An On-Line Fault Detection Scheme for SBoxes in Secure Circuits

Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

An On-Line Fault Detection Scheme for SBoxes in Secure Circuits

G. Di Natale, 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,flottes,rouzeyre}@lirmm.fr

Abstract

In this paper we propose an on-line fault detection architecture for bijective Substitution Boxes used in cryptographic circuits. Concurrent fault detection is important not only to protect the encryption/decryption process from random and production faults, it also protects the system against side-channel attacks, in particular those based on fault injection. We will prove that our solution is very effective while keeping the area overhead very low. Besides, we will analyze the correlation between the information processed by the circuit and the power consumption in order to asses the quality of the solution with respect to other side-channel attacks such as Power Analysis techniques.

1. Introduction

From a mathematical point of view the cryptographic device is a function that allows calculating the encoded text based on the value of its clear text to encode and the secret key. All the classical cryptographic attacks try to identify some additional associations between the input and the output of the function to discover some hidden relations that allow gathering the secret key.

The classic cryptanalysis is purely theoretical. On the other hand, when the function is implemented in hardware, other attacks are possible because the attacker has access to the physical cryptographic device and he can play around with it. These types of attacks are called "Implementation Attacks" which target the cryptographic device itself. These attacks can range from the physical opening of the cryptographic device to changing and/or observing the environmental conditions, e.g. attacks based on the observation of the inherent leakage of the cryptographic device.

Among all the attacks proposed in the literature, *Side-Channel Attacks* exploit the fact that the cryptographic device itself leaks physical information during the processing of a cryptographic algorithm. This physical leakage (e.g., power dissipation, timing information, ...) can be captured externally and can then be used to compromise secret keys of cryptographic algorithms by using standard statistical tools. On the other side, *Fault Attacks* (e.g., [1] [2] [3]) aim to cause errors during the processing of a cryptographic device. An additional information flow can be caused if the cryptographic device returns erroneous cryptograms or a modified execution path is entered. By comparing the results obtained with and without the fault injection the secret key can be retrieved.

Designing a secure circuit requires to implement protections against such attacks. It is also important to be sure that the implementation of one countermeasure for a specific attack will not enable an attacker to use a different attack more easily.

In this paper we propose a low cost concurrent error detection technique able to resist to fault attacks. At the same time, we will analyze the properties of the proposed solution when the circuit is under side-channel attacks based on the observation of the power consumption.

The technique we propose in this paper is not designed particularly for circuits implementing a specific cryptographic algorithm but for all those that resort to bijective Substitution Boxes. A substitution box (or Sbox) is a basic component of symmetric key algorithms, used to obscure the relationship between the plaintext and the ciphertext. In general, an Sbox takes some number of input bits, m, and transforms them into some number of output bits, n. Fixed tables are normally used, as in the Data Encryption Standard (DES) or in the Advanced Encryption Standard (AES), but in some ciphers the tables are generated dynamically from the key (e.g. the Blowfish and the Twofish encryption algorithms).

The main idea of the approach is to use two parity bits, one for the input word of the Sbox and one for its output word. SBoxes perform an operation that is not linear and thus is not invariant with respect to the parity of the processed data, i.e., the parity bit is not preserved after the transformation. It is necessary to insert an additional circuit able to predict the value of the output parity bit starting from the input value. However, since we focus on bijective SBoxes (like the one used for the AES), it is possible to predict the value of the input parity bit starting from the output value.

We will compare our solution with other concurrent error detection techniques for an AES implementation, since AES is nowadays the state-of-the-art algorithm for symmetric cryptography.

We will prove that, compared to previous works, our solution has higher fault coverage and lower area overhead. Moreover, we will prove that this solution does not increase the chances for an hacker to use power analysis attacks.

The paper is organized as follows. Section 2 introduces the basic concepts and the characteristics of the Advanced Encryption Standard algorithm, used as test-bench for the experimental results. Section 3 summarizes the state-of-the-art on this topic, while Section 4 presents the parity-based concurrent error detection approach. Section 5 discusses the results in terms of area overhead, fault detection capability (to both stuck-at and bit-flip on the registers of the circuit), and power analysis. Eventually, Section 6 concludes the paper.

2. Advanced Encryption Standard

The Advanced Encryption Standard (AES) [4] is a block cipher adopted as an encryption standard by the U.S. government. Several hardware implementations for AES circuit have been proposed [5]. No matter the type of implementation, the most expensive part of the circuit in terms of area is the so called SBox.

The AES is an iterative algorithm. Each iteration is called a round. The total number of rounds is 10. At the start of encryption, input is copied to the State array. After the initial key addition, 10 rounds of encryption are performed. The first 9 rounds are the same, with small difference in the final round. As illustrated in Figure 1, each of the first 9 rounds consists of 4 transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey. The final round excludes the MixColumns transformation.

The encryption structure in Figure 1 can be inverted to get a straightforward structure for decryption.

SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (SBox). This transformation can be pre-calculated for each possible input value since it works on a single byte, therefore there are only 256 values. Implementations of the SBox are discussed in Section 3.

ShiftRows Transformation

In this transformation, the bytes in the first row of the State do not change. The second, third, and fourth rows shift cyclically to the left one byte, two bytes, and three bytes, respectively.

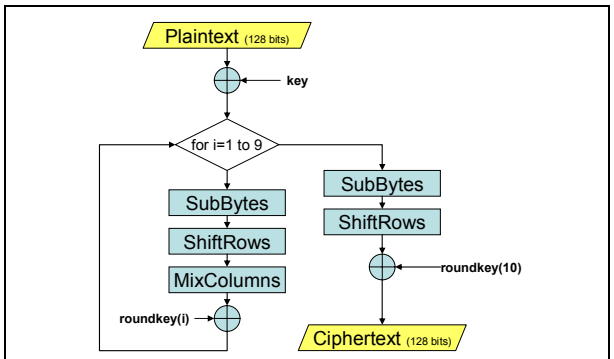

Figure 1: AES Algorithm (encryption)

MixColumns Transformation

The MixColumns transformation is performed on the State array column-by-column. Each column is considered as a four-term polynomial over GF(2^8) and multiplied by a(x) modulo $x^4 + 1$, where:

$$a(x) = (00000011)_2\, x^3 + (00000001)_2\, x^2 + (00000001)_2\, x + (00000010)_2$$

AddRoundKey Transformation

In AddRoundKey transformation, a roundkey is added to the State array by bitwise XOR operation. Each roundkey consists of 16 words generated from Key Expansion described below.

Key Expansion

The key expansion routine, as part of the overall AES algorithm, takes the input key of `128 bits`. The output is an expanded key of `11*128` bits, i.e., the expanded key is composed of the secret key and 10 roundkeys, one for each round. Details of the algorithm that allows determining the value of each roundkey are given in [4].

3. State-of-the-Art

Fault detection and tolerance schemes for various implementations of cryptographic algorithm have recently been considered. Several motivations led to increase the reliability of these circuits. From one side

the circuit implementation of cryptographic algorithms can be quite area consuming, increasing the probability of device failures. Fault detection is therefore helpful in finding faults during the production tests. From the other side, intentional intrusions and attacks based on the malicious injection of transient faults into the device are very efficient in order to extract the secret key [6] [7] and must be prevented.

Since crypto chips are consumer products of mass production, cheap solutions for concurrent error detection and correction are of great importance [8] [9] [10] [11]. A natural choice for concurrent error detection is the application of parity codes. Concurrent checking for the AES by parity prediction was first introduced in [13] and [14]. One of the main problems targeted in the literature is the prediction of the output parity given the input state and the input parity bit.

The prediction of the output parity bit (when a parity bit is added to each byte) is almost straightforward for the ShiftRows, MixColumns and AddRoundKey steps because these transformations are either linear or they just perform some permutation of the position of the bits in the state array (see [13] for more details). On the contrary, the prediction of the output parity bit is not trivial for the Sbox. In this section we summarize the solutions based on the parity bit for the SBox.

The SBox is usually implemented either as a 256x8 bits memory consisting of a data storage section and an address decoding circuit, or a combinational circuit. The incoming data bytes will normally have properly generated even parity bits. A solution to generate the outgoing parity bits is proposed in [14]: an even parity bit is either stored with each data byte in the SBox (memory implementation, see Figure 2.a), or on-line generated with an ad-hoc combinational circuit (in the case of combinational logic implementation for the SBox). This solution is not very expensive and it guarantees acceptable fault coverage.

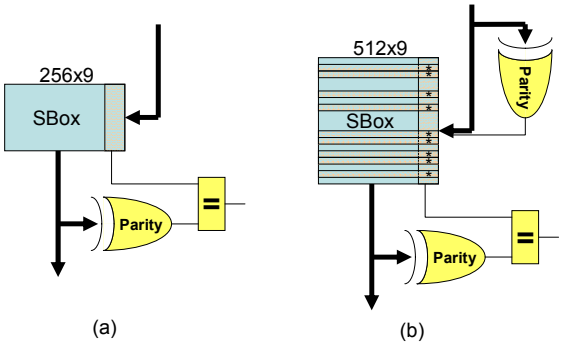

Figure 2: State-of-the-Art solutions based on parity bit. a) solution presented in [14]; b) solution presented in [13]

To increase the dependability and to detect additional input parity errors and some internal memory errors (data or decode), [13] proposes replacing the original 8-bit decoder with a 9-bit one, yielding a 512x9 bits memory (Figure 2.b). If a 9-bit address with an even parity is decoded, the corresponding output byte with its associated even parity bit is produced. Otherwise, a constant word of 9 bits with a deliberately odd parity is output, e.g., "00000000 1". Thus, half of the entries in the SBox memory will be deliberately wrong (in the Figure, all the rows marked with a '*'). In case of a single error in the input value, a wrong cell will be addressed. That cell will contain an erroneous parity bit that will be detected during the parity bit check. This solution guarantees higher fault coverage but it is very expensive in terms of area overhead.

4. Architecture Description

The technique we propose in this paper is designed for all the circuits implementing symmetric cryptographic algorithms that resort to bijective Substitution Boxes.

The main idea of the approach is to use two parity bits, one for the input word of the Sbox and one for its output word. SBoxes perform an operation that is not linear and thus is not invariant with respect to the parity of the processed data, i.e., the parity bit is not preserved after the transformation. It is necessary to insert an additional circuit able to predict the value of the output parity bit starting from the input value. Moreover, since we focus on bijective SBoxes, it is possible to predict the value of the input parity bit starting from its output value.

We do not add any parity bit in the memory that stores the SBox values (or into the combinational logic that implements it). On the contrary, we calculate the parity of the input value and we compare it with the parity bit predicted starting from the output value of the Sbox. In addition, we calculate the parity bit of the output of the SBox and we compare it with the prediction of this bit starting from the input value (see Figure 3).

If we compare this solution with those presented in the past, we can find that they were based on the prediction of the output parity, only. In [13], a parity bit of the input value has been used to detect an even number of faults in the input word. It was anyway not predicted starting from the output. Therefore an odd number of faults in the input word leads to an undetectable fault. We will prove that our solution is able to identify a certain number of faults in this category.

We calculated the Output Parity Predictor and the Input Parity Predictor using the truth tables of the SBox and of the parity bits, calculated for both the input value and the output value.

This scheme allows double parity bit prediction of the SBox circuit and it should allow covering more faults than the architectures proposed in the literature. Section 5 will prove that actually this scheme is more effective.

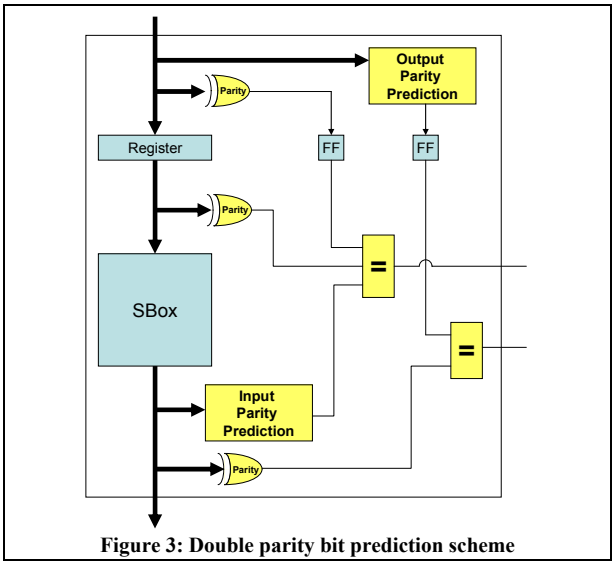

Figure 3: Double parity bit prediction scheme

5. Experimental Results

In this section we provide some results related to the area overhead, the stuck-at fault coverage, the bit-flip coverage, and power consumption of the device implemented with the proposed approach. We also compare these results with the architectures proposed in [13] and [14].

Area Overhead

The architecture proposed in Figure 3 has been described in VHDL and synthesized using Cadence RTL Compiler [16]. Both the SBox and the prediction circuits have been synthesized as combinational logic. However, the proposed solution can be implemented using a ROM for the SBox.

Table 1 summarizes the area of the circuit described in Figure 3. The area overhead is 38,33%.

Table 1: Area

Instance	Cells	Cell Area	Net Area
SboxInOutPredictionV3	765	51979	16758
inst_sbox	553	34707	10134
inst_Outprediction	93	5879	1377
inst_Inprediction	91	5606	1377
inst_OutParity	4	710	27
inst_InParity	4	710	27
inst_InParityFF	4	710	27

Comparative results with state-of-the-art are shown in Table 2.

Stuck-at Fault Coverage

In order to measure the detection capability of the proposed architecture, we used the fault simulator provided by Synopsys [16] (TetraMax). The circuit has been modified in such a way that the only output signals visible by the fault simulator are the comparator signals. In this way, the obtained fault coverage gives a measure of the detection capability when a single error affects the circuit. The obtained fault coverage is equal to 99.55%.

Table 2 summarizes some comparison between our solution and the architectures proposed in [14] and [13], sketched respectively in Figure 2.a and Figure 2.b. Architectures proposed in [13] and [14] have been synthesized using the same technological library in order to get comparable results. In both cases the SBox has been implemented as combinational logic.

The solution proposed in [14] allows covering 91.95% of the faults only, guaranteeing anyway a lower area overhead. The solution proposed in [13] guarantees higher fault coverage than the solution proposed in [14], but at the expense of a very high overhead (47.28%). Furthermore, the area overhead is even higher when the Sbox is implemented by mean of a ROM (about 125%).

Table 2: Comparison

Architecture	Area Overhead	Fault Coverage
Our approach	38.33%	99.55%
[14] (Figure 2.a)	18.17%	91.95%
[13] (Figure 2.b)	47.28%	93.43%

Bit-Flip Fault Coverage

Besides the stuck-at coverage, we used Verilog simulation in order to calculate the Bit-flip detection capability of the approach.

Bit flip in the flip-flops of the circuit are very important because all the techniques of fault attacks target this behavior. A fault may be induced in many ways, such as power glitch, clock pulse or radiation of many kinds (laser, etc...).

The simulation has been performed considering the occurrence of all the combinations of N concurrent bit flips, with $1 \le N \le 10$ (8 flip flops for the input register, plus two flip flops for the two parity bits). Moreover, for each combination, all the possible input values have been simulated. Thus, we performed a number of injections equal to:

$$\#Injections = \sum_{N=1}^{10} \binom{10}{N} \bullet 256 = 261888$$

The simulation of the bit flip is detailed in Figure 4. After setting the input pattern, the bit flip is forced right after the rising edge of the clock. The output value is sampled at the next clock cycle, after that the bit flip is released and a new input pattern is applied.

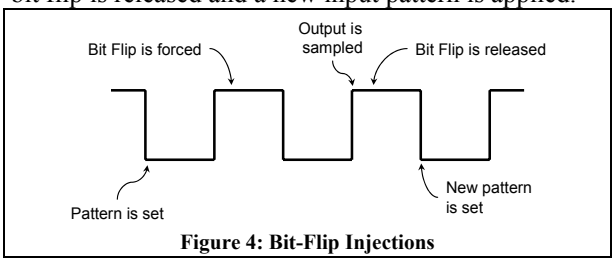

Figure 4: Bit-Flip Injections

Results of the fault coverage are shown in Table 3. Each row in the table corresponds to a set of experiments performed with the same amount of bit flips. The first column indicates the number of bit flips (#BF) in that experiment. In the second column there is the overall number of possible faults (all the combinations of #BF faults over 10 flip flops). The third column contains the number of detected faults while the last column expresses this result in percentage.

In opposition to detection schemes based on a single parity bit, this solution allows covering also a percentage of even faults.

Table 3: Bit-Flip Coverage

# Bit flip	# Faults	# Detected Faults	**Coverage [%]**
1	2560	2560	**100 %**
2	11520	3387	**29.4 %**
3	30720	30720	**100 %**
4	53760	13548	**25.2 %**
5	64512	64512	**100 %**
6	53760	14892	**27.7 %**
7	30720	30720	**100 %**
8	11520	2707	**23.5 %**
9	2560	2560	**100 %**
10	256	135	**52.9 %**

Power analysis

Although the results of the previous section showed that the circuit is well protected against fault attacks, there are other types of side-channel attacks that allow finding the secret key. To reach high security it is necessary to guarantee good protection from all the types of attacks. In fact, it would be useless to spend resources to be protected from one type of attack if this addition facilitates other types of attacks.

Since their introduction by Kocher in 1998 [17], power analysis attacks have attracted significant attention within the cryptographic community. While early works in the field mainly threatened the security of smart cards and simple processors, several recent publications have shown the vulnerability of hardware implementations as well.

In this section we will prove that the correlation between power consumption and processes data is not increased due to the fault detection technique. This information guarantees that the power analysis attacks does not become easier because of the introduction of the error detection scheme.

We carried out gate level power estimation by the Synopsys Power Compiler, which is included in the framework of the Synopsys Design Compiler. Cell internal power and net toggling (i.e. the frequency of the input transitions) have a direct effect on the dynamic power of a design. Power Compiler needs such information in order to perform power reporting or optimizations; net toggling is also called switching activity. Switching activity can be given to the tool in two ways: (a) by specifying the toggle rate in terms of static probability, in which case the results are potentially inaccurate; and (b) by measuring the switching activity under a certain simulation scenario and back-annotating it to the power estimation tool; measurements can be carried out on some or all design objects. Switching activity annotation is performed by compiling and simulating the design within an HDL simulator that can capture switching activity; in our case, Cadence Verilog XL verilog_toggle command was used.

The power consumption of the circuit has been calculated for each transition of the input signal, i.e. considering all the possible couples of different inputs. In particular, we simulated 256*255=65280 transitions.

The results of the power consumption have been classified as in Table 4. The first column contains the input transition, while the second column contains the related output transition (the output of the Sbox). For both columns, in parenthesis it is reported the Hamming distance between the two words. The last column contains the power (expressed in mW) that has been consumed for that transition.

Table 4: Power analysis

Input Transition	**Output Transition**	**Power [mW]**
00000000 → 00000001 (#1)	01100011 → 01111100 (#5)	0.8043
00000000 → 00000010 (#1)	01100011 → 01110111 (#2)	0.9522
…	…	…

We estimated the power consumption for both the circuit with the double parity prediction and for the circuit without any error detection mechanism. We calculated then the correlation between the Hamming distance of the output transition and the estimated power, in two cases:

1. for all the items in the array. In this case we obtained a correlation of 0.025 for the circuit without error detection and 0.065 for the circuit with double prediction;
2. for the items in the array grouped by Hamming distance, and with the related average of estimated power consumption. Figure 5 shows these results. In this case the correlation is equal to 0.361 for the circuit without error detection and 0.338 for the circuit with double prediction.

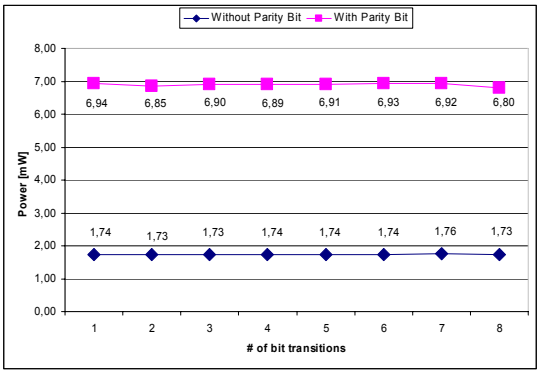

Figure 5: Correlation between Hamming distance and power consumption, with and without parity bit

Thus, while our technique successfully prevents from fault injection attacks, it also does not make easier side-channel attacks based on the observation of the power consumption, such as DPA.

6. Conclusions

Crypto-systems 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 error detection technique based on double parity prediction, able to resist to fault attacks. At the same time, we analyzed the properties of the proposed solution when the circuit is under side-channel attacks based on the observation of the power consumption.

The introduction of the parity bit prediction, both in input and output, increased significantly the fault coverage of the circuit, without resorting to expensive solutions requiring large extra memory area and without enabling an attacker to exploit power attacks more easily.

7. References

[1] E. Biham, A. Shamir, "Differential fault analysis of secret key cryptosystems," In Advances in Cryptology – CRYPTO'97, LNCS 1294, pp. 513–525, Springer-Verlag, 1997

[2] P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," Cryptology Archive of IACR, No. 010, 2003

[3] C. Giraud, "DFA on AES," Cryptology Archive of IACR, No. 008, 2003

[4] "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, November 26, 2001.

[5] 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

[6] D. Boneh, R. DeMillo, R. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations", Journal of Cryptology, vol. 14, pp. 101-119, 2001

[7] M. Akkar, C. Giraud, "An Implementation of DES and AES, Secure against some Attacks", Proc. Of CHES'01, LNCS, vol. 2162, pp. 315-325, 2001

[8] G. Bertoni, et al., "Detecting and Locating Faults in VLSI Implementations of the Advanced Encryption Standard", Proc. 18th IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems, pp. 105-113, Nov. 2003

[9] K. Wu, et al., "Low Cost Concurrent Error Detection for the Advances Encryption Standard", Proc. Int'l Test Conference, pp. 1242-1248, 2004

[10] 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

[11] 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

[12] G. Bertoni, et al., "A parity Code Based Fault Detection for an Implementation of the Advanced Encryption Standard", Proc. IEEE Int. Symposium on Defect and Fault Tolerance in VLSI, pp. 51-59, Nov. 2002

[13] 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

[14] V. Ocheretnij, G. Kouznetsov, R. Karri, M. Gossel, "On-Line Error Detection and BIST for the AES Encryption Algorithm with Different S-Box Implementations", Proc. IEEE Int. On-Line Testing Symposium, 2005, pp. 141-146

[15] http://www.cadence.com

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

[17] P. Kocher, J. Jaffe and B. Jun, "Introduction to differential power analysis and related attacks," 1998, available at http://www.cryptography.com/dpa/technical