F1 0.2205	EF 0.2380	5B 0.2091	A5 0.2254	D7 0.2256	93 0.2390	E3 0.2214	B6 0.2379	3B 0.2139	1D 0.2256	33 0.2162	FA 0.2288	AE 0.2220	DB 0.2480
4	66 0.2255	C6 0.2226	48 0.2198	EB 0.2281	9E 0.2064	B3 0.2215	CF 0.2245	BB 0.2389	AC 0.2197	FB 0.2337	31 0.2122	3E 0.2245	F8 0.2139	9C 0.2281	8F 0.2133	10 0.2467
5	70 0.2236	3F 0.2212	FC 0.2177	AE 0.2274	6C 0.2042	0E 0.2214	2D 0.2241	85 0.2314	D8 0.2173	6E 0.2273	A3 0.2120	15 0.2229	F2 0.2108	7C 0.2170	C6 0.2133	12 0.2453
6	0C 0.2200	91 0.2200	A3 0.2164	62 0.2233	D7 0.2023	AC 0.2145	3B 0.2239	DD 0.2264	EB 0.2153	74 0.2132	0A 0.2032	8C 0.2203	3A 0.2081	63 0.2156	60 0.2129	4B 0.2451
7	C2 0.2173	9A 0.2165	3B 0.2163	82 0.2213	F2 0.2008	6C 0.2122	5B 0.2186	00 0.2210	B6 0.2118	06 0.2127	84 0.2005	F3 0.2171	DA 0.2076	8B 0.2115	6B 0.2122	20 0.2444

Fig. 2. Result of the attack on a device with AES 128 encryption after analyzing 50 traces. The image was obtained with the CWAnalyzer software.

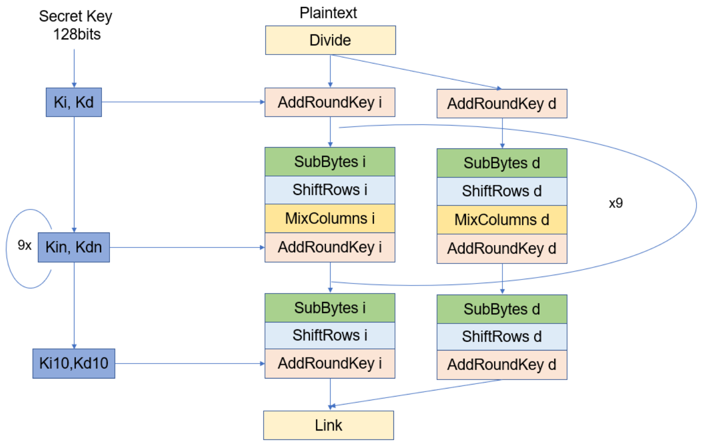


Fig. 3. Flowchart for an AES 128 encryption with the designed countermeasure.

1's or two 0's. In the Table IV the operation of this logic is observed in more detail. It is also observed that the Hamming Weight of the bytes with this logic will always be 0, 2, 4, 6 or 8 and that there are only 16 bytes with this logic.

TABLE IV
LOGICAL REPEATED EXAMPLE.

Byte Type	Bits								HW	Hex
Normal	1	0	0	0	1	1	1	1	5	D6
Rep. Left	1	1	0	0	1	1	1	1	6	CF
Rep. Right	0	0	0	0	1	1	1	1	4	0F

3) Logical Complemented and Logical Repeated Properties: By using the exclusive-or operation with the exposed logics, you find useful results to apply in AES encryption. Here are some useful properties:

1) When performing an exclusive-or operation between two bytes of repeated logic, the resulting byte will always have repeated logic.
2) When performing the exclusive-or addition between two bytes with complemented logic, the resulting byte will always have repeated logic.
3) When performing the exclusive-or sum between a byte of logic complemented with one of repeated logic, the resulting byte will have complemented logic.

The following is a summary of the modifications made to the AES sub-functions and the added sub-functions:

- *Divide()*: This function splits both the plaintext bytes and the secret key bytes. For this, each byte of plaintext is separated into two bytes with complement logic as shown in the table III finally obtaining 32 bytes that contain the hidden information of the original 16 bytes. These 32 bytes are separated into two groups of 16 bytes (one group with the left bytes and the other with the right bytes). The left 16 bytes are called the left state matrix and the right ones are called the right state matrix. The same process is performed to the secret key bytes but transforming them into bytes with repeated logic as seen in the table IV getting a left and a right secret key.
- *Link()*: This function takes the information from the bytes of the left and right state matrix uniting them into a normal byte.
- *AddRoundKey()*: Performs XOR operation between the complemented and repeated logic bytes (left-to-left and right-to-right bytes).
- *SubBytes()*:In this function instead of having a single replacement table, now there are two. Each byte of these tables is generated through the process explained in the Table III, so one table will contain the information ”left” and the other the ”right”. Then the bytes from the left state array use the left replacement table and the bytes from the right state array use the right replacement table. It should be mentioned that in this function it is impossible to work completely independently the bytes of the left state matrix with those of the right.
- *ShiftRows()*: This function can be applied without major modifications directly to both the left and right state matrix.
- *MixColumns()*: In this function both the left and right matrix are multiplied by fixed polynomials obtained from the separation of the original fixed polynomial. In this separation, fixed polynomials are obtained with repeated logic. Then the left bytes are worked as independently as possible from the right ones.

D. Test setup

Once the designed countermeasure has been programmed and implemented, a CPA attack is performed in order to verify its effectiveness. The measurement parameters used were 3500 offset points, 7000 points per trace, and 30000 traces. The results obtained were: Time required for sampling

(T1) 43 minutes 20 seconds, time required for analysis of samples (T2) 1 hour 26 minutes and 40 seconds, total time (Tt) equal at 2 hours 8 minutes and 20 seconds, 93768 clock cycles are required to encrypt 16 bytes of plaintext, the weight of the code is 5047 bytes and as seen in the Fig. 4, 14 bytes of secret key were retrieved successfully. In the table V the summary of the results of the attacks on an AES 128 encrypted microcontroller without countermeasure and with countermeasure is shown.

TABLE V
RESULTS TABLE FOR CPA ATTACK.

Parameters	Without Countermeasure	With Countermeasure
Offset Points	1.250	3.500
Points per Trace	4.000	7.000
Weight in Bytes	2.979	5.047
Clock Cycles	43.120	93.768
Time 1	00:00:05	00:43:20
Time 2	00:00:10	01:26:40
Total time	00:00:15	02:10:00
Correct Bytes	16	14
Numbers of Traces	50	30.000

The results of the countermeasure show that it adds security, however, it is still possible to break the cryptographic secret through a CPA attack (with a higher number of traces). During the experiment it was noted that the countermeasure uses 1684 bytes of RAM, this is a limitation for devices that have RAM equal to or less than 2Kb, since stability problems could occur. Also, like most algorithm countermeasures, the biggest cost to this countermeasure is increases in memory usage and execution times.

To compare this countermeasure with other algorithm countermeasures a few things need to be clarified. There are combined attacks that use CPA attacks only to find the group of bytes most likely to be part of the secret key and then perform a brute force attack with these bytes. Assuming only one CPA attack is performed on devices with common countermeasures like Dummy Instructions (when applied as a double run with a fake secret key) or Shuffling, a CPA attack will never find all the bytes of the secret key complete or in the correct order. However, all the bytes of the secret key fall within the most likely 32 bytes, so a brute force attack with those 32 bytes would hit the correct secret key fairly quickly (only requires a CPA with about 250 traces and a brute force attack with the possible combinations of the 32 bytes). On the other hand, if you want to find the 16 bytes of secret key to a device with the designed countermeasure, you need a CPA attack of more than 30000 traces or a CPA of 30000 traces plus a brute force attack with combinations of 80 possible bytes (as shown in Fig. 4 secret key bytes that were not found were five and four positions away from being the most likely).

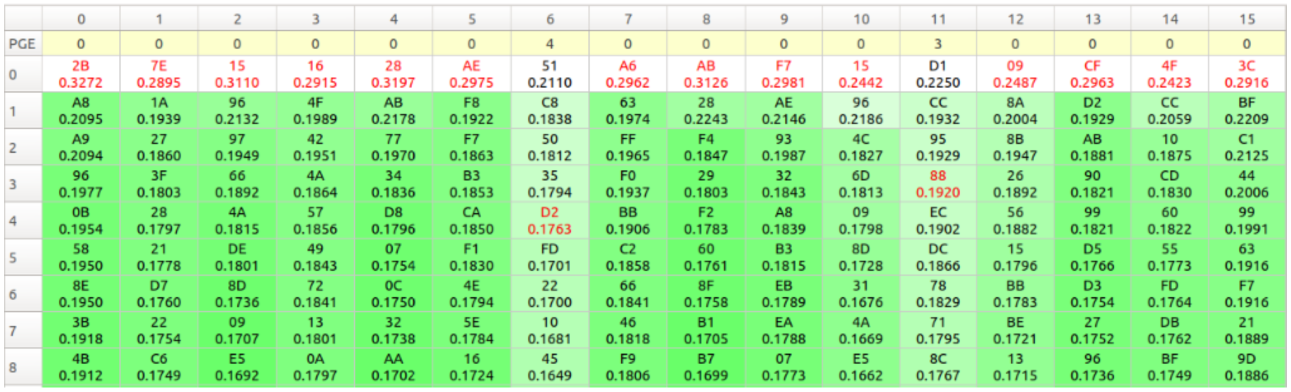
It is worth mentioning that the amount of memory of the countermeasure varies slightly depending on the compiler used, for this investigation the compiler *WinAVR 20100110* provided by the Chipwhisperer Lite device was used. The sampling and analysis times also depend on the device with which the analysis is made, but since the same device was used for all the attacks, these values are comparable. Finally, the increase in memory required and clock cycles can be optimized by improving the programming in C language of the encryption, despite this, the increase in clock cycles will remain close to twice that of an encryption without countermeasure, since the countermeasure performs approximately twice as many operations.

V. CONCLUSION

The research shows how powerful a correlation analysis is, since even with only 1 bit of information per byte it is possible to recover the secret key. This shows that no matter what modification is applied to the encryption algorithm, it will always have information leaks that can be exploited by SCA.

Regarding the designed countermeasure, it is concluded that it adds security in a significant amount and provides a new way of hiding the relationship between energy consumption and processed data. However, almost all countermeasures for SCA are situational. This means that it depends on which device you want to protect and against, for example, if you want to protect only a device with 2Kb of RAM against CPA, it is not recommended to use this countermeasure despite providing greater security than a common countermeasure.

It is concluded that to find a global countermeasure to the problem of SCA it must be done through modifications to the hardware. Unfortunately, the size and requirements of some technologies do not yet provide the ability to implement hardware countermeasures on them. It is expected that with the advancement of technology, we are getting closer and closer to finding a hardware countermeasure that offers protection against SCA with 100% effectiveness and is implementable in low-power devices.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PGE	0	0	0	0	0	0	4	0	0	0	0	3	0	0	0	0
0	2B 0.3272	7E 0.2895	15 0.3110	16 0.2915	28 0.3197	AE 0.2975	51 0.2110	A6 0.2962	AB 0.3126	F7 0.2981	15 0.2442	D1 0.2250	09 0.2487	CF 0.2963	4F 0.2423	3C 0.2916
1	A8 0.2095	1A 0.1939	96 0.2132	4F 0.1989	AB 0.2178	F8 0.1922	C8 0.1838	63 0.1974	28 0.2243	AE 0.2146	96 0.2186	CC 0.1932	8A 0.2004	D2 0.1929	CC 0.2059	BF 0.2209
2	A9 0.2094	27 0.1860	97 0.1949	42 0.1951	77 0.1970	F7 0.1863	50 0.1812	FF 0.1965	F4 0.1847	93 0.1987	4C 0.1827	95 0.1929	8B 0.1947	AB 0.1881	10 0.1875	C1 0.2125
3	96 0.1977	3F 0.1803	66 0.1892	4A 0.1864	34 0.1836	B3 0.1853	35 0.1794	F0 0.1937	29 0.1803	32 0.1843	6D 0.1813	88 0.1920	26 0.1892	90 0.1821	CD 0.1830	44 0.2006
4	0B 0.1954	28 0.1797	4A 0.1815	57 0.1856	D8 0.1796	CA 0.1850	D2 0.1763	BB 0.1906	F2 0.1783	A8 0.1839	09 0.1798	EC 0.1902	56 0.1882	99 0.1821	60 0.1822	99 0.1991
5	58 0.1950	21 0.1778	DE 0.1801	49 0.1843	07 0.1754	F1 0.1830	FD 0.1701	C2 0.1858	60 0.1761	B3 0.1815	8D 0.1728	DC 0.1866	15 0.1796	D5 0.1766	55 0.1773	63 0.1916
6	8E 0.1950	D7 0.1760	8D 0.1736	72 0.1841	0C 0.1750	4E 0.1794	22 0.1700	66 0.1841	8F 0.1758	EB 0.1789	31 0.1676	78 0.1829	BB 0.1783	D3 0.1754	FD 0.1764	F7 0.1916
7	3B 0.1918	22 0.1754	09 0.1707	13 0.1801	32 0.1738	5E 0.1784	10 0.1681	46 0.1818	B1 0.1705	EA 0.1788	4A 0.1669	71 0.1795	BE 0.1721	27 0.1752	DB 0.1762	21 0.1889
8	4B 0.1912	C6 0.1749	E5 0.1692	0A 0.1797	AA 0.1702	16 0.1724	45 0.1649	F9 0.1806	B7 0.1699	07 0.1773	E5 0.1662	8C 0.1767	13 0.1715	96 0.1736	BF 0.1749	9D 0.1886

Fig. 4. Result of the attack on a device with AES 128 encryption with countermeasure after analyzing 30,000 traces. The image was obtained with the CWAnalyzer software.

Acknowledgment

This work was supported by projects DICYT 061913VC and 62033AO, Vicerrectoría de Investigación, Desarrollo e Innovación. Universidad de Santiago de Chile.

References

[1] L. C. Karen Rose, Scott Eldridge, "LA INTERNET DE LAS COSAS — UNA BREVE RESEÑA," Internet Society, Tech. Rep., 2015.

[2] Cisco, "Wi-Fi 6 and Private LTE / 5G Technology and Business Models in Industrial IoT," Cisco, Tech. Rep., 2019.

[3] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM Side – Channel (s)," pp. 29–45, 2003.

[4] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*, springer ed., M. Tehranipoor and C. Wang, Eds. new york: Springer, 2012.

[5] P. C. Kocher, "Timing attacks on implementations of," *Diffie-Hellman, RSA, DSS, and other systems, Proc. of Crypto '96, LNCS, vol. 1109, IACR, Springer-Verlag*, pp. 104–113, 1996.

[6] V. B. B and E. Oswald, "Constructive Side-Channel Analysis and Secure Design," vol. 7864, pp. 29–40, 2013. [Online]. Available: http://link.springer.com/10.1007/978-3-642-40026-1

[7] P. Kocher, P. Kocher, B. Jun, B. Jun, C. Research, C. Research, S. Francisco, and S. Francisco, "Differential Power Analysis," *Analysis*, pp. 1–10, 2012.

[8] K. Mpalane, N. Gasela, B. M. Esiefarienrhe, and H. D. Tsague, "Vulnerability of Advanced Encryption Standard algorithm to Differential Power Analysis attacks implemented on ATmega-128 microcontroller," *2016 3rd International Conference on Artificial Intelligence and Pattern Recognition, AIPR 2016*, pp. 70–74, 2016.

[9] C. N. T. INC., "Breaking AES (Manual CPA Attack)." [Online]. Available: https://wiki.newae.com/V4: Tutorial_B6_Breaking_AES_(Manual_CPA_Attack)

[10] Ø. Ekelund, "Low Energy AES Hardware for Microcontroller," Ph.D. dissertation, Norwegian University of Science and Technology, 2009.

[11] B. M. Damian and Z. Hascsi, "Presilicon evaluation on Correlation Power Analysis attacks and countermeasures," pp. 0–4, 2017.

[12] F. Information, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," National Institute of Standards and Technology, Tech. Rep., 2001. [Online]. Available: http://csrc.nist.gov/publications/

[13] T. Popp and S. Mangard, "Masked Dual-Rail Pre-charge Logic : DPA-Resistance Without Routing Constraints," pp. 172–186, 2005.

[14] K. Tiri and I. Verbauwhede, "Securing Encryption Algorithms against DPA at the Logic Level : Next Generation Smart Card Technology 3 Sense Amplifier Based Logic : A CMOS Logic Style with Signal," no. 1, pp. 125–136, 2003.

[15] A. Arora, J. A. Ambrose, J. Peddersen, and S. Parameswaran, "A Double-width Algorithmic Balancing to prevent Power Analysis Side Channel Attacks in AES," *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 76–83, 2013.

[16] J. Daemen and V. Rijmen, "Resistance Against Implementation Attacks A Comparative Study of the AES Proposal," 1999.

[17] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving Smart Card Security using Self-timed Circuits," 2002.

[18] J. J. A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor, "Security Evaluation of Asynchronous Circuits," pp. 137–151, 2003.

[19] J. Rabaey, *Low Power Design Essentials, Series on Integrated Circuits and Systems*, springer science+business media ed., 2009, no. 0.

[20] J. Lagasse, C. Bartoli, and W. Burleson, "Combining Clock and Voltage Noise Countermeasures against Power Side-Channel Analysis," *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, vol. 2160-052X, pp. 214–217, 2019.

[21] J. Lee and D.-g. Han, "Security analysis on dummy based side-channel countermeasures — Case study : AES with dummy and shuffling," *Applied Soft Computing Journal*, vol. 93, p. 106352, 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106352

[22] A. Xmega, "ATxmega128A1U / ATxmega64A1U," p. 216, 2014.

[23] C. N. T. INC., "CW303." [Online]. Available: https://rtfm.newae.com/Targets/CW303 XMEGA/

[24] ——, "ChipWhisperer-Lite (CW1173) Two-Part Version." [Online]. Available: http://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/