A New SPN Type Architecture to Strengthen Block Cipher Against Fault Attack

Gitika Maity[1], Jaydeb Bhaumik[2], and Anjan Kundu[3]

(Corresponding author: Gitika Maity)

Computer Science and Engineering, Haldia Institute of Technology[1]

I.C.A.R.E Complex, H.I.T Campus, Hatiberia, PO-HIT, Midnapore, West Bengal 721657, India

(Email: Gitika.Maity@gmail.com)

Department of Electronics & Communications Engineering, Haldia Institute of Technology[2]

Institute of Radio Physics and Electronics, University of Calcutta[3]

(Received Dec. 2, 2016; revised and accepted Apr. 11 & May 14, 2017)

Abstract

In recent years, Differential Fault Analysis (DFA) has been proven as the most efficient technique to attack any block cipher by introducing a computational error. In this paper, a new Substitution Permutation Network (SPN) type architecture is proposed which has better resistance against DFA as compared to Advanced Encryption Standard (AES). The proposed architecture is similar to AES except round key mixing function. Here, round key is mixed with round output, using nonlinear vectorial Boolean function called 'Nmix'. Using 4 faulty-fault free ciphertext pairs, 32 bits of 10^{th} round key is retrieved by injecting a random byte fault at the input of 9^{th} round. The computational complexity will be in the order of 2^{36} to obtain 128 bits 10^{th} round key. Total 16 numbers of faulty and fault free ciphertext pairs are required. Similarly, when a fault is injected at the input of 8^{th} round, then the 10^{th} round key is obtained with computational complexity of 2^{53} and 20 numbers of faulty-fault free ciphertext pairs are required.

Keywords: Block Cipher; Fault Attack; Nonlinear Boolean Function; Substitution and Permutation Network

1 Introduction

Cryptography is an important mathematical tool which is used to provide security in several systems like e-Commerce, RFID, sensor network, mobile phones, smart cards, personal digital assistants (PDAs) etc. Cryptographic algorithms are mainly used to satisfy a subset of four cryptographic properties namely confidentiality, message integrity, authentication and non repudiation. For high speed applications algorithms are usually implemented in hardware. But when implemented in ASIC or FPGA, mathematical security of cryptographic algorithms are not sufficient and hence susceptible to fault attack. The fault is introduced by attacker during the execution of cryptographic algorithm to derive the secret key. This type of fault attack was introduced by Boneh, DeMillo and Lipton [8, 9]. Subsequently Differential Fault Analysis (DFA) on secret key cryptosystem has been discussed by Biham *et al.* in [6]. They shows lower complexity compared to simplified fault attack.

The US National Institute of Standard and Technology (NIST) selected Rijndael as the Advanced Encryption Standard (AES) in 2000 [11]. This algorithm has been adopted as a world wide standard for symmetric key encryption. Till date, fault based attack on advanced encryption algorithm has lowest computational complexity compared to all other attacks. Presently very low cost methods are used for fault injection such as variation of supply voltages, clock glitches, temperature variation, UV light radiations etc. Optimal fault injection method has been reported in [21]. How to find key from algebraic equation, is discussed in [15]. Fault attack by inducing byte level fault at the input of 9^{th} round of AES has been reported in [16], where 250 faulty ciphertexts are needed to recover the key. DFA against AES analyzed by Dusart *et al.* in [7]. They show that by injecting fault at byte level in the 8^{th} round and 9^{th} round, the attacker can derive the key using 40 ciphertext pairs. A survey on fault attack against AES and their counter measures are discussed in [2]. Several counter measures are proposed to resist fault attack on hardware implementation of block cipher AES. Counter measure techniques are hardware redundancy, time redundancy, information redundancy and hybrid redundancy. Differential Fault Analysis on ultra-lightweight cipher PRESENT, has been delineated in [10]. To recover the secret key, it takes 2 faulty encryptions and an exhaustive search of 2^{16}. An improved fault attack against Eta Pairing is described in [13]. An improved fault attack against Miller's algorithm has been presented in [14]. Differential power attack resistant Rijndael circuit is presented in [1]. In [20], it is shown that, by introducing fault at byte level in the 8^{th} round and 9^{th} round of 128 bits AES algorithm, attacker can easily recover the total key using two faulty ciphertexts. A fault based attack on a modified version of AES called MDS-AES has been reported in [12], where one pair of faulty-fault free ciphertext are used to derive 10^{th} round key with a computational complexity of 2^{16}. A complete differential fault analysis against LS-designs and on other families of SPN-based block ciphers have been shown in [17]. They have also validated DFA using a practical example of hardware implementation of SCREAM running on an FPGA. In [24, 19], the block cipher key were deduced by inducing a single random byte fault at the input of the eighth round of the AES algorithm. By exploiting the key-scheduling algorithm, DFA on AES reported in [23, 22]. It takes two faulty ciphertexts and a brute force search of 48 and 40 bits respectively.

In this paper, a modified SPN-type architecture has been proposed to strengthen it against fault attack without affecting area and time significantly. Here XOR operation in AddRoundKey step is replaced by a Boolean nonlinear Nmix function [4]. Effectiveness of the proposed architecture is then analysed against fault attack, by introducing a random byte fault at the input of 9^{th} round and 8^{th} round. The attacker has to search for 2^{36} times to obtain the desired 128 bit key. Also 16 numbers of faulty-fault free ciphertext pairs are necessary, which is much greater than the complexity of attacking original AES. When a random byte fault is introduced at the the input of 8^{th} round, then to recover 32 bits key it takes 5 faulty ciphertext pairs. So, the attacker has to search for approximately 2^{53} times to obtain the 128 bits key and 20 faulty-fault free ciphertext pairs are necessary.

This paper is organized as follows. Following the introduction, a description of proposed SPN type block cipher algorithm is given in Section 2. Fault analysis on proposed SPN-type architecture when fault is injected at the input of 9^{th} and 8^{th} round have been discussed in Sections 3 and 4 respectively. Comparison with existing works is discussed in Section 5. Finally the paper is concluded in Section 6.

2 Description of SPN Type Block Cipher Algorithm

The description of AES-Rijndael algorithm has been provided in detail by Daemen *et al.* in [11]. The proposed SPN type architecture is a modified version of AES-Rijndael algorithm. In the proposed architecture, SubByte, ShiftRow and MixColumn operations are exactly same as in AES. Only round key mixing operation of AES has been modified where XOR operation is replaced by Nmix. In this algorithm, key size and block size are 128 bits. Similar to AES, the 128 bits message block is arranged as a 4×4 array of bytes. The elements of the matrix are represented by variables s_{ij} where $0 \leq i \leq 3$ and $0 \leq j \leq 3$ where i, j denoting the row and column indexes of the state matrix. Number of round is 10 and

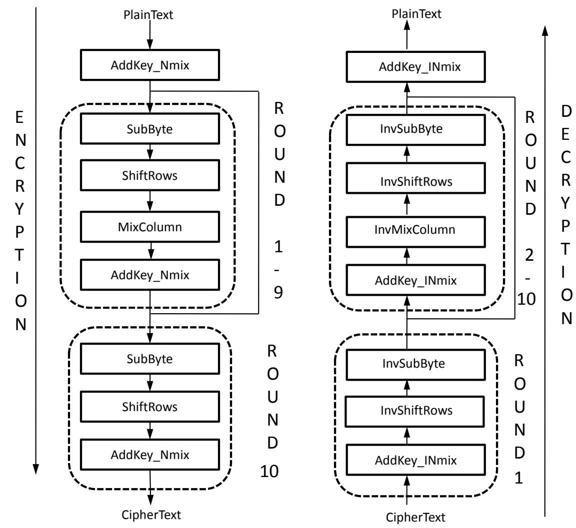

Figure 1: Block diagram of SPN type block cipher algorithm

in each round key is generated from cipher key by key expansion algorithm. At the end of 10^{th} round ciphertext is generated. Similar to AES-128, round 1-9 consists of SubByte, ShiftRow, MixColumn and Round Key Mixing. Round 10 consists of following 3 operations: SubByte, ShiftRows and Round Key Mixing. Basics of each functional block is as follows

SubBytes: This is a non linear substitution step where each byte is replaced by a new byte.

ShiftRows: In this step 2^{nd}, 3^{rd} and 4^{th} rows are circular shifted left by 1, 2 and 3 bytes respectively. First row remains unchanged.

MixColumns: Here the four bytes of each column of the state matrix are multiplied by the following matrix.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

Round Key Mixing: In each round the corresponding byte of state matrix are mixed with the generated roundkey. A non linear function Nmix is used here in AdRoundKey step of encryption. For decryption, the Inverse Nmix (INmix) is used. Details of Nmix and INmix has been discussed in [4]. In [5], Nmix function is used to design an integrated scheme for error correction and message authentication. In [3] nonlinear mixing function Nmix has been used to design the block cipher HDNM8. For the sake of completeness, an overview of Nmix and INmix is given in the following subsections.

2.1 Nonlinear Mixing (Nmix) Function

It operates on two n-bit variables $X = (x_{n-1}x_{n-2}....x_0)$ and $K = (k_{n-1}k_{n-2}....k_0)$ and produces an n-bit output $Y = (y_{n-1}y_{n-2}....y_0)$ where the each output bit y_i and carry bit c_i are defined as

$$\begin{aligned} y_i &= x_i \oplus k_i \oplus c_{i-1} \\ c_i &= \bigoplus_{j=0}^{i} x_j k_j \oplus x_{i-1}x_i \oplus k_{i-1}k_i \end{aligned} \quad (1)$$

where $0 \leq i \leq n-1$, $c_{-1} = 0$, $x_{-1} = 0$, $k_{-1} = 0$ and c_i is the carry propagated from bit position i^{th} to $(i+1)^{th}$. The end around carry c_{n-1} is ignored. Each output y_i is balanced for all $0 \leq i \leq n-1$.

In case of function Nmix, output XOR difference is not equal to input difference (XOR) when single input changes i.e $Nmix(A, K) \oplus Nmix(B, K) \neq A \oplus B$ where A, B and K are three n-bit variables. This property of Nmix function is utilized to strengthen proposed SPN-type architecture against fault attack.

2.2 Inverse Nonlinear Mixing (INmix) Function

INmix takes two n-bit variables $Y = (y_{n-1}y_{n-2}....y_0)$ and $K = (k_{n-1}k_{n-2}....k_0)$ as inputs and produces an $n-bit$ output $X = (x_{n-1}x_{n-2}....x_0)$, where each output bit x_i and carry d_i are defined as

$$\begin{aligned} x_i &= y_i \oplus k_i \oplus d_{i-1} \\ d_i &= \bigoplus_{j=0}^{i} x_j k_j \oplus x_{i-1}x_i \oplus k_{i-1}k_i \end{aligned} \quad (2)$$

where $0 \leq i \leq n-1$, $d_{-1} = 0$, $x_{-1} = 0$, $k_{-1} = 0$ and d_i is the carry propagated from bit position i^{th} to $(i+1)^{th}$. The end around carry d_{n-1} is ignored.

3 Fault Attack on Ninth Round of Proposed SPN-type Architecture

In this section, a single random non zero byte fault is induced in the first byte of 9^{th} round input. Propagation of fault is shown in Figure 2. After SubBytes operation, injected fault f has been change to f'. Fault remains in the same position after ShiftRows and after MixColumns, it is distributed within 4 bytes of 1st column. If the attacker wants to retrieve 128 bit key then 4 byte faults have to be injected at 1^{st}, 5^{th}, 9^{th} and 13^{th} bytes respectively. Assume a fault is injected at the first byte of 9^{th} round input and let the expressions of $CT1$ and $CT2$ are

$$CT_1 = \begin{pmatrix} y_0 & y_4 & y_8 & y_{12} \\ y_1 & y_5 & y_9 & y_{13} \\ y_2 & y_6 & y_{10} & y_{14} \\ y_3 & y_7 & y_{11} & y_{15} \end{pmatrix}$$

$$CT_2 = \begin{pmatrix} (y_0 + F_1) & y_4 & y_8 & y_{12} \\ y_1 & y_5 & y_9 & (y_{13} + F2) \\ y_2 & y_6 & (y_{10} + F_3) & y_{14} \\ y_3 & (y_7 + F_4) & y_{11} & y_{15} \end{pmatrix}$$

The associated keys K_9 and K_{10} of 9^{th} and 10^{th} round are considered as

$$K_9 = \begin{pmatrix} k_{90} & k_{94} & k_{98} & k_{912} \\ k_{91} & k_{95} & k_{99} & k_{913} \\ k_{92} & k_{96} & k_{910} & k_{914} \\ k_{93} & k_{97} & k_{911} & k_{915} \end{pmatrix}$$

$$K_{10} = \begin{pmatrix} k_{100} & k_{104} & k_{108} & k_{1012} \\ k_{101} & k_{105} & k_{109} & k_{1013} \\ k_{102} & k_{106} & k_{1010} & k_{1014} \\ k_{103} & k_{107} & k_{1011} & k_{1015} \end{pmatrix}$$

From the fault pattern shown in Figure 2, following equations are constructed

$$\begin{aligned} &[(INmix(ISB(INmix(y_0, k_{100})), k_{90})) \oplus \\ &(INmix(ISB(INmix(y_0 + F_1), k_{100})), k_{90})] = \\ &2[(INmix(ISB(INmix(y_{13}, k_{1013})), k_{91})) \oplus \\ &(INmix(ISB(INmix(y_{13} + F_2), k_{1013})), k_{91})] \end{aligned} \quad (3)$$

$$\begin{aligned} &(INmix(ISB(INmix(y_{13}, k_{1013})), k_{91})) \oplus \\ &(INmix(ISB(INmix(y_{13} + F_2), k_{1013})), k_{91}) = \\ &(INmix(ISB(INmix(y_{10}, k_{1010})), k_{92})) \oplus \\ &(INmix(ISB(INmix(y_{10} + F_3), k_{1010})), k_{92}) \end{aligned} \quad (4)$$

$$\begin{aligned} &[(INmix(ISB(INmix(y_7, k_{107})), k_{93})) \oplus \\ &(INmix(ISB(INmix(y_7 + F_4), k_{107})), k_{93})] = \\ &3[(INmix(ISB(INmix(y_{13}, k_{1013})), k_{92})) \oplus \\ &(INmix(ISB(INmix(y_{13} + F_2), k_{1013})), k_{92})] \end{aligned} \quad (5)$$

In Equations (3), (4) and (5) the keys k_{100}, k_{107}, k_{1010}, k_{1013} are the 10^{th} round keys and k_{90}, k_{91}, k_{92} and k_{93} are the 9^{th} round keys. From Equations (3), (4) and (5) 4 bytes of 10^{th} round keys can be recovered. Similarly, if an attacker injects a non zero random fault at 5^{th} byte, then another 32 bits key is obtained. Assuming $CT3$ be a fault free ciphertext and $CT4$ the corresponding faulty ciphertext and expressions of $CT3$ and $CT4$ are

$$CT_3 = \begin{pmatrix} y_0 & y_4 & y_8 & y_{12} \\ y_1 & y_5 & y_9 & y_{13} \\ y_2 & y_6 & y_{10} & y_{14} \\ y_3 & y_7 & y_{11} & y_{15} \end{pmatrix}$$

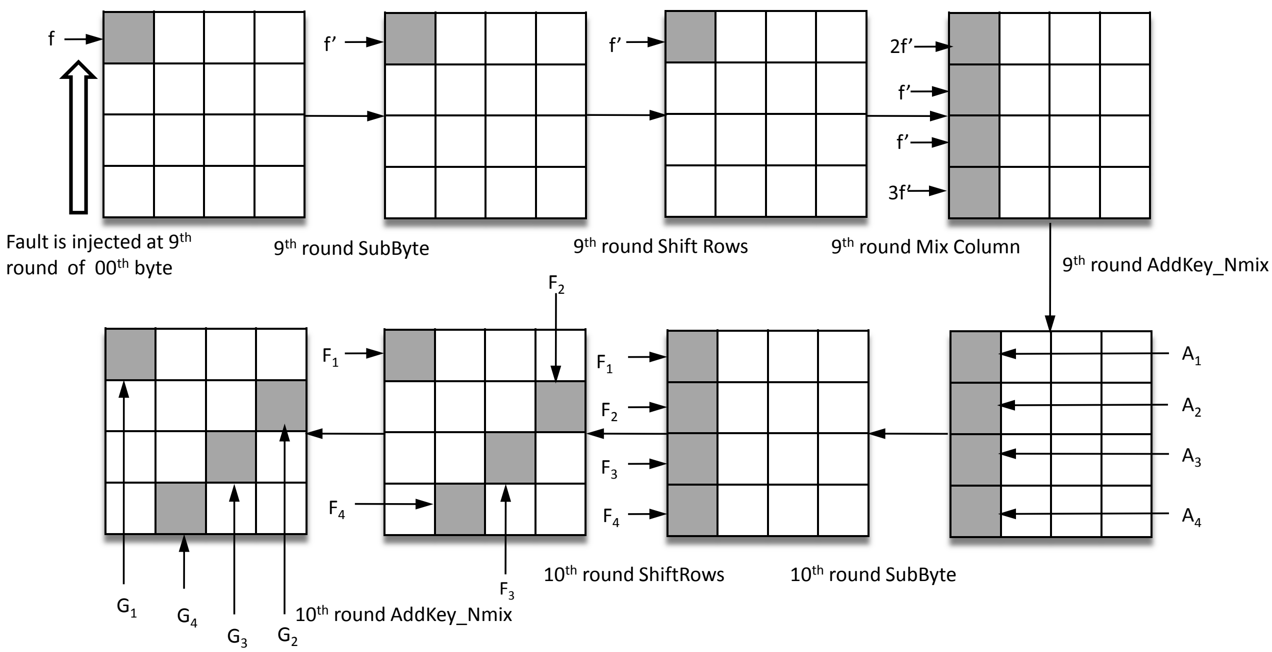

Figure 2: Fault propagation when fault is injected at the 1^{st} byte of 9^{th} round input

$$CT_4 = \begin{pmatrix} y_0 & (y_4+G_1) & y_8 & y_{12} \\ (y_1+G_2) & y_5 & y_9 & y_{13} \\ y_2 & y_6 & y_{10} & (y_{14}+G_3) \\ y_3 & y_7 & (y_{11}+G_4) & y_{15} \end{pmatrix}$$

Following equations are formulated using CT_3 and CT_4.

$$\begin{aligned} &(INmix(ISB(INmix(y_4, k_{104})), k_{94})) \oplus \\ &(INmix(ISB(INmix(y_4+G_1), k_{104})), k_{94})] = \\ &3[(INmix(ISB(INmix(y_{14}, k_{1014})), k_{96})) \oplus \\ &(INmix(ISB(INmix(y_{14}+G_3), k_{1014})), k_{96})] \end{aligned} \tag{6}$$

$$\begin{aligned} &(INmix(ISB(INmix(y_{14}, k_{1014})), k_{96})) \oplus \\ &(INmix(ISB(INmix(y_{14}+G_3), k_{1014})), k_{96}) = \\ &(INmix(ISB(INmix(y_{11}, k_{1011})), k_{97})) \oplus \\ &(INmix(ISB(INmix(y_{11}+G_4), k_{1011})), k_{97}) \end{aligned} \tag{7}$$

$$\begin{aligned} &[(INmix(ISB(INmix(y_1, k_{101})), k_{95})) \oplus \\ &(INmix(ISB(INmix(y_1+G_2), k_{101})), k_{95})] = \\ &2[(INmix(ISB(INmix(y_{14}, k_{1014})), k_{96})) \oplus \\ &(INmix(ISB(INmix(y_{14}+G_3), k_{1014})), k_{96})] \end{aligned} \tag{8}$$

From Equations (6), (7) and (8) another 4 bytes of 10^{th} round keys k_{101}, k_{104}, k_{1011}, k_{1014} can be recovered. Similarly if the attacker inject a non zero random fault at 9^{th} byte then another 32 bits key is obtained. From the fault propagation, following equations are constructed.

$$\begin{aligned} &[(INmix(ISB(INmix(y_5, k_{105})), k_{99})) \oplus \\ &(INmix(ISB(INmix(y_5+H_2), k_{105})), k_{99})] = \\ &3[(INmix(ISB(INmix(y_8, k_{108})), k_{98})) \oplus \\ &(INmix(ISB(INmix(y_8+H_1), k_{108})), k_{98})] \end{aligned} \tag{9}$$

$$\begin{aligned} &(INmix(ISB(INmix(y_{15}, k_{1015})), k_{911})) \oplus \\ &(INmix(ISB(INmix(y_{15}+H_4), k_{1015})), k_{911}) = \\ &(INmix(ISB(INmix(y_8, k_{108})), k_{98})) \oplus \\ &(INmix(ISB(INmix(y_8+H_1), k_{108})), k_{98}) \end{aligned} \tag{10}$$

$$\begin{aligned} &[(INmix(ISB(INmix(y_2, k_{102})), k_{910})) \oplus \\ &(INmix(ISB(INmix(y_2+H_3), k_{102})), k_{910})] = \\ &2[(INmix(ISB(INmix(y_8, k_{108})), k_{98})) \oplus \\ &(INmix(ISB(INmix(y_8+H_1), k_{108})), k_{98})] \end{aligned} \tag{11}$$

From Equation (9), (10) and (11) another 4 bytes 10^{th} round keys k_{102}, k_{105}, k_{108}, k_{1015} can be recovered. Similarly if the attacker inject a non zero random fault at 13^{th} byte then another 32 bits key can be obtained.

Following equations are constructed employing fault propagation diagram

$$[(INmix(ISB(INmix(y_6, k_{106})), k_{914})) \oplus$$

$$\begin{aligned}(INmix(ISB(INmix(y_6+I_3),k_{106})),k_{914})] = \\ 3[(INmix(ISB(INmix(y_{12},k_{1012})),k_{912}))\oplus \\ (INmix(ISB(INmix(y_{12}+I_1),k_{1012})),k_{912})]\end{aligned} \quad (12)$$

$$\begin{aligned}(INmix(ISB(INmix(y_{12},k_{1012})),k_{912}))\oplus \\ (INmix(ISB(INmix(y_{12}+I_1),k_{1012})),k_{912}) = \\ (INmix(ISB(INmix(y_9,k_{109})),k_{913}))\oplus \\ (INmix(ISB(INmix(y_9+I_2),k_{109})),k_{913})\end{aligned} \quad (13)$$

$$\begin{aligned}[(INmix(ISB(INmix(y_3,k_{103})),k_{915}))\oplus \\ (INmix(ISB(INmix(y_3+I_4),k_{103})),k_{915})] = \\ 2[(INmix(ISB(INmix(y_{12},k_{1012})),k_{912}))\oplus \\ (INmix(ISB(INmix(y_{12}+I_1),k_{1012})),k_{912})]\end{aligned} \quad (14)$$

Similarly, another 4 bytes of 10^{th} round keys k_{103}, k_{106}, k_{109}, k_{1012} can be recovered by the attacker employing Equations (12), (13) and (14).

3.1 Working Example

An example is provided in this subsection. Here a fault is injected at 1^{st} byte of 9^{th} round input. Assume $PT1$ is a given plaintext

$$PT_1 = \begin{pmatrix} 2f & cb & c7 & 9e \\ 28 & a0 & 81 & 23 \\ 8e & 9f & bd & 5b \\ 28 & 3e & e4 & 4b \end{pmatrix}$$

and the cipher key K_0 is as follows

$$K_0 = \begin{pmatrix} c7 & bd & d7 & be \\ 9b & d9 & 9b & 9c \\ da & cd & 6c & fa \\ bc & 28 & f8 & 9c \end{pmatrix}$$

The 9^{th} round key is obtained by employing AES key expansion algorithm is as follows

$$K_9 = \begin{pmatrix} af & 35 & 24 & 90 \\ f0 & fa & 45 & 99 \\ a7 & 87 & 34 & 33 \\ d8 & 17 & be & 59 \end{pmatrix}$$

and the 10^{th} round key is as follows

$$K_{10} = \begin{pmatrix} 77 & 42 & 66 & f6 \\ 33 & c9 & 8c & 15 \\ 6c & eb & df & ec \\ b8 & af & 11 & 48 \end{pmatrix}$$

Corresponding fault free ciphertext is as follows

$$CT_1 = \begin{pmatrix} ca & ea & 6f & 6d \\ fe & cb & 3f & 16 \\ 27 & 89 & 26 & 6d \\ 8a & 62 & 2e & d0 \end{pmatrix}$$

The corresponding faulty ciphertext after injecting fault at 1^{st} byte of 9^{th} round is as follows

$$CT_1' = \begin{pmatrix} \mathbf{71} & ea & 6f & 6d \\ fe & cb & 3f & \mathbf{a6} \\ bb & 71 & \mathbf{8e} & 11 \\ 8a & \mathbf{bd} & 2e & d0 \end{pmatrix}$$

Bolded bytes show how the faults have been propagated in the ciphertext.

Let another plaintext be

$$PT_2 = \begin{pmatrix} a8 & f4 & bc & 6c \\ cd & c3 & 76 & aa \\ e7 & 80 & af & d5 \\ 0b & a1 & 9a & e1 \end{pmatrix}$$

The corresponding ciphertext is for the same cipher key is

$$CT_2 = \begin{pmatrix} eb & 8d & 5a & 15 \\ c6 & cd & a4 & ba \\ 27 & 89 & 26 & 6d \\ af & ae & b3 & 1b \end{pmatrix}$$

The corresponding faulty ciphertext after injecting fault at 1^{st} byte of 9^{th} round input is as follows

$$CT_2' = \begin{pmatrix} \mathbf{75} & 8d & 5a & 15 \\ c6 & cd & a4 & \mathbf{2e} \\ 27 & 89 & \mathbf{0a} & 6d \\ af & \mathbf{86} & b3 & 1b \end{pmatrix}$$

First by using equation 3 and a pair of faulty-fault free ciphertext, set of $k_{90}, k_{91}, k_{100}, k_{1013}$ values are obtained. Then by using another faulty-fault free ciphertext pair another set of values of k_{90}, k_{91}, k_{100} and k_{1013} are obtained which are satisfying Equation (3). Intersection of these two sets produces a reduced set of values k_{90}, k_{91}, k_{100} and k_{1013}. A third set of $k_{90}, k_{91}, k_{100}, k_{1013}$ values are obtained from another pair of faulty-fault free ciphertext and reduced key set and then by intersecting more reduce key set is obtained. And finally using 4^{th} pair of ciphertext, a set of $k_{90}, k_{91}, k_{100}, k_{1013}$ values are obtained and set of values obtained in previous step are intersected to obtain correct 10^{th} round 16 bits key.

In this way, employing Equations (4) and (5) and 4 faulty-fault free ciphertext pairs similar analysis is done and finally, four bytes $k_{100}, k_{107}, K_{1010}$ and K_{1013} of 10^{th} round keys are obtained correctly.

To recover the set of $k_{100}, k_{107}, k_{1010}, k_{1013}$ keys, it needs computational complexity of 4×2^{32} i.e. 2^{34}. Computational complexity of 16×2^{32} i.e. 2^{36} is required to obtain 128-bits key.

4 Fault Attack on Eighth Round of Proposed SPN Architecture

In proposed SPN architecture, a non-zero fault has been induced at the input of 8^{th} round. After 8^{th} round MixColumn step, the fault is distributed into 4 bytes. Again

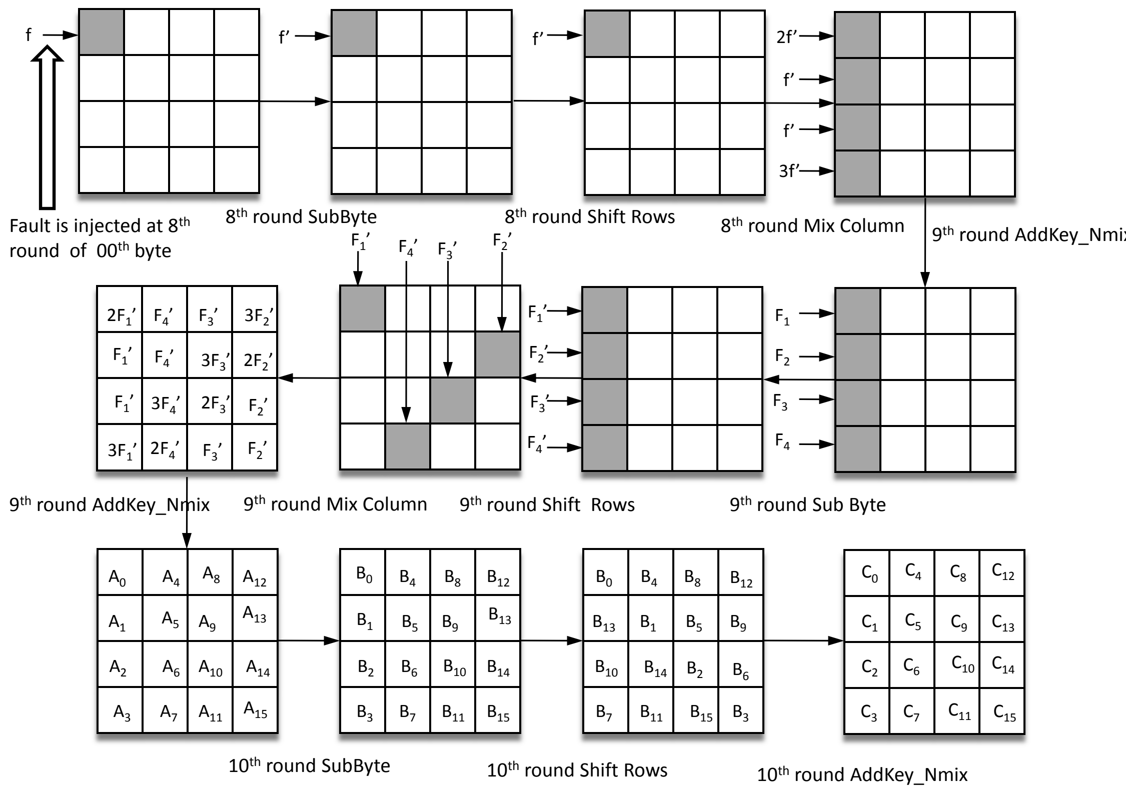

Figure 3: Fault propagation when fault is injected at the input of 8^{th} round

after MixColumn step of 9^{th} round the fault is spreaded throughout all the bytes of state matrix as shown in Fig.3.

Assume $CT1$ is a fault free ciphertext

$$CT_1 = \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}$$

If $CT2$ is the corresponding faulty ciphertext, then it can be expressed in following matrix form

$$CT_2 = \begin{pmatrix} x_0+F_0 & x_4+F_4 & x_8+F_8 & x_{12}+F12 \\ x_1+F_1 & x_5+F_5 & x_9+F_9 & x_{13}+F13 \\ x_2+F_2 & x_6+F_6 & x_{10}+F_{10} & x_{14}+F14 \\ x_3+F_3 & x_7+F_7 & x_{11}+F_{11} & x_{15}+F15 \end{pmatrix}$$

Let the associated round keys are K_8, K_9 and K_{10} in 8^{th}, 9^{th} and 10^{th} rounds respectively

$$K_8 = \begin{pmatrix} k_{80} & k_{84} & k_{88} & k_{812} \\ k_{81} & k_{85} & k_{89} & k_{813} \\ k_{82} & k_{86} & k_{810} & k_{814} \\ k_{83} & k_{87} & k_{811} & k_{815} \end{pmatrix}$$

$$K_9 = \begin{pmatrix} k_{90} & k_{94} & k_{98} & k_{912} \\ k_{91} & k_{95} & k_{99} & k_{913} \\ k_{92} & k_{96} & k_{910} & k_{914} \\ k_{93} & k_{97} & k_{911} & k_{915} \end{pmatrix}$$

$$K_{10} = \begin{pmatrix} k_{100} & k_{104} & k_{108} & k_{1012} \\ k_{101} & k_{105} & k_{109} & k_{1013} \\ k_{102} & k_{106} & k_{1010} & k_{1014} \\ k_{103} & k_{107} & k_{1011} & k_{1015} \end{pmatrix}$$

From the fault pattern, following equations are constructed

$$[INmix(a, k_{80}) \oplus (INmix(b, k_{80}))] = 2[INmix(c, k_{81}) \oplus INmix(d, k_{81})] \quad (15)$$

Where,
$a = (ISB((INmix(ISB(INmix(x_0, k_{100})), k_{90})))$,
$b = (ISB((INmix(ISB(INmix(x_0 + F_0), k_{100})), k_{90}))$
$c = (ISB((INmix(ISB(INmix(x_{13}, k_{1013})), k_{91})$ and
$d = ISB((INmix(ISB(INmix(x_{13} + F_{13}), k_{1013})), k_{91}))$

$$(INmix(e, k_{81}) \oplus (INmix(f, k_{81}) = (INmix(g, K_{82}) \oplus (INmix(h, K_{82}) \quad (16)$$

Where,
$e = ISB(INmix(ISB(INmix(x_{13}, k_{1013})), k_{91}))$,
$f = ISB(INmix(ISB(INmix(x_{13} + F_{13}), k_{1013})), k_{91})$,
$g = ISB(INmix(ISB(INmix(x_{10}, k_{1010})), k_{92}))$ and
$h = ISB(INmix(ISB(INmix(x_{10} + F_{10}), k_{1010})), k_{92})$.

$$[(INmix(i, k_{83}) \oplus (INmix(j, k_{83})] =$$

$$3[(INmix(k, K_{81}) \oplus (INmix(l, K_{81})] \quad (17)$$

Where,
$i = ISB(INmix(ISB(INmix(x_7, k_{107})), k_{93}))$,
$j = ISB(INmix(ISB(INmix(x_7 + F_7), k_{107})), k_{93})$,
$k = ISB(INmix(ISB(INmix(x_{13}, k_{1013})), k_{91}))$ and
$l = ISB(INmix(ISB(INmix(x_{13} + F_{13}), k_{1013})), k_{91})$.
Five pairs of fault free-faulty ciphertexts are needed to recover 32 bits of 10^{th} round key. To recover 32 bits key, computational complexity of 5×2^{48} i.e. 2^{51} is required. From the fault distribution, similarly another set of 9 equations can be constructed and from these equations, rest of the keys can be recovered. To recover 128 bits 10^{th} round keys, total 20 faulty-fault free ciphertext pairs are necessary and computational complexity is 20×2^{48} i.e. 2^{53}.

5 Comparison with Existing Works

In this section, a comparison is provided in terms of fault model, fault location, number of faulty encryptions and computational complexity, of our work and with the works reported in [7, 15, 16, 19, 20]. Existing related works either based on byte level or bit level fault model. Our work is based on byte level fault model. From Table 1, it is observed that in proposed architecture, 16 and 20 faulty-fault free ciphertext pairs are necessary to mount fault attack by injecting fault at the input of 9^{th} and 8^{th} round respectively. Also fault attack complexity in proposed scheme is relatively higher than that of AES [19]. Fault attack on AES [19] requires minimum 2 faulty-fault free ciphertext pairs with complexity 2^{32}. Fault attack on MDS-AES [12] needs 2 faulty cipher text pairs with brute-force search of complexity 2^{16}. Whereas to mount fault attack on proposed SPN-type architecture minimum 16 faulty-fault free ciphertext pairs are necessary with complexity 2^{36}.

6 Conclusion

In this paper, a new SPN-type architecture has been proposed to improve the security of block cipher against fault attack. Here, instead of linear round key mixing function, first time effect of nonlinear round key mixing function is used and analysed, to protect fault attack. Proposed architecture also provides better security against fault attack compared to AES. To derive 128 bits 10^{th} round key it needs computational complexity of 2^{36} and 16 faulty-fault free ciphertext pairs, when fault is injected at input of 9^{th} round. When a fault is introduced at input of 8^{th} round then it needs computational complexity of 2^{53} and 20 faulty-fault free ciphertext pairs, to recover 128 bits of 10^{th} round key.

References

[1] M. Alam, S. Ghosh, D. R. Choudhury, and I. Sengupta, "First-order DPA vulnerability of Rijndael: Security and area-delay optimization trade-off," *International Journal of Network Security*, vol. 15, no. 3, pp. 219-230, 2013.

[2] S. Ali, X. Guo, R. Karri, and D. Mukhopadhyay, "Fault attacks on AES and their countermeasures," in *Secure System Design and Trustable Computing*, pp. 163-208, 2016.

[3] J. Bhaumik, D. R. Chowdhury, "HDNM8: A round-8 high diffusion block cipher with nonlinear mixing function," *Springer Proceedings in Mathematics and Statistics*, vol. 91, pp. 41-55, 2013.

[4] J. Bhaumik, D. R. Chowdhury, "NMIX: An ideal candidate for key mixing," in *Proceedings of the International Conference on Security and Cryptography*, pp. 285-288, 2009.

[5] J. Bhaumik, D. R. Chowdhury, "An integrated ECC-MAC based on RS code," *Transactions on Computational Science*, LNCS 5430, pp. 117-135, 2009.

[6] E. Biham, A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Annual International Cryptology Conference (CRYPTO'97)*, LNCS 1294, pp. 513-525, 1997.

[7] J. Blomer, J. P. Seifert, "Fault based cryptanalysis of the advanced encryption standard (AES)," in *International Conference on Financial Cryptography (FC'03)*, pp. 162-181, 2003.

[8] D. Boneh, R. A. DeMillo, R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *Journal of Cryptology*, vol. 12, pp. 241-246, 2001.

[9] D. Boneh, R. A. DeMillo, R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'97)*, LNCS 1233, pp. 37-51, 1997.

[10] J. Breier, W. He, "Multiple fault attack on PRESENT with a hardware trojan implementation in FPGA," in *International Workshop on Secure Internet of Things (SIoT'15)*, pp. 58-64, 2015.

[11] J. Daemen, V. Rijmen, *The Design of Rijndael*, Springer, Heidelberg 2002.

[12] S. Das, J. Bhaumik, "A fault based attack on MDS-AES," *International Journal of Network Security*, vol. 16, no. 3, pp. 193-198, 2014.

[13] Y. Dou, J. Weng, Y. Wei, and C. Ma, "Improved fault attack against Eta pairing," *International Journal of Network Security*, vol. 16, no. 1, pp. 71-77, 2014.

[14] Y. Dou, J. Weng, Y. Wei, and C. Ma, "Fault attack against Miller's algorithm for even embedding degree," *International Journal of Network Security*, vol. 16, no. 3, pp. 199-207, 2014.

[15] P. Dusart, G. Letourneux and O. Vivolo, *Differential Fault Analysis on A.E.S.*, 2002. (`http://eprint.iacr.org/2003/010`)

Table 1: Comparison of Fault attack on AES with our proposed SPN type architecture accomplishing properties of the encryption function

Reference	Fault Model	Algorithm	Fault Location	No. of Faulty Encryptions	Complexity
[7]	Force 1 bit to 0	AES	Chosen	128	
[7]	Implementation Dependent	AES	Chosen	256	
[16]	Switch 1 bit	AES	Any bit of chosen bytes	50	
[16]	Disturb 1 byte	AES	Anywhere among 4 bytes	250	
[15]	Disturb 1 byte	AES	Anywhere between last two MixColumn	40	
[20]	Disturb 1 byte	AES	Anywhere between 7^{th} round and 7^{th} round MixColumn	2	
[19]	Disturb 1 byte	AES	Anywhere between 7^{th} round and 7^{th} round MixColumn	2	2^{32}
[12]	Disturb 1 byte	MDS-AES	Input of 9^{th} round	2	2^{16}
This paper	Disturb 1 byte	Proposed SPN	Input of 9^{th} round	16	2^{36}
This paper	Disturb 1 byte	Proposed SPN	Input of 8^{th} round	20	2^{53}

[16] C. Giraud, "DFA on AES," *Cryptology ePrint Archive*, Report 2003/008.

[17] B. Lac, A. Canteaut, J. J. A. Fournier, *DFA on LS-designs with a Practical Implementation on SCREAM (extended version)*, Dec. 25, 2017. (`http://eprint.iacr.org/2017/076.pdf`)

[18] A. Mirsaid, T. Gulom, "The encryption algorithm AES-RFWKPES32-4," *International Journal of Electronics and Information Engineering*, vol. 5, no. 1, pp. 20-29, 2016.

[19] D. Mukhopadhyay, "An improved fault based attack of the advanced encryption standard," in *Advances in Cryptography (AFRICACRYPT'09)*, LNCS 5580, pp. 421-434, 2009.

[20] G. Piret, J. J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and Khazad," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03)*, pp. 77-88, 2003.

[21] S. Skorobogatov, R. Anderson, "Optical fault induction attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02)*, pp. 2-12, 2003.

[22] J. Takahashi, T. Fukunaga, *Differential Fault Analysis on the AES Key Schedule*, 2007. (`http://eprint.iacr.org/2007/480`)

[23] J. Takahashi, T. Fukunaga, K. Yamakoshi, "DFA mechanism on the AES schedule," in *Proceedings of 4th International Workshop on Fault Detection and Tolerance in Cryptography*, pp. 27-41, 2007.

[24] M. Tunstall, D. Mukhopadhyay, S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," *Information Security Theory and Practice*, vol. 6633, pp. 224-233, 2011.

Biography

Gitika Maity is currently working as an Asistant Professor in the Department of Computer Science and Engineering, Haldia Institute of Technology, Haldia, India. She received her B. Tech. and M. Tech. Degrees from West Bengal University of Technoloy, India. Her research interests include Cryptography, Cellular Automata and Digital VLSI Design.

Jaydeb Bhaumik is currently working as a Professor and Head in the Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, India. He received his B. Tech. and M. Tech. degrees in Radio Physics and Electronics from University of Calcutta in 1996 and 1999 respectively and PhD degree from Indian Institute of Technology Kharagpur in 2010. His research interests include Cryptography, Cellular Automata, Error Correcting Codes and Digital VLSI Design. He is a member of IEEE and life member of Cryptology Research Society of India since 2008. He has published more than 40 research papers in international journals and conferences.

Anjan Kumar Kundu received his B. Tech.and M. Tech. degree, in Radio Physics and Electronics from the University of Calcutta, India. He has more than ten years of experience in the field of teaching and research. He joined the Radio Physics and Electronics in 2005 as Lecturer and serving the department till date. His research interest includes Microwave Tomography, Cryptography and RFID, Electromagnetics and Microwave engineering. He has published several papers in national and international journals and conferences.