I. J. Computer Network and Information Security, 2013, 6, 49-55
Published Online May 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.06.07

A Robust Fault Detection Scheme for the Advanced Encryption Standard

Hassen Mestiri, Noura Benhadjyoussef, Mohsen Machhout and Rached Tourki
Electronics and Micro-Electronics Laboratory (E. µ. E. L)
Faculty of Sciences of Monastir, Tunisia
hassen.mestiri@yahoo.fr

Abstract **—** Fault attacks are powerful and efficient cryptanalysis techniques to find the secret key of the Advanced Encryption Standard (AES) algorithm. These attacks are based on injecting faults into the structure of the AES to obtain the confidential information. To protect the AES implementation against these attacks, a number of countermeasures have been proposed.

In this paper, we propose a fault detection scheme for the Advanced Encryption Standard. We present its details implementation in each transformation of the AES. The simulation results show that the fault coverage achieves 99.999% for the proposed scheme. Moreover, the proposed fault detection scheme has been implemented on Xilinx Virtex-5 FPGA. Its area overhead and frequency degradation have been compared and it is shown that the proposed scheme achieves a good performance in terms of area and frequency.

Index Terms **—** Security, Fault Attacks, Fault Detection Scheme, Countermeasure, Advanced Encryption Standard (AES)

I. Introduction

In October 2000, The Advanced Encryption Standard was finalized by the National Institute of Standards and Technology (NIST), when the Rijndael algorithm was adopted [1]. The AES algorithm replaced the data encryption standard (DES), which had been in use since 1976. Until now, many architectures, for efficient VLSI realization of AES algorithm, have been proposed and their performance have been evaluated by using ASIC libraries and FPGA [2-5]. Cryptographic algorithm AES is currently used in a very large variety of scenarios. The most common examples: e-commerce and financial transactions, which have strong security requirements [6]. Therefore, it is necessity to protect the AES implementation against the fault attacks, such as the Differential Fault Analysis (DFA) [7-12] and the Simple Fault Analysis (SFA) [13].

The fault attacks are based on injecting faults into the structure of the AES to obtain the secret cryptographic keys. The cryptanalyst injects faults during the execution of the processing algorithm. This disturbs the normal execution behavior and results in creating faulty ciphertext. Therefore, the cryptanalyst can guess secret key after certain number of the fault injections and analyzing faulty ciphertexts.

To make a robust implementation against fault attacks, several countermeasures have been proposed [14-21].

In [14], Karri et al proposed an error detection method, called concurrent error detection (CED). This fault detection model exploits the inverse relationship between encryption and decryption to detect the fault at the operation level, the round level and the algorithm level.

Yen et al proposed in [15] a fault detection based on the cyclic redundancy check (CRC). This approach uses an (n+1, n) CRC to detect the faults, where n (4, 8, 16) and the parity of the output of each transformation is predicted. The CRC fault detection can also be implemented in the operation level, round level and algorithm level.

In [16], Natal et al proposed an error detection method based on hardware redundancy. They used four S-Box for the implementation of the SubBytes transformation. They used one additional S-Box to check the result of every four S-Box.

Rajendran et al proposed in [17] a new CED mechanism based on the slide attack. This mechanism is independent of the implementation scheme of the S-Box. It can be applicable to all the symmetric block ciphers. It is applicable to both the encryption and decryption mechanisms.

In [18], Chu et al proposed new error detection method using the polynomial residue number systems (PRNS) to protect the AES implementation.

In this paper, in order to improve the security of the AES, we propose a fault detection scheme against the Differential Fault Analysis. This method combines several countermeasures presented in the literature.

The organization of this paper is as follows. The basic structure of AES is given in section II. In section III, several architectural solutions for fault detection in cryptographic systems are presented and categorized according to the type of the redundancy scheme. In section IV, we present the proposed a fault detection scheme for the AES. In section V, The fault coverage of the proposed fault detection scheme is obtained. In section VI, the implementation results and the

performances report of the proposed fault detection scheme are discussed and compared in terms of area, frequency and fault coverage. Section VII concludes the paper.

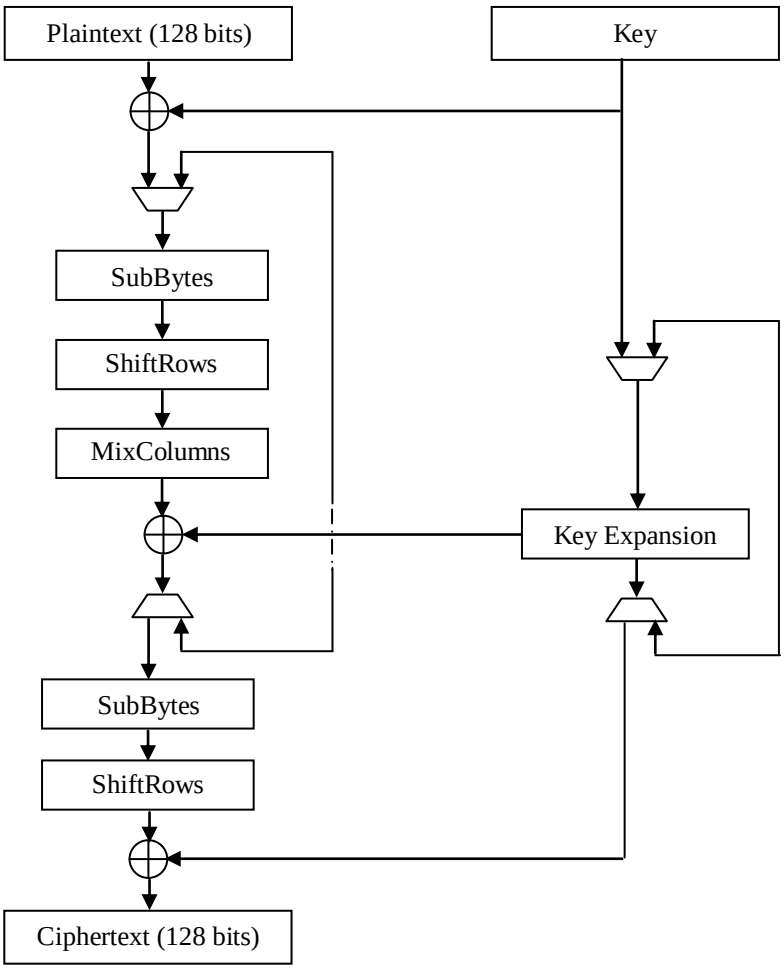

Figure 1. AES algorithm: Encryption structure

II. AES Algorithm

AES is a symmetric-key block cipher with a data block length of 128 bits, which supports different key lengths of 128, 192 or 256 bits. The AES is a round-based encryption algorithm. The number of rounds, Nr, is 10, 12, or 14, when the key length is 128, 192 or 256 bits, respectively.

In the encryption of the AES algorithm, each round, except the final round, performs four transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey, while the final round does not have the MixColumns transformation. The key used in each round, called the round key, which is generated from the initial key by a separate key scheduling module.

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the state using a substitution table (S-Box). This S-Box, which is invertible, is constructed by composing two transformations:

- Take the multiplicative inverse in the finite field $GF(2^8)$, the element {00} is mapped to itself.
- Apply the following affine transformation (over GF(2)).

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \quad (1)$$

for $0 \le i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value {63h}.

The ShiftRows transformation is a circular shifting operation on the rows of the state with different numbers of bytes. As shown in (2), the first row of the state is kept as it is, while the second, third and fourth rows cyclically shifted by one byte, two bytes and three bytes to the left, respectively.

$$\begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix} \xrightarrow{ShiftRows} \begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{bmatrix} \quad (2)$$

The MixColumns transformation operates on the state column by column, treating each column as a four term polynomial. The columns are considered as polynomials

over GF(2^8) and multiplied $x^4 + 1$ with a fixed polynomial a(x) given by:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (3)$$

In matrix form, the MixColumns transformation can be expressed as:

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} \quad (4)$$

$$0 \le j \le 3,$$

The AddRoundKey is a XOR operation that adds a round key (K) to the state in each iteration, where the round keys are generated during the key expansion phase.

$$C = S' + K \quad (5)$$

where $C=\{c_{i,j}\ 0 \le i,j \le 3\}$ is the round output.

The AES algorithm takes the cipher key and performs a Key Expansion routine to generate a key schedule. The key expansion generates a total Nb(Nr + 1) words, where Nb = 4.

Block diagram of the AES encryption is shown in Fig. 1.

The transformations in the decryption process perform the inverse of the corresponding transformations in the encryption process. In the AES decryption rounds, four transformations are used: InvShiftRows, InvSubBytes, AddRoundKey, and InvMixColumns. The AddRoundKey is the same for both encryption and decryption

In this paper, we consider the implementation of the 128-bit key system only, as this is the most commonly implemented form of AES.

III. Countermeasures Against Fault Attacks

In the literature, a large number of countermeasures proposed against fault attacks are based on some sort of redundancy: temporal redundancy, hardware redundancy and information redundancy.

A. Temporal Redundancy

In temporal redundancy, the same hardware is used to repeat the same process twice using the same input data. This technique uses minimum hardware overhead. Yet, it entails 100% time overhead.

B. Hardware Redundancy

In hardware redundancy, two copies of the hardware are used concurrently to perform the same computation on the same data. After each computation, the results are compared and every difference is reported as a fault. The advantage of this technique is that it can detect both transient and permanent faults. However, it requires at least 100% hardware overhead.

C. Information Redundancy

According to [22], the Information redundancy can be classified into three categories: parity-1, parity-16 and nonlinear robust codes.

The parity-1 uses only single bit parity for the entire 128-bit output and the fault is detected by comparing the predicted parity with the calculated parity at the end of each round.

In Parity-16, one parity bit is generated for each byte of the input and the fault is detected in the same way in Parity-1. While gaining higher fault coverage, the area overhead of Parity-16 is more than Parity-1.

The nonlinear robust codes based on the addition of two cubic networks. It allows to produce r-bit signatures to detect fault. This method offers good fault coverage. Yet, its hardware overhead is comparable to the hardware redundancy.

IV. The Proposed Fault Detection Scheme

In this section, we present the proposed fault detection scheme to protect the AES 128-bit implementations against fault attacks. It is noted that this scheme can also be applied to AES-192 and AES-256.

The proposed fault detection scheme structure for the AES is shown in Fig. 2.

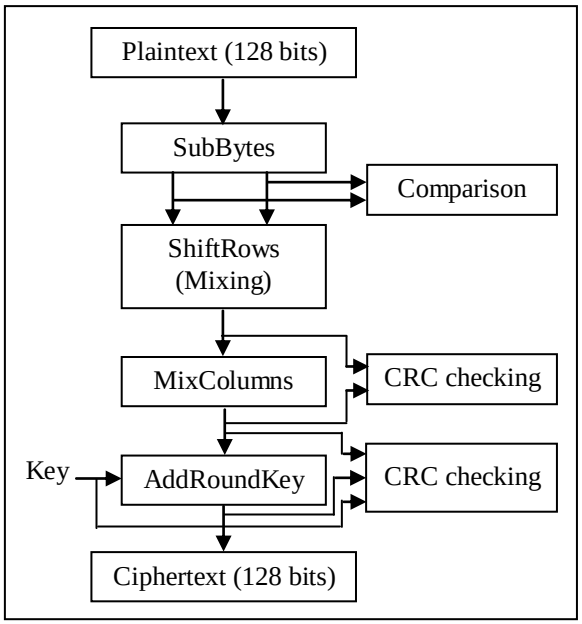

Figure 2. The proposed fault detection scheme structure for the AES

A. In SubBytes

SubBytes, which is the only non-linear transformation in the AES, consists of 16 S-Box. Using the technique of parity to make the prediction parity for the SubBytes is complex due to nonlinearity of this transformation.

To protect the SubBytes, our method uses the hardware redundancy method. We implement two SubBytes transformations in parallel. At the end of SubBytes

computation, the results are compared and every discrepancy is considered as a fault.

The same method can be applied in the decryption process by using two InvSubBytes transformation in parallel.

B. In ShiftRows

The output of the SubBytes transformation acts as the input to ShiftRows. As seen in (2), the output state of ShiftRows is obtained by shifting the matrix state. To secure the ShiftRows transformation, we used the scrambling method proposed in [19]. This method consists in scrambling the output of ShiftRows as presented in Fig. 3.

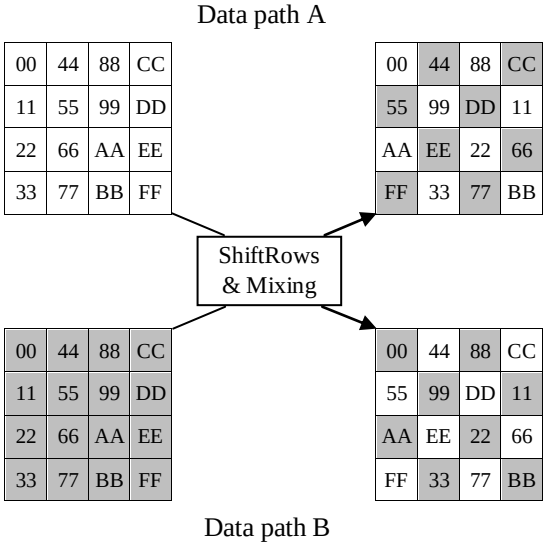

Figure 3. Scrambling bytes in ShiftRows [19]

To improve the security of our AES implementation, the scrambling method can be applied at the bit level.

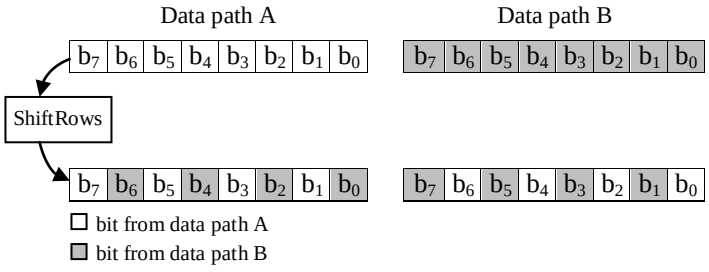

Figure 4. Bit-level scrambling in ShiftRows

The bit level scrambling approach leads to a more robust implementation. Further, from a hardware viewpoint, it is easy to implement and does not increase the complexity.

The scrambling method can also be applied to the InvShiftRows transformation.

C. In MixColumns

The MixColumns is protected by using the cyclic redundancy check (CRC) [15]. According to (4), after adding the columns of S', one reaches the following:

$$\sum_{i=0}^{3} s'_{i,0} = s'_{0,0} \oplus s'_{1,0} \oplus s'_{2,0} \oplus s'_{3,0} = (02 \oplus 03 \oplus 01 \oplus 01)(s_{0,0} \oplus s_{1,0} \oplus s_{2,0} \oplus s_{3,0}) \quad (6)$$

$$\sum_{i=0}^{3} s'_{i,1} = s'_{0,1} \oplus s'_{1,1} \oplus s'_{2,1} \oplus s'_{3,1} = (02 \oplus 03 \oplus 01 \oplus 01)(s_{0,1} \oplus s_{1,1} \oplus s_{2,1} \oplus s_{3,1}) \quad (7)$$

$$\sum_{i=0}^{3} s'_{i,2} = s'_{0,2} \oplus s'_{1,2} \oplus s'_{2,2} \oplus s'_{3,2} = (02 \oplus 03 \oplus 01 \oplus 01)(s_{0,2} \oplus s_{1,2} \oplus s_{2,2} \oplus s_{3,2}) \quad (8)$$

$$\sum_{i=0}^{3} s'_{i,3} = s'_{0,3} \oplus s'_{1,3} \oplus s'_{2,3} \oplus s'_{3,3} = (02 \oplus 03 \oplus 01 \oplus 01)(s_{0,3} \oplus s_{1,3} \oplus s_{2,3} \oplus s_{3,3}) \quad (9)$$

Considering the fact that $01 \oplus 02 = 03$, we have $02 \oplus 03 \oplus 01 \oplus 01 = 01$. Equations (6), (7), (8) and (9) can be rewritten as follows:

$$\sum_{i=0}^{3} s'_{i,0} = \sum_{i=0}^{3} s_{i,0} \quad (10)$$

$$\sum_{i=0}^{3} s'_{i,1} = \sum_{i=0}^{3} s_{i,1} \quad (11)$$

$$\sum_{i=0}^{3} s'_{i,2} = \sum_{i=0}^{3} s_{i,2} \quad (12)$$

$$\sum_{i=0}^{3} s'_{i,3} = \sum_{i=0}^{3} s_{i,3} \quad (13)$$

where $\sum_{i=0}^{3} s'_{i,j}$ and $\sum_{i=0}^{3} s_{i,j}$ are the addition of the column elements of the MixColumns state, and the ShiftRows state, respectively ($0 \le j \le 3$).

According to (10), (11), (12) and (13), the addition of the column elements of ShiftRows are equal to that of the corresponding column of MixColumns. At the end of the MixColumns computation, (10), (11), (12) and (13) are verified and every difference is reported as a fault.

The coefficients of InvMixColumns transformation are 09, 0B, 0D, 0E. The summation of the four coefficients used in decryption process, is also 1. Therefore, the CRC can be applied to the InvMixColumns transformation.

D. In AddRoundKey

The AddRoundKey operation adds the input state (output of the MixColumns) with round key $K=\{k_{0,0},k_{1,0},\ldots,k_{3,3}\}$ to obtain the output of the round.

We apply the cyclic redundancy check (CRC) to (5) to obtain:

$$\sum_{i=0}^{3} c_{i,0} = \sum_{i=0}^{3} s'_{i,0} \oplus \sum_{i=0}^{3} k_{i,0} \quad (14)$$

$$\sum_{i=0}^{3} c_{i,1} = \sum_{i=0}^{3} s'_{i,1} \oplus \sum_{i=0}^{3} k_{i,1} \quad (15)$$

$$\sum_{i=0}^{3} c_{i,2} = \sum_{i=0}^{3} s'_{i,2} \oplus \sum_{i=0}^{3} k_{i,2} \quad (16)$$

$$\sum_{i=0}^{3} c_{i,3} = \sum_{i=0}^{3} s'_{i,3} \oplus \sum_{i=0}^{3} k_{i,3} \quad (17)$$

where $\sum_{i=0}^{3} c_{i,j}$, $\sum_{i=0}^{3} s'_{i,j}$ and $\sum_{i=0}^{3} k_{i,j}$ are the addition of the column elements of the round output, the MixColumns state and the round key, respectively ($0 \le j \le 3$).

At the end of AddRoundKey computation, (14), (15), (16) and (17) are verified and every discrepancy is considered as a fault.

V. Fault Detection

In this section, we describe the results of the simulation experiments which were carried out to evaluate the fault coverage of the proposed fault detection scheme for the encryption process. We injected random faults affecting any of the 128-bit state, with the number of faulty bits ranging from 1 to 20.

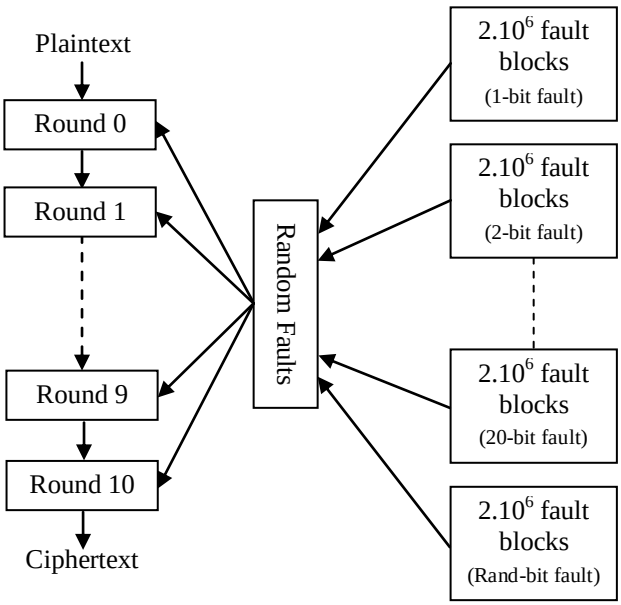

Figure 5. Simulation model

The simulation model is shown in Fig. 5. The fault detection scheme is simulated by 21 tests distinguished by the number of faulty bits in 128-bit AES state. The last test in table 1, labeled as random, used fault patterns with random faulty bit number. Each fault pattern has 2000000 blocks. Faults have been randomly injected, and the faulty operations and the faulty rounds were also randomly chosen. Simulation results are presented in table 1.

As seen in table 1, the percentage of the undetected faults dropped dramatically as the faulty bit number increased. For the random test, the percentage is about 0.001%. That means that the fault coverage achieves 99.999% for the proposed fault detection scheme.

Table 1. Detection Capabilities of the Proposed Fault Detection Scheme

Number of faulty bits	Percentage of the undetected faults
1	0%
2	1.1847%
3	0%
4	0.0851%
5	0%
6	0.0093%
7	0%
8	0.0013%
9	0%
10	0.0003%
11	0%
12	0.0002%
13	0%
14	0.00003%
15	0%
16	0.00003%
17	0%
18	0.00001%
19	0%
20	0.00001%
Random	0.001%

VI. FPGA Impelentation and Comparison

The original AES algorithm design and the proposed secured AES have been described using VHDL, simulated by ModelSim 6.6 and synthesized with Xilinx ISE 10.1.03. The FPGA target was XC5VFX70T from Xilinx Virtex-5 family.

As seen in table 2, the fault Coverage (FC), the number of occupied slices, the frequency (in megahertz), the throughput (in megabits per second), the area overhead and the frequency degradation for the original and secured AES implementation are presented.

Table 2. FPGA Implementation of the Proposed Fault Detection Scheme for the AES

AES Design	FC (%)	Area (slice) (Overhead)	Freq. (MHz) (Degradation)	Throu. (Mbps)
Original AES	-	342	263.92	2815.15
Protected AES	99.999	419 (22.51%)	227.33 (13.86%)	2424.86

The implementation of our original AES takes 342 slices for 263.92 MHz Frequency. The protected AES occupies 22.51% more slices and the frequency decreases by 13.86% than the original AES.

We also compared our proposed method with some previous work for FPGA implementation and the results are shown in table 3.

Table 3. Fault Detection Schemes: Comparison

Fault Detection Schemes	FC (%)	Area Overhead (%)	Frequency Degradation (%)
Algorithm-level [21] *	100	97.6	23.48
Hardware Redundancy *	100	56.7	≈ 0
Structure-Independent [20]	99.996	26.9	≈ 0
Proposed Method	99.999	22.51	13.86

* The percentage are obtained from [20]

Compared to other works, our proposed method has the minimum area overhead and requires less frequency degradation than [21]. From a security viewpoint, the results show that the fault coverage achieves 99.999 % for the proposed scheme.

Therefore, these results show that our work achieves compromise between the implementation cost and the security of the AES.

VII. Conclusion

In this paper, in order to improve the security of the AES, we propose a fault detection scheme against the fault attacks. This method combines several countermeasures presented in the literature.

The proposed method has been implemented on Xilinx Virtex-5 FPGA. Its fault coverage, area overhead and frequency degradation have been obtained and compared.

Compared to some previous works, our proposed method has less area overhead and its fault coverage achieves 99.999%. Therefore the proposed fault detection scheme allows a trade-off between the security and the implementation cost of the AES.

References

[1] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS Publication 197, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[2] H. Mestiri, M. Machhout, R. Tourki, "Performances of the AES design in 0.18µm CMOS technology," IEEE, 7th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2012.

[3] L. Lan, "The AES encryption and decryption realization based on FPGA," Seventh International Conference on Computational Intelligence and Security (CIS 2011), pp. 603-607, 2011.

[4] A. Moh'd, Y. Jararweh and L. Tawalbeh, "AES-512: 512-bit Advanced Encryption Standard algorithm design and evaluation," 7th International Conference on Information Assurance and Security (IAS 2011), pp. 292-297, 2011.

[5] C. Qingfu, L. Shuguo, "A high-throughput cost-effective ASIC implementation of the AES algorithm," 8th International Conference on ASIC (ASICON 2009), pp. 805-808, 2009.

[6] H. Mestiri, N. Benhadjyoussef, M. Machhout and R. Tourki, "A Comparative Study of Power Consumption Models for CPA Attack," International Journal of Computer Network and Information Security, Vol. 5, No. 3, pp. 25-31, 2013.

[7] C. Giraud, "DFA on AES," In H. Dobbertin, V. Rijmen, A. Sowa (Eds.): AES 2004, Lecture Notes in Computer Science, Vol. 3373, pp. 27–41, 2005.

[8] G.Piret and, J.J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad," In Cryptographic Hardware and Embedded Systmes - CHES 2003, Lecture Notes in Computer Science Vol. 2779, pp.77-88, 2003.

[9] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S," ACNS 2003, Lecture Notes in Computer Science Vol. 2846, pp. 293–306, 2003.

[10] A. Moradi, M.T. Manzuri Shalmani, and M. Salmasizadeh, "A Generalized Method of Differential FaultAttack Against AES Cryptosystem," CHES 2006, Lecture Notes in Computer Science Vol. 4249, pp. 91–100, 2006.

[11] J. Takahashi, T. Fukunaga, K. Yamakoshi, "DFA Mechanism on the AES Key Schedule" In IEEE computer society, editor, Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 62 – 74, FDTC 2007.

[12] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault," Available from: http://eprint.iacr.org/2009/575.pdf, 2009.

[13] D. Boneh, R.A. DeMillo, R.J. Lipton, "On the importance of checking cryptographic protocols for faults," EUROCRYPT 1997, Lecture Notes in Computer Science, vol. 1233, pp. 37-51, 1997.

[14] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent Error Detection Schemes of Fault Based Side-Channel Cryptanalysis of Symmetric Block

Ciphers," IEEE Transactions on Computer-Aided Design, Vol 21, N°12, Dec 2002.
[15] C. Yen, and B. Wu, "Simple error detection methods for hardware implementation of Advanced Encryption Standard," IEEE Transactions on Computers, Vol. 55, N°. 6, June 2006.
[16] G.D. Natale, M.L. Flottes, B. Rouzeyre, "On-Line Self-Test of AES Hardware Implementations," DSN'07, Workshop on Dependable and Secure Nanocomputing, Edinburgh, Royaume-Uni, 2007.
[17] J. Rajendran, H. Borad, S. Mantravadi, R. Karri, "SLICED: Slide-based concurrent error detection technique for symmetric block ciphers,"IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 70-75, 2010.
[18] J. Chu, M. Benaissa, "Error Detecting AES using Polynomial Residue Number Systems," Microprocessors and Microsystems, Elsevier, 2012.
[19] M. Joye, P. Manet, and J.B. Rigaud, , "Strengthening hardware AES implementations against fault attacks," IET Information Security, pp. 106-110, Sept, 2007.
[20] M. Mozaffari-Kermani, and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," IEEE Transactions on Computers, Vol. 59, pp. 608-622, 2010.
[21] R. Karri, K. Wu, P. Mishra, and K. Yongkook, "Fault-Based Side Channel Cryptanalysis Tolerant Rijndael Symmetric Block Cipher Architecture," Proceedings. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp.427-435, 2001.
[22] X. Guo, D. Mukhopadhyay, and R. Karri, "Provably Secure Concurrent Error Detection Against Differential Fault Analysis," IACR Cryptology ePrint Archive, Available from:eprint.iacr.org/2012/552.pdf, 2012.

Hassen. Mestiri received his M.S. degree in Microelectronic Systems from the Faculty of Sciences of Monastir, Tunisia, in 2011. Currently, he is a PhD student. His research interests include implementation of standard cryptography algorithm and security of embedded system.

Noura. Benhadjyoussef received MS in Electronic Engineering from National Engineering School of Sousse, Tunisia, in 2010. Currently, she is a PhD student. Her research interests include implementation of cryptography algorithm on FPGA and ASIC, security of smart card and embedded system with ressource contraints.

Mohsen. Machhout was born in Jerba, on January 31 1966. He received MS and PhD degrees in electrical engineering from University of Tunis II, Tunisia, in 1994 and 2000 respectively. Dr Machhout is currently Assistant Professor at University of Monastir, Tunisia. His research interests include implementation of standard cryptography algorithm, key stream generator and electronic signature on FPGA.

Rached. Tourki was born in Tunis, on May 13 1948. He received the B.S. degree in Physics (Electronics option) from Tunis University, in 1970; the M.S. and the Doctorat de 3eme cycle in Electronics from Institut d'Electronique d'Orsay, Paris south University in 1971 and 1973 respectively. From 1973 to 1974 he served as microelectronics engineer in Thomson CSF. He received the Doctorat d'etat in Physics from Nice University in 1979. Since this date he has been professor in Microelectronics and Microprocessors with the physics department, Faculty of Sciences of Monastir. His current research interests include: Digital signal processing and hardware software codesign for rapid prototyping in telecommunications.

