A Reliable Architecture for Substitution Boxes in Integrated Cryptographic

Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

A Reliable Architecture for Substitution Boxes in Integrated Cryptographic Devices

G. Di Natale, M. Doulcier, M. L. Flottes, B. Rouzeyre
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université Montpellier II / CNRS UMR 5506
161 rue Ada, 34392 Montpellier Cedex 5, France
{dinatale,doulcier,flottes,rouzeyre}@lirmm.fr

Abstract

In this paper we propose an on-line self-test architecture for hardware implementations of Advanced Encryption Standard (AES). The solution assumes a parallel architecture and exploits the inherent spatial replications of this implementation. Because Substitution boxes (S-Box) represent the largest hardware in this architecture, we focus on faults affecting these S-Boxes and propose a trade-off between hardware overhead and fault latency. We show that our solution is very effective for on-line fault detection while keeping the area overhead very low. Moreover, it does not weak the device with respect to side-channel attacks based on power analysis.

1. Introduction

Cryptography enables to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. Cryptographic functions are implemented on present secure systems such as smart cards (e.g. ATM cards, credit cards, SIM cards, signature cards, pay TV cards,…) and we have to expect that new technologies (including biotechnologies, sensors, actuators) will allow the design of systems cooperating with human beings, requiring therefore even more security and dependability than current applications. We all expect these devices respect very strict constraints in terms of security and are carefully tested after production and during mission time.

Several motivations lead to increase the reliability of these circuits and to carefully select the test procedure. On the one hand the circuit implementation of cryptographic algorithms can be quite area consuming, increasing the probability of device failure, while errors in the cryptographic operation can potentially cause loss of service or compromise the security, on the other hand the cost for hardware reliability is generally high.

Fault detection and tolerance schemes for various implementations of cryptographic algorithm have been recently considered. Mainly, two approaches have been developed: based on codes ([1] [2] [3]) and based on functional redundancy ([3] [4] [5]).

All the techniques based on codes add some bits to the original data word. The main issue in these approaches is the prediction of the value of the code, given the input value and the executed operation. For example, the prediction of a parity bit is almost straightforward for the ShiftRows, MixColumns and AddRoundKey operations performed by the Advanced Encryption Standard [7] because these transformations are either linear or they just perform some permutation of the position of the bits in the state array (see [2] for more details). On the contrary, the prediction of the parity bit is not trivial for the so-called Substitution Boxes (S-Boxes) of the same standard and a dedicated circuit must be added in order to calculate it. Parity codes based solutions ([1] [2]) lead to an overhead of about 20% and the coverage of single faults is very high. But in case of multiple faults or in case of single faults that lead to an even number of errors, these solutions are not effective. Other solutions based on the use of more complex codes such as CRC [1] or systematic nonlinear robust codes [3] allow reaching higher coverage in case of multiple faults but in this case the area overhead significantly increases (> 60%).

Techniques based on functional redundancy ([3] [4] [5]) can be used whenever encryption and decryption modules are implemented on the same circuit. Each encoding phase is followed by a decoding/compare phase in order to check if the resulting decoded text matches with the initial plaintext. An equivalent procedure is used when the circuit is used for decoding a ciphertext: an encoding/compare phase allows to check if the resulting encoded text matches with the initial ciphertext.

None of the previous works considered the characteristics of the proposed approaches when the circuit is attacked by mean of power analysis [6]. Cryptanalysis, the art of breaking ciphers, i.e. retrieving the plaintext without knowing the proper key, is classically a complex process of looking for weaknesses in the implementation of an algorithm or in the algorithm itself, involving statistical analysis, analytical reasoning, math tools…. But when the cryptographic function is implemented in hardware, specific attacks can be developed which target the cryptographic device itself. These types of attacks are called "Implementation Attacks". They can range from the physical opening of the cryptographic device to the monitoring of environmental conditions. Side-Channel Attacks exploit the fact that the cryptographic device leaks physical information during the processing of a cryptographic algorithm. This physical leakage (e.g., power dissipation, timing information...) can be captured and used to compromise the secret key of a cryptographic algorithm by using standard statistical tools. An efficient cryptographic device must therefore prevent the possibility of gathering the secret code by mean of a side-channel attack.

In this paper we propose a low cost concurrent on-line self-test technique for detecting single and multiple faults in the hardware implementation of the Advanced Encryption Standard (AES) during its mission mode. Since the S-Boxes represent the biggest part of the AES circuit, counting up to 75% of the whole circuit and since the prediction of a parity bit is straightforward for the ShiftRows, MixColumns and AddRoundKey operations we only focus on the on-line selft-testability of the S-boxes. The solution is based on the spatial duplication inherent to the parallel implementation of the AES and exploits the native property to have 16 identical repetitions of the same S-boxes.

We present a trade-off between hardware overhead and fault detection latency where one additional S-Box is added every 4 S-Boxes in the circuit. We will prove that our solution is very effective while keeping the area overhead very low. Moreover, although not specifically designed to protect against tampering, we will explain why this architecture makes more difficult the side-channel attack based on Differential Power Analysis (DPA).

The paper is organized as follows. Section 2 presents the proposed on-Line self-test approach, while section 3 discusses the results in terms of area overhead and fault detection capability. Section 4 describes the analysis of the resistance against the Differential Power Analysis. Eventually, Section 5 concludes the paper.

2. Architecture Description

The Advanced Encryption Standard (AES) [7] is a block cipher adopted as an encryption standard by the U.S. government. AES began immediately to replace the Data Encryption Standard (DES), which has been in use since 1976. AES outperforms DES in improved long-term security because of, among other things, larger key sizes (128, 192, or 256 bits).

The AES algorithm is an iterative algorithm composed of 10 rounds. As illustrated in Figure 1, each round consists mainly (except for the last one, where MixColumns is not performed) of 4 transformations: SubBytes (a non-linear byte substitution also called S-Box that operates independently on each byte of the word), ShiftRows (a permutation of the whole 128 bits), MixColumns (XOR among some bits, it operates on 32 bits), and AddRoundKey (XOR between the obtained value and a "round-key", generated at each round as function of the input key). Details of the algorithm are given in [4].

Several hardware implementations for AES circuit have been proposed [8]. No matter the type of implementation, the most expensive part of the circuit in terms of area is the S-Box.

The technique we propose in this paper is designed for all the AES cores (encryption and decryption) that use 16 S-Box repetitions. We do not consider low-area implementations, where there is only one S-Box at the cost of several clock cycles for completing one encryption/decryption round. The proposed solution is anyway applicable to AES implementations where the computation is performed in a semi-parallel way: for instance, AES with 4 or 8 S-Boxes can be modified in order to increase their dependability with the technique we propose in this paper. Typical hardware architecture of the AES where 16 S-Boxes are used at the same time is sketched in Figure 2.

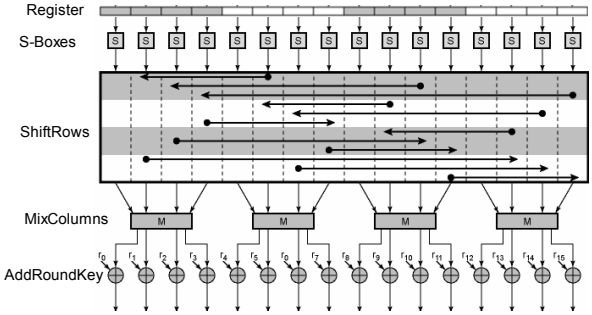

Figure 1: Typical AES Implementation

The most critical part of the circuit in terms of silicon area is composed by the registers and the S-

Boxes, counting up to 75% of the whole circuit. In this paper we focus on the registers and the S-Boxes.
The main idea of the approach is to use one additional SubBytes block (8-bits register and S-Box) every 4 blocks, and to on-line test a pair of SubBytes blocks each clock cycle (see figure 2). In particular, at each clock cycle two blocks are fed by the same inputs and the related outputs are compared in order to detect possible faults.

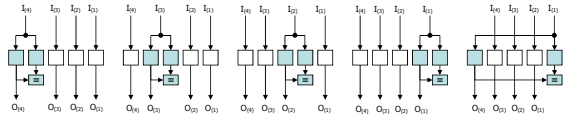

Figure 2: Different test configurations

Figure 3 details the behavior of a part of the circuit where one SubBytes block has been added to 4 blocks. In this figure, `LMux(2)`, `LMux(3)` and `LMux(4)` are multiplexers with an additional output that is asserted whenever the two inputs are equal (i.e., a multiplexer with a comparator). Table 1 details the signals controlled and observed by the control unit. For instance, when `SB4` and `SB3` work together, the `UMux(3)` let the input `I(4)` go into the `SB3`. Among the four signals coming from the comparators, only one at a time is considered by the control unit. For example, in the above case, the `check(4)` signal is verified, i.e., the two related SubBytes blocks are checked.

Table 1: Signals controlled and observed by the Control Unit

Compared	Um	Lm	To check
SB4, SB3	1000	11	check(4)
SB3, SB2	1100	01	check(3)
SB2, SB1	1110	00	check(2)
SB1, SB0	1111	00	check(1)
SB0, SB4	0000	10	check(1)

The scheduling of the comparisons of a pair of SubBytes blocks is a very important issue of the proposed method. One AES encryption lasts 10 clock cycles and there are 5 different configurations, so it's possible to use the 5 configurations twice during one encryption. Through the 5 configurations, each SubBytes block is compared twice (once with the left block, once with the right block). Thus, if the 5 configurations are activates twice, each SubBytes block is compared 4 times during an encryption. A counter can guarantee the test configuration scheduling.

In order to make more difficult the attacks based on fault injection or power analysis, the scheduling of the pairs of SubBytes blocks can be randomly performed. Instead of using a counter for the selection of which pairs must be on-line tested, it's possible to use an LFSR or a True Random Number Generator. In this case, the control unit has to guarantee that anyway all the combinations of pairs are tested before the end of the 10 rounds.

This technique is applicable for different levels of redundancy starting from one spare SubBytes block every 16 blocks up to one spare for each nominal Sbox. We decided to add one spare block every 4 blocks because this solution has a low overhead (as shown in Section 5) while keeping the fault detection latency at an acceptable level.

3. Results and Fault Detection Analysis

In this section we provide some results related to the area overhead and the fault coverage of the proposed approach. The proposed architecture has been described in VHDL and synthesized using Synopsys Design Compiler [9] using a 130nm CMOS library provided by STM [10]. All the S-Boxes have been synthesized as combinational logic. However, the proposed solution can be implemented using a ROM for the S-Box. Moreover, we implemented two different versions of the circuit: in the first version we considered that all the keys used in the AddRoundKey step (see Section 2) were pre-computed and stored in the circuit; in the second version we also implemented the module able to generate all the round keys. In this case, since the Key Generator uses 4 S-Boxes, we implemented the same architecture of Figure 3 for this module. Table 2 summarizes the area of the circuit described in Figure 3.

Table 2: Area

	Base [μm^2]	Redundant [μm^2]
Registers, S-Boxes	33911	+12626
MixColumns, ShiftRows, AddKey	10725	+0
Control Unit	279	+106
Total (w/o Key Generator)	**44915**	**57647**
Key Generator	13789	+3489
Total	**58704**	**74925**

The area overhead of the first version is 28,34% while the overhead of the second version is 27,63%.

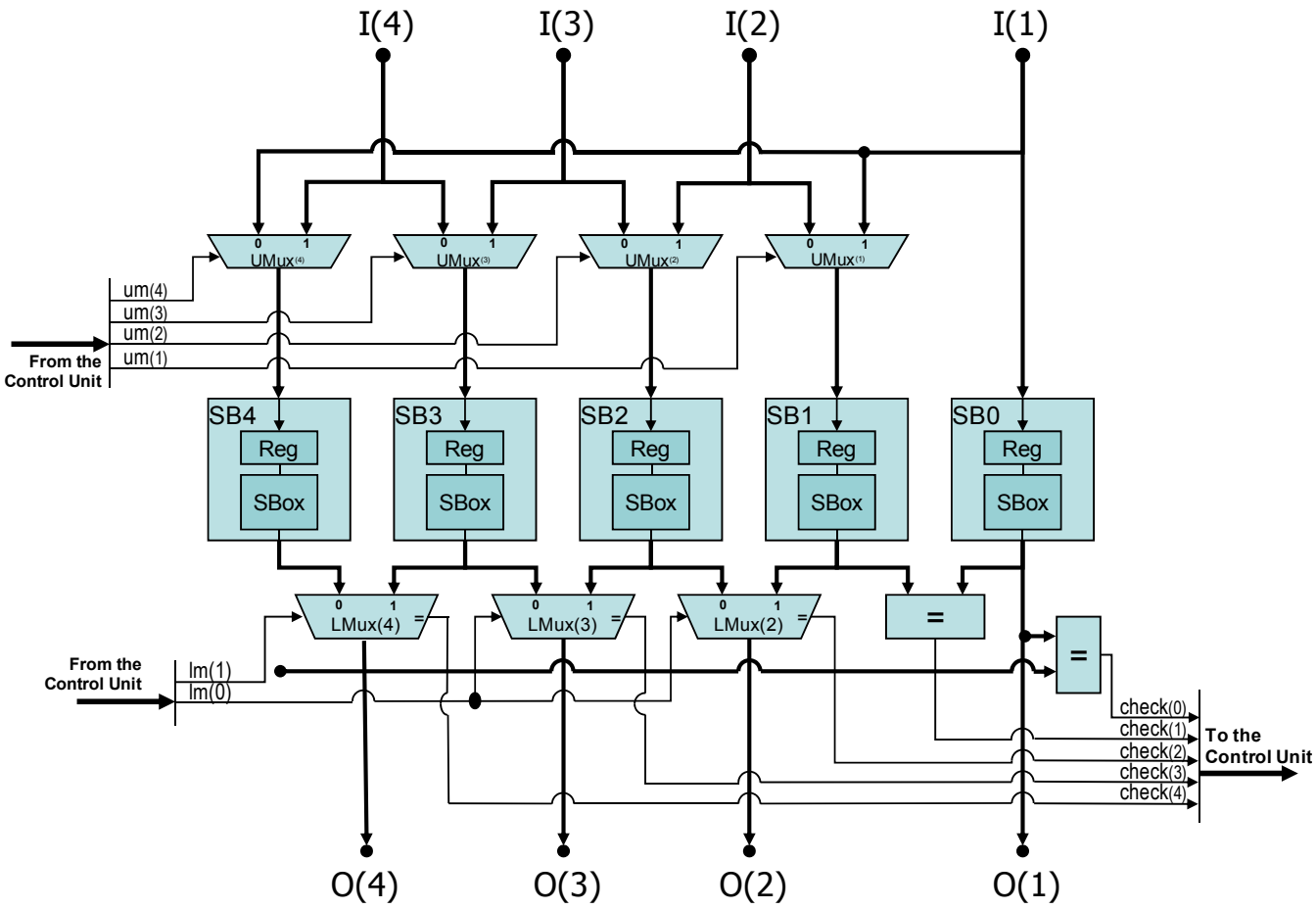

Figure 3: On-Line BIST Architecture

This technique is able to detect any fault (single or multiple) that leads to a different output value of the S-Box while it is tested. We question here the error probability of the proposed architecture. Even if in the proposed architecture the scheduling algorithm ensures that each SubBytes block is compared 4 times over 10, in the following sections all the equations are parameterized according to x the number of clock-cycles of comparison (here x=4) and y the number of clock-cycles without comparison (here y=6).

The error probability can be analyzed by computing the probability $P_e(f)$ of not detecting an error on the circuit's outputs due to a fault *f*. Here, only non-redundant faults are of interest, i.e. we focus on testable faults only. Furthermore, we focus on permanent stuck-at faults, only.

With regard to the proposed architecture, $P_e(f)$ is the probability that the fault f is not sensitized during the clock cycles when the SBox **is** compared (because in this case the error would be detected by the architecture) and sensitize during the clock-cycles when the SBox **is not** compared.

Let denote p_f the probability of sensitizing *f*. As demonstrated in [11], the inherent properties of the AES makes that the sequence of input values that are applied to consecutive rounds can be considered as random. In case of N encryptions, each round is fed by a random sequence of 10·N words of 128 bits (8 bits for each SBox). Therefore, the probability p_f is equal to the ratio of input vectors that sensitize *f* over the number of possible input vectors. Here, the number of possible inputs is 256 since Sboxes have 8 input bits. For each fault *f*, p_f can be obtained through fault simulation.

For the proposed architecture, the probability that *f* is not sensitized during the clock-cycles of comparison is equal to $(1-p_f)^x$ while the probability that *f* is sensitized during the clock-cycles without comparison is equal to $1-(1-p_f)^y$.

Finally, it comes:

$$P_e(f) = (1-p_f)^x \times (1-(1-p_f)^y) \quad (1)$$

Figure 4 represents $P_e(f)$ in function of p_f, when x=4 and y=6. It must be noticed that the hard-to-test faults $(p_f \sim 0)$ and the easy-to-test faults $(p_f \sim 1)$ are not those that most likely produce undetected errors. On the contrary, the maximum value (32.57%) corresponds to faults with p_f equal to 0.14.

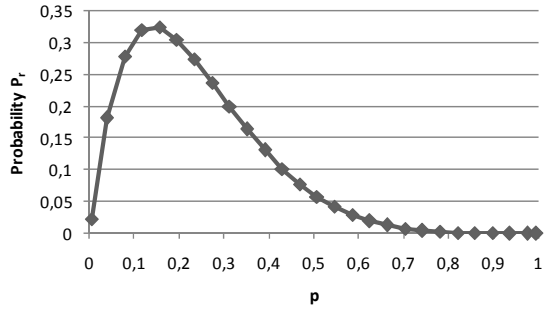

Figure 4: Pe(f)

In order to calculate the error probability of the whole circuit, we fault simulated the circuit to determine the distribution of probabilities of sensitizing the faults of the circuit. Basically, we calculated how many faults are sensitized by one test pattern (p_f = 1/256), how many faults are sensitized by 2 patterns (p_f = 2/256), and so on. Figure 5 summarizes, for each sensitizing probability p, the number FD(p) of faults with that sensitizing probability.

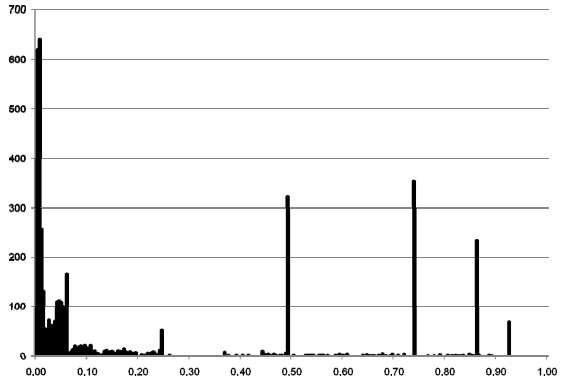

Figure 5: FD(p)

Assuming that each fault has the same probability to appear in the circuit, the overall error probability P_E of the Sbox is calculated as the weighted average of the values $P_e(f)$ according to the distribution FD(p):

$$P_E = \frac{1}{\#Faults} \sum_{i=1}^{256} FD\left(\frac{i}{256}\right) \times P_e\left(\frac{i}{256}\right) \qquad (2)$$

It comes that P_E is equal to 10.20%, i.e. in case of a generic fault the architecture has a probability of 90% to detect it in a single encryption (10 clock cycles).

Let's now analyze the evolution of this probability based on the number of encryptions. When we perform E encryptions, an Sbox is compared for E·X clock cycles, while it is not compared for E·Y clock cycles. The error probability can therefore be rewritten as follows:

$$P_e(f) = (1 - p_f)^{x \cdot E} \times \left(1 - (1 - p_f)^{y \cdot E}\right)$$

Considering x=4, y=6 and the fault distribution FD given in Figure 7, we can re-calculate the overall error probability P_E of the Sbox in function of the number of encryptions (see Figure 6). As can be seen, the error probability slightly increases up to 14% for 5 encryptions, while for higher encryption numbers it tends to 0. Namely, for 300 encryptions, the reliability probability is equal to 99.9%

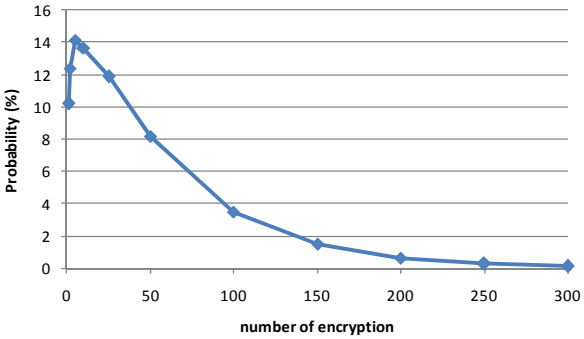

Figure 6: Pr evolution for the "SBox faults"

4. DPA Analysis

Regarding the protection against attacks based on power analysis, it's important to notice that this architecture has several power profiles based on the configuration of the pairs of SubBytes blocks. In particular, for the same input, the circuit can be in 5 different states (see Figure 2) for each of the 4 groups of 5 SubBytes blocks. This leads to a 625 different combinations of pairs, i.e., 625 different power profiles. This characteristic makes the power analysis extremely difficult, particularly when the scheduling of the pairs of SubBytes blocks is randomly selected because in this case the 625 power profiles are not cyclically repeated.

We performed Differential Power Analysis (DPA) on both versions of the circuit (i.e., with and without the concurrent on-line self-test technique) using an in-house DPA simulator [12]. We focused on this attack because among all the known possible attacks, this is one of the cheapest and easiest to perform. We found that in the case of on-line BIST architecture, the DPA attack is slightly more difficult to perform. Figure 7 shows the DPA curves for the standard and the

proposed implementations. The DPA attack succeeds when the curve with largest peak corresponds to the right secret key. After 256 encryptions the curve corresponding to the secret key clearly emerges among the others while, for our architecture, at least 512 encryptions are necessary to have the same confidence level.

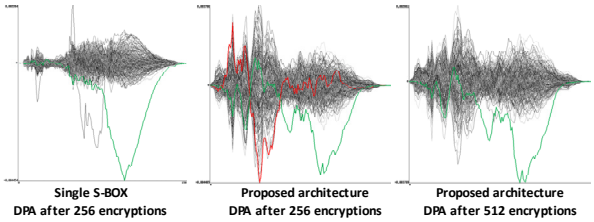

Figure 7: DPA Curves

5. Conclusions

Cryptosystems are inherently computationally complex, and in order to satisfy the high throughput requirements of many applications, they are often implemented by means of VLSI devices.

In this paper we proposed a low cost concurrent on-line self-test technique able to detect faults in the registers and in the S-Boxes of the AES. The solution, based on spatial redundancy, exploits the native AES property to have 16 identical repetitions of the same block (the S-Box). We presented a trade-off between hardware overhead and fault latency where one additional S-Box is added every 4 S-Boxes in the circuit leading to 4 tests of every S-Box per encryption cycle (10 rounds).

The solution is very effective while keeping the area overhead very low (about 28%). Moreover, although not specifically designed to protect against tampering, this architecture makes more difficult side-channel attacks based on power analysis.

The proposed approach can be easily adapted in order to trade off the level of redundancy against the desired reliability. For instance, fixing the maximum number of encryptions required to reach the desired level of reliability, one can determine the number of spare SubBytes blocks that must be added to the base architecture. This solution can be easily enhanced with the use of other detection and/or correction techniques like those based on codes.

10. References

[1] K. Wu, R. Karri, G. Kuznetsov, M. Goessel, “Low Cost Concurrent Error Detection for the Advances Encryption Standard”, Proc. Int’l Test Conference, pp. 1242-1248, 2004

[2] G. Di Natale, M.-L. Flottes, B Rouzeyre, “An On-Line Fault Detection Scheme for SBoxes in Secure Circuits”, Proc. IEEE Int. On-Line Testing Symposium, 2007, pp. 57-62

[3] R. Karri, K. Wu, P. Mishra, Y. Kim, “Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers”, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 12, Dec. 2002, pp. 1509-1517

[4] C. Yen, B. Wu, “Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard”, IEEE Trans Computers, vol. 55, no. 6, June 2006, pp. 720-731

[5] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, V. Piuri “Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard”, IEEE Trans. Computers, vol. 52, no. 4, pp.492-505, Apr. 2003

[6] P. Kocher, J. Jaffe, B. Jun, “Differential Power Analysis”, Proc. CRYPTO’99, 1999, pp 388-397

[7] “Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication 197, November 26, 2001.

[8] X. Zhang, K. K. Parhi, “Implementation Approaches for the Advanced Encryption Standard Algorithm”, IEEE Circuits and Systems Magazine, vol. 2, Issue 4, pp. 24-46, 2002

[9] http://www.synopsys.com

[10] http://www.st.com

[11] P. Hellekalek, S. Wegenkittl, "Empirical evidence concerning AES", ACM Trans. Model. Comput. Simul., Vol. 13, Issue 4 (Oct. 2003), pp 322-333.

[12] G. Di Natale, M.-L. Flottes, B. Rouzeyre, “An Integrated Validation Environment for Differential Power Analysis”, IEEE International Symposium on Electronic Design, Test & Applications (DELTA 2008), Hong Kong, January 2008, pp. 527-532