The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.71 e-ISSN: 2251-7545

Security Verification Simulator for Fault Analysis Attacks

Masaya Yoshikawa, Hikaru Goto
Dept. of Information Engineering
Meijo University
Nagoya, Japan

Abstract- **The advanced encryption standard (AES) is the most popular encryption standard in the world. Although the AES algorithm is theoretically safe, it has been recently reported that confidential information could be illegally specified when the AES algorithm is used in electronic circuits. In particular, the menace posed by fault analysis attacks has become extremely serious. This study develops a software simulator to evaluate the vulnerability of a cryptographic circuit against fault analysis attacks in which multiple analytical methods are combined. Simulation results proved the validity of the proposed simulator.**

Keywords-Software simulator, Security verification, Fault analysis attacks, Tamper registance, Cryptographic circuit

I. INTRODUCTION

Credit cards and electronic identification devices use cryptographic circuits to protect confidential information, such as monetary and personal identifiable information. These cryptographic circuits use encryption standards, the theoretical safety of which has been sufficiently verified. Although the encryption standards are theoretically safe, it has been recently reported that confidential information could be illegally specified when the encryption standards are used in electronic circuits. In particular, the menace posed by fault analysis attacks[1]-[15] has become extremely serious. Fault analysis attacks intentionally generate operation errors during the encryption processing and illegally obtain confidential information by pairing an incorrect cryptogram and a correct cryptogram.

Several methods have been proposed for fault analysis attacks. The advanced encryption standard (AES) is the most popular encryption standard in the world. To address fault analysis attacks against AES, researchers have proposed a fault analysis against the key scheduling part and a fault analysis that is based on the estimation of the differences among multiple errors. Therefore, it is important to evaluate the vulnerability of a cryptographic circuit against fault analysis attacks during the circuit's design stage.

The present study develops a simulator to evaluate the vulnerability of a cryptographic circuit against fault analysis attacks in which multiple analytical methods are combined. In the analytical methods that have been previously reported, because the types of information used for analysis differ from each other, the bytes of a secret key, which is intended to be derived, also differ from each other. To improve analytical accuracy, the proposed simulator introduces a new hybrid method that pays attention to the key byte location that is to be derived. The present study also verifies the validity of the proposed simulator by performing several evaluation experiments.

II. ADVANCED ENCRYPTION STANDARD (AES)

AES consists of 128-bit block ciphers, in which a round is composed of SubBytes, ShiftRows, MixColumns, and AddRoundKey processes, and in which data are transformed by repeating the round processing multiple times. The number of rounds is determined according to the key length. The present study adopts the key length of 128 bits, which is a key length that is most often used. In this present case, 10 rounds are used. MixColumns is omitted only at round 10, the final round. For the round processing, the key values used at each round are repeatedly calculated using the KeySchedule process. SubBytes is used for numeric transformation in the form of a byte unit. ShiftRows is used for the shift of a byte location.

III. PROPOSED SIMULATOR

Fault analysis uses the results obtained by incorrect encryption due to an operation error (hereinafter referred to as a fault) during the encryption processing. A fault can be realized by forcibly changing the intermediate value during the encryption processing. When the intermediate value is changed, the subsequent calculation results differ from the normal results; consequently, the cryptogram, which is the final output, differs from the normal cryptogram. Using the cryptogram that has been output due to the fault and the normal cryptogram, a secret key is analyzed, which is the most important element in the cipher.

As shown in Fig.1, AES encryption can be divided into the cryptogram generation and the key scheduling sections. Therefore, in fault analysis against AES, different analytical methods are used, depending on the cryptogram generation section or the key scheduling section in which a fault is to be generated. In the proposed simulator, a hybrid analytical method is introduced in which two different fault analyses are combined.

A. *Fault analysis against the key scheduling section*

Fault analysis against the key scheduling section (hereinafter referred to as key analysis) consists of three steps, (a), (b), and (c), which are explained below.

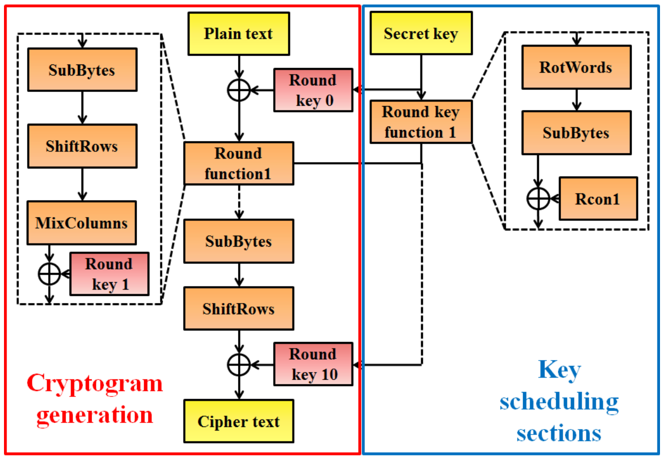


Figure 1 Example of AES encryption

(a) Fault generation and propagation

Key analysis generates a fault as it generates round key 9 and analyzes round key 9. The key value, which has changed due to a fault being mixed into certain bytes of the round key 9, is used for the subsequent calculation and affects the values of other bytes. Actually, the values of round keys 9 and 10 are varied from the normal key values and the cryptogram using the varied key values is also varied. Figure 2 shows the generation processes of round keys 9 and 10 in the key scheduling section.

A block, consisting of four rows and four columns surrounded by a square (since each row and column is a byte, the block is 16 bytes), expresses a round key. For example, when a fault occurs at the first row and the first column of round key 9 (the top left byte is defined as being at the 0th row and 0th column), the results of the fault propagation are exhibited as the shaded area.

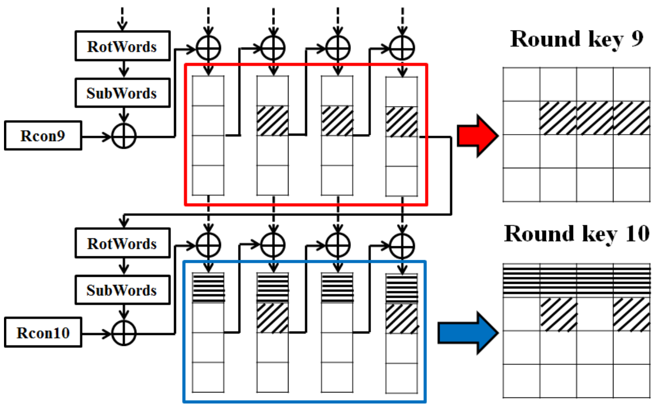
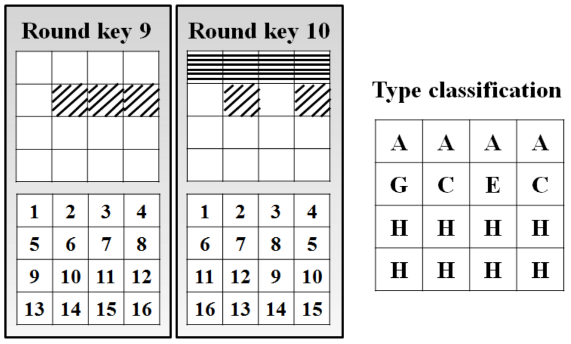


Figure 2 Example of generation processes of round keys 9 and 10 in the key scheduling section

Here, a fault that has passed through a key substitution is expressed in the horizontal line and a fault that has not passed through a key substitution is expressed in the oblique line. This differentiation is required for Step (b), type classification.

(b) Type classification

Based on the results of the fault propagation obtained in Step (a), type classification is performed. Figure 3 shows the results of type classification of the example in Step (a) and the correspondence table. The type classification procedure uses the following steps: (1) the type of a fault (blank or oblique line) at round key 9 is confirmed; its location is the same as that of the block for type classification; (2) the type of a fault (blank, oblique line, horizontal line, or oblique line + horizontal line) of the corresponding block at round key 10 (a block with the same number in this figure) is confirmed; and (3) the confirmed type of a fault is compared with the correspondence table in order to perform type classification.



(1) Results of type classification in Step (a)

		Type of a fault (Round key 10)			
		Blank	Horizontal line	Oblique line	Horizontal line + Oblique line
Type of a fault (Round key 9)	Blank	TypeH	TypeA	TypeG	TypeB
	Oblique line	TypeC	TypeD	TypeE	TypeF

(2) Correspondence table

Figure 3 Results of type classification of the example in Step (a) and the correspondence table

(c) Application of attack rules

The type classification results obtained in Step (b), a cryptogram, in which a fault is mixed, and a normal cryptogram were used to derive the value of round key 9. In actual attacks, it is difficult to generate more than two faults in the same row. The proposed simulator performs an analysis, except for cases where more than two faults are generated in the same row. In the proposed simulator, five analysis (attack) rules are applicable.

Rule1. When types A and B exist in the same row, the fault value of the row can be obtained.

Rule2. When the fault value of a certain row is already known, and type A exists in the upper row, the key value at the third column in the row, where the fault value is already known, can be obtained.

Rule3. When types A and D (or F) exist in the same row, and the key value of the row is already known, the value of a polynomial used in the bytes of the row can be obtained.

Rule4. When two already-known polynomials exist in the same row, the key value or the exclusive disjunction of the key value can be derived by calculating the exclusive disjunction of these polynomials.

Rule5. When the key value at the third column in a certain row is unknown, type A exists in the upper row, and types A and D (or F) exist in the row where an unknown key value exists, the unknown key can be derived by round-robin scheduling (256 ways).

B. Fault analysis against the cryptogram generation section

This analysis used an analytical method that is applicable to multiple faults (hereinafter referred to as finite difference analysis). Finite difference analysis can obtain information about a secret key by estimating the differences among the faults generated in the cryptogram that has been output. This section explains the principle of finite difference analysis.

1) Fault occurrence point

Finite difference analysis defines the time of inputting round 10 as the fault occurrence point and assumes that multiple faults occur in both the cryptographic intermediate value during the processing and in the key value. Figure 4 shows a fault model at the time of inputting round 10. As shown, Error D expresses the fault value that occurred in the cryptographic intermediate value, and Error K expresses the fault value that occurred in the key value.

2) Estimation of difference

The difference at every error byte is estimated based on the cryptogram containing a fault that has been output in the fault model, shown in Fig.4, and the value obtained by performing exclusive disjunction on the normal cryptogram. Three differences (differences A, B, and C) are generated from a one-byte error. The estimation methods of these differences are explained in (i), (ii), and (iii), respectively.

(i) Estimation of difference A

Difference A is defined as a difference in the case where fault values, which have passed through the SubBytes process, are lined up in a row and a fault value exists at the byte of the right endpoint in a row just below the row in which the fault values are lined up.

Difference A is generated when a fault occurs in the rightmost column of the key value. Since the average of the fault values that have passed through the SubBytes process is obtained by round-robin scheduling (from 0 to 255), the average of the hamming weights is 4. Moreover, a high possibility exists that the hamming weight of a fault value, which has not passed through the SubBytes process, is below 2. These characteristics are used to estimate difference A. When one of the following three conditions is satisfied, a difference is judged as difference A:

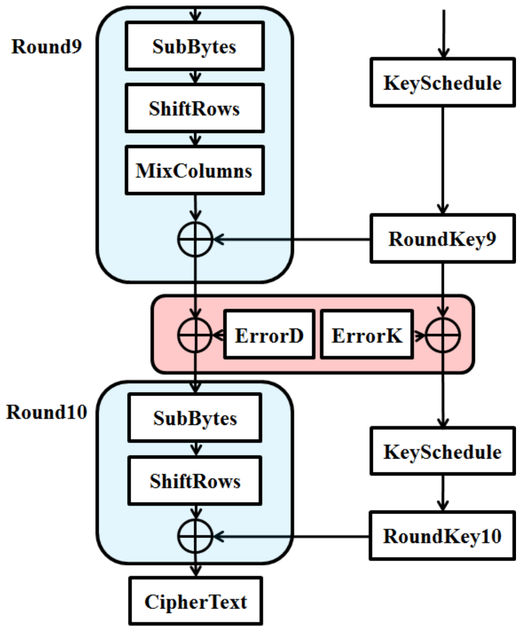


Figure 4 Example of a fault model at the time of inputting round 10

Condition(a). Fault values exist in an entire row and their hamming weights are above 3. Moreover, the hamming weight of a fault value at the right endpoint in a row just below the entire row is below 2.

Condition(b). More than three fault values exist in a row and the hamming distances between these fault values are more than 3. Moreover, the hamming weight of a fault value at the right endpoint, in a row just below the row in which more than three fault values exist, is below 2.

Condition(c). Two different fault values exist in an entire row and the hamming distance between these different fault values is below 2. Moreover, the hamming weight of one of these different fault values is above 3 and the hamming weight of a fault value at the right endpoint, in a row just below the row in which two different fault values exist, is below 2.

Condition (b) is established on the assumption that difference A and difference C occur at the same time. Condition (c) is established on the assumption that difference A and difference B occur at the same time. Figure 5 shows examples of differences that satisfy each condition. In this figure, all the values are expressed in hexadecimal numbers. The judgment method mentioned above cannot be used when other multiple differences occur in a row where fault values, which have passed through the SubBytes process, are lined up.

00	00	00	00
00	00	00	00
57	57	57	57
00	00	00	01

(1) Example of condition (a)

00	00	00	00
00	00	00	00
57	FF	57	57
00	00	00	04

(2) Example of condition (b)

00	00	00	00
00	00	00	00
57	57	5F	5F
00	00	00	02

(3) Example of condition (c)

Figure 5 Examples of differences that satisfy each condition regarding the estimation of difference A

(ii) Estimation of difference B

The estimation of difference B is performed after removing difference A from the differences that occur between a normal cryptogram and a cryptogram in which the faults are mixed. When multiple fault values other than 00 are lined up in a row, these fault values are judged as difference B. Difference B occurs when a fault is generated in a row other than the rightmost column of the key value. When the numbers of bytes used to judge difference B are different from each other, the following two conditions are used for judging difference B:

Condition(a). When more than two fault values are continuously generated from the rightmost column in a row and the hamming weight of the leftmost fault value is below 2, the leftmost fault value is judged as difference B.

Condition(b). When the hamming weight of the leftmost fault value in condition (a) is above 3, all the fault values that are lined up are judged as difference B.

Condition (b) is established on the assumption that difference B and difference C occur at the same time. Figure 6 shows examples of differences that satisfy each condition.

00	00	00	00
00	04	04	04
00	00	00	00
00	00	10	10

(1) Example of condition (a)

4F	08	08	08
00	00	00	00
00	00	00	00
00	AA	01	01

(2) Example of condition (b)

Figure 6 Examples of differences that satisfy each condition regarding the estimation of difference B

(iii) Estimation of difference C

The estimation of difference C is performed after removing difference A and difference B from the differences that occur between a normal cryptogram and a cryptogram in which the faults are mixed. After removing difference A and difference B, all the remaining differences can be considered as difference C. However, there is a possibility that difference A and difference B are incorrectly estimated. To avoid this, the following differences are not judged as difference C:

Condition(a). When a fault value, which has not passed through the SubBytes process of difference A, and an existing byte are superposed.

Condition(b). When a difference is adjacent to difference A.

Condition(c). When a difference and the byte of a fault value, which has passed through the SubBytes process of difference A, are superposed.
Condition(d). When the hamming weight of a fault value is small.

3) *Calculation of key value candidates and the selection of key values*

The candidates for key values are calculated using difference A and difference C obtained in Section (2). For difference A and difference C, the analytical methods proposed by Chen et al. [1] and Giraud [2] are used, respectively. The candidates for key values that are obtained include candidates that are obtained due to incorrect difference estimations. Therefore, the key values are selected from the candidates that are obtained based on the ratio of the number of key values being candidates to the number of key values without being candidates.

C. *Hybrid analysis*

The analytical methods described in Sections 3.1 and 3.2 do not always derive all the key values (16 bytes). In the analytical method described in Section 3.1, the number of derived keys changes according to the fault location. Under the condition that not more than two faults occur in the same row, keys with up to 10 bytes can be derived. However, key byte locations that cannot be derived at any fault location also exist. In the analytical method described in Section 3.2, almost all the key values can be derived when approximately 200 cryptograms are used. However, all the key values are not always derived.

The proposed simulator pays attention to the derived key byte locations and introduces a hybrid fault analysis that can complement the key values using key analysis, which could not be derived using finite difference analysis. Figure 7 shows the configuration of the proposed simulator.

As shown, calculation A of a key value candidate expresses the calculation of a key value candidate using difference C, and calculation B of a key value candidate expresses the calculation of a key value candidate using difference A.

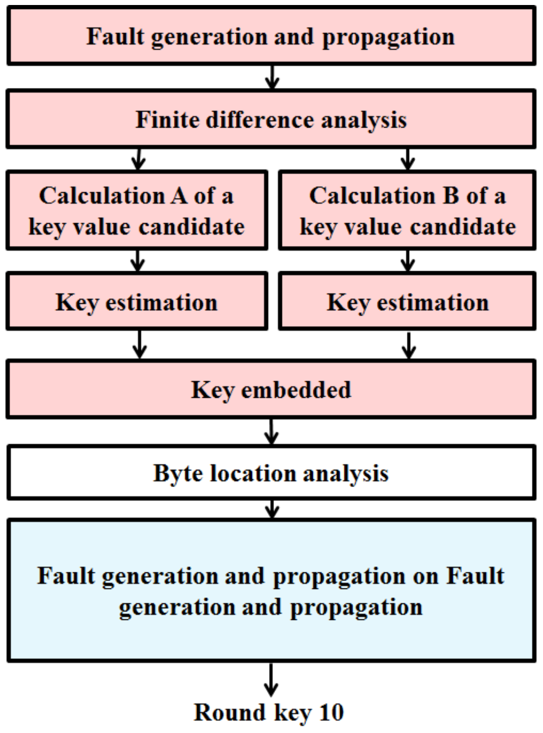

Fault generation and propagation
Finite difference analysis
Calculation A of a key value candidate
Calculation B of a key value candidate
Key estimation
Key estimation
Key embedded
Byte location analysis
Fault generation and propagation on Fault generation and propagation
Round key 10

Figure 7 Configuration of the proposed simulator

IV. EVALUATION EXPERIMENTS

A. *Experimental conditions*

In order to evaluate the proposed simulator, we have conducted several experiments. In the experiments, a conflation method was used as the configuration method for SubBytes transformation in AES. Table 1 shows the detail of the experimental conditions.

TABLE I. EXPERIMENTAL CONDITIONS

Name	Value
Key length	128 bit
Block length	128 bit
Key value	Constant
Input plain text	Random
Fault location	Random
Fault probability	30%-70%
The number of Cipher text	200

B. *Evaluation of the hybrid analytical method*

In order to verify the key analysis, we conducted the several simulations using the key analysis. Figures 8 and 9 show the results. As shown in these figures, the number of derived keys changes according to the fault locations.

Figure 10 shows the number of derived keys obtained using the hybrid analytical method that was newly introduced in the proposed simulator and the number of derived keys obtained using only finite difference analysis. In this figure, the vertical axis represents the number of derived keys and the horizontal axis represents the number of simulations. As shown in Fig.10, the number of derived keys was larger when the hybrid analytical method was used than when the finite difference analysis was used.

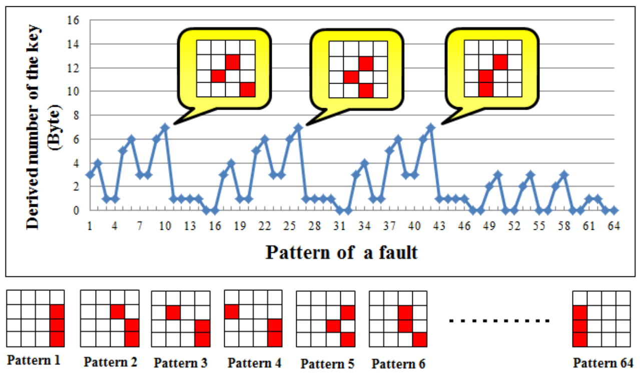


Figure 8 Result of three fault locations

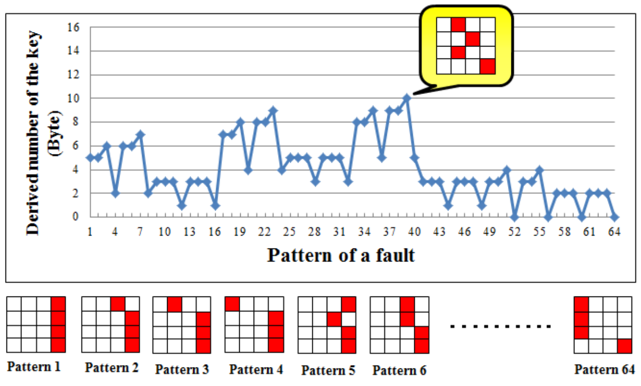


Figure 9 Result of four fault locations

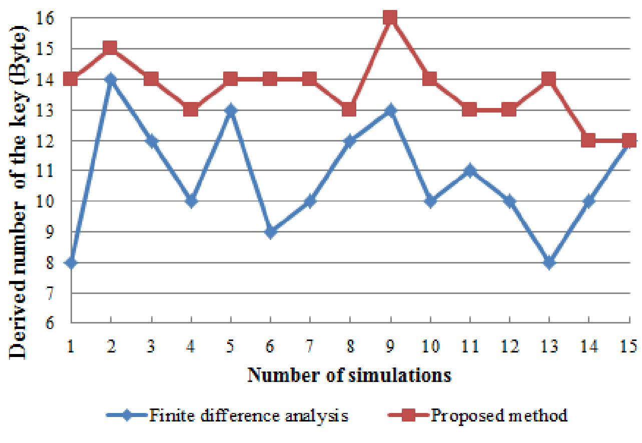


Figure 10 The number of derived keys obtained using the proposed method

This means that the key analysis could complement the key values that could not be derived using the finite difference analysis. Thus, a hybrid analytical method that paid attention to the derived key byte locations could improve the analytical efficiency.

There was a case where the difference in the number of derived keys was small when comparing the results of the hybrid analytical method and the finite difference analysis. Most likely the reason for this was that a condition existed in the key analysis under which the key byte locations could not be derived if more than two faults occurred in the same row.

V. CONCLUSION

The present study developed a new simulator that used a hybrid analytical method in which finite difference analysis and key analysis were hierarchically combined. The proposed simulator paid attention to the byte locations of the keys that could be derived using both analyses so the analytical accuracy was improved. Evaluation experiments confirmed that the proposed hybrid analytical method could complement two analytical methods (finite difference and key analyses), which were the bases of the proposed method.

In the future, we will examine a method to estimate the key values that could not be derived using the proposed method.

ACKNOWLEDGMENT

This study was supported by Japan Science and Technology Agency (JST), Core Research for Evolutional Science and Technology (CREST).

REFERENCES

[1] C.-N. Chen, S.-M. Yen,"Differential fault analysis on AES key schedule and some countermeasures", Proc. 8th Australasian Conf. Information Security and Privacy (ACISP2003), vol.2727, pp.118–129, 2003

[2] C.Giraud,"DFA on AES", Proc. 4th Int. Conf. AdvancedEncryption Standard-AES (AES 2004), vol.3373, pp.27–41,2005.

[3] S.S.Ali, D.Mukhopadhyay, "A Differential Fault Analysis on AES Key Schedule Using Single Fault", Proc. of 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp.35-42, 2011.

[4] Chong Hee Kim, J.J.Quisquater, "Faults, Injection Methods, and Fault Attacks", IEEE Design & Test of Computers, Vol.24, No.6, pp.544-545, 2007.

[5] Gaoli Wang, Shaohui Wang, "Differential Fault Analysis on PRESENT Key Schedule", Proc. of 2010 International Conference on Computational Intelligence and Security (CIS), pp.362-366, 2010.

[6] Wei Li, Dawu Gu, Yong Wang, Juanru Li, Zhiqiang Liu, "An Extension of Differential Fault Analysis on AES", Proc. of Third International Conference on Network and System Security (NSS), pp.443-446, 2009.

[7] P.Maistri, R.Leveugle,"Double-Data-Rate Computation as a Countermeasure against Fault Analysis", IEEE Transactions on Computers, Vol.57, No.11, pp.1528-1539, 2008.

[8] P.Dusart, G.Letourneux, O.Vivolo, "Differential fault analysis on AES", Proc. First Int. Conf. Applied Cryptography and Network Security (ACNS 2003), vol.2846, pp.293–306,2003.

[9] Li Yang, K.Ohta, K.Sakiyama, "New Fault-Based Side-Channel Attack Using Fault Sensitivity", IEEE Trans. on Information Forensics and Security, Vol.7, Issue 1, Part 1, pp.88-97, 2012.

[10] Z.Wang, M.Karpovsky, A.Joshi, "Secure Multipliers Resilient to Strong Fault-Injection Attacks Using Multilinear Arithmetic Codes", IEEE Trans. on Very Large Scale Integration (VLSI) Systems, pp.1-13, 2011.

[11] H.Li, S.Moore, "Security evaluation at design time against optical fault injection attacks", IEE Proc. on Information Security, Vol.153 , Issue 1, pp.3-11, 2006.

[12] A.P.Fournaris, "Fault and simple power attack resistant RSA using Montgomery modular multiplication", Proc. of IEEE International Symposium on Circuits and Systems, pp.1875-1878, 2010.

[13] A.Pellegrini, V.Bertacco, T.Austin, "Fault-based attack of RSA authentication", Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.855-860, 2010.

[14] JeaHoon Park, SangJae Moon, DooHo Choi, YouSung Kang, JaeCheol Ha, "Fault attack for the iterative operation of AES S-Box", Proc. of 5th International Conference on Computer Sciences and Convergence Information Technology, pp.550-555, 2010.

[15] K.J.Kulikowski, Wang Zhen, M.G.Karpovsky, "Comparative Analysis of Robust Fault Attack Resistant Architectures for Public and Private Cryptosystems", Proc. of 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, pp.41-50, 2008.