# B. TECH. PROJECT REPORT
# On
# Countermeasures against DFA on PRINCE Cipher


BY

**Ayush Prasad, 160001013**

**Ayush Mantri, 160001012**



**DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY INDORE**


**November 2019**

# Countermeasures against DFA on PRINCE Cipher

**A PROJECT REPORT**

*Submitted in partial fulfillment of the requirements for the award of the degrees*

***of***
**BACHELOR OF TECHNOLOGY**
**in**

**COMPUTER SCIENCE AND ENGINEERING**

*Submitted by:*
**Ayush Prasad, 160001013**
**Ayush Mantri, 160001012**

*Guided by:*
**Dr. Bodhisatwa Mazumdar**

**INDIAN INSTITUTE OF TECHNOLOGY INDORE**
**November 2019**

## CANDIDATE’S DECLARATION

We hereby declare that the project entitled **“Countermeasures against DFA on PRINCE cipher”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Dr. Bodhisatwa Mazumdar, Assistant Professor, Computer Science and Engineering,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

**Ayush Prasad, 160001013** **Ayush Mantri, 160001012**

---

## CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my knowledge.

**Dr. Bodhisatwa Mazumdar**
**Assistant Professor**
**Computer Science and Engineering**
**IIT Indore**

# Preface

This report on “Countermeasures against DFA on PRINCE cipher" is prepared under the guidance of Dr. Bodhisatwa Mazumdar.

Through this report, we have provided some countermeasures against DFA on PRINCE cipher, maintaining its security, robustness, and cryptographic properties. Modifications that we have suggested are feasible and practical. We have also described approaches and ideas which have led us to the conclusion of our work.

We have tried to the best of our abilities and knowledge to explain the content lucidly. We have also added tables and figures to it more illustrative.

**Ayush Prasad, 160001013**

**Ayush Mantri, 160001012**

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

# Acknowledgements

We wish to thank Dr. Bodhisatwa Mazumdar for his kind support and valuable guidance.

It is his help and support, due to which we became able to complete the design and technical report.

Without his support this report would not have been possible. It was only possible because of his enthusiasm, beforehand schedule, knowledge and sincerity towards us and our work to produce better results. We wouldn't have achieved our goals without his encouragement at each step.

**Ayush Prasad, 160001013**

**Ayush Mantri, 160001012**

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

## Abstract

PRINCE is a new lightweight block cipher proposed at the ASIACRYPT'2012 conference. In this report, observations on the DFA fault model [2] of the cipher are presented. Based on the observations, changes are proposed in the core while maintaining its cryptographic properties. These proposed changes will impede the current attack against specific bit faults. We have also included some work that may not have provided desired results but have significance in validating the robustness of its components and the features they provide. The results here show and compare the residual key search space for both the current PRINCE cipher and one with the proposed changes.

# Table of Contents

# List of Figures

## List of Tables

# I. Introduction

Differential fault analysis (DFA) is a type of side-channel attack in the field of cryptography, specifically cryptanalysis. The principle is to induce faults under unexpected environmental conditions into cryptographic implementations, to reveal their internal states.

The idea of injecting faults during the execution of cryptographic algorithms to retrieve the key was first introduced by Boneh, DeMillo, and Lipton who succeeded in breaking a CRT version of RSA [8]. Later, Biham and Shamir adapted this idea to differential analysis on block ciphers and introduced the concept of Differential Fault Attack (DFA) [9]. Block ciphers implemented on smart cards and other low-end devices are vulnerable to such attacks, which exploit the links between right ciphertexts and the faulty counterparts. Usually, the faults are injected by disturbing the power supply voltage, the frequency of the external clock, or by applying a laser beam, etc. [10].

PRINCE is a novel lightweight block cipher proposed in 2012 [1], which is optimized for latency when implemented in hardware. PRINCE is the first lightweight block cipher that takes latency as the main priority.

In this thesis, we have provided the countermeasures against DFA on PRINCE cipher by analyzing its Confusion Layer and Diffusion Layer. We have first analyzed the attack strategy and the properties it exploits. Then we have worked on both the layers to make suitable modifications to impede the attack. Different approaches have been tried to find suitable results. In the further sections of this thesis, we have given a detailed description of the approaches and their results.

## II. Brief Description of PRINCE

PRINCE is a 64-bit block cipher with a 128-bit key. The key schedule is very simple, namely, the 128-bit key is split into two 64-bit parts:

$$k = k_0 \,||\, k_1 \,,$$

and extended to 192 bits by the following mapping:

$$(k_0 \,||\, k_1) \rightarrow (k_0 \,||\, k_0' \,||\, k_1) := (k_0 \,||\, (k_0 \ggg 1) \oplus (k_0 \gg 63) \,||\, k_1).$$

During the encryption the first two subkeys $k_0$ and $k_0'$ are used as pre- and post- whitening keys respectively, while the third subkey $k_1$ is the key for a 12-round block cipher referred to as $\text{PRINCE}_{\text{core}}$ . The high level structure of PRINCE is demonstrated in Fig. II.1.


**Fig. II.1.** The high level structure of PRINCE

**Specification of $\text{PRINCE}_{\text{core}}$ :**

The 12-round process of $\text{PRINCE}_{\text{core}}$ is depicted in Fig. II.2. Each round of $\text{PRINCE}_{\text{core}}$ consist of a key addition, an Sbox-layer, a linear layer, and the

addition of a round constant. The intermediate computation result, called state is usually represented by a 64-bit vector or a 16-nibble vector.

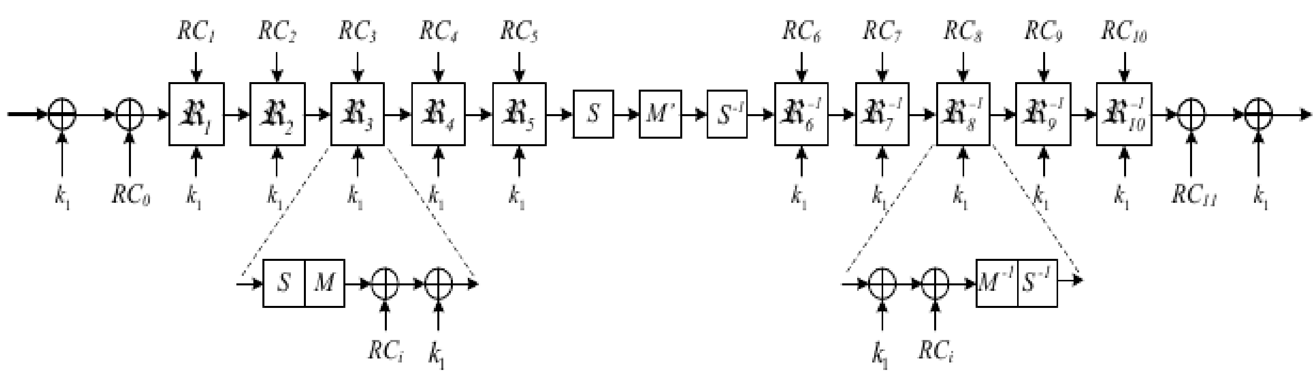



**Fig. II.2.** $\text{PRINCE}_{\text{core}}$

**Sbox-layer:**

The cipher uses a 4-bit Sbox, which is given in Table II.1. The table gives the action of the Sbox in hexadecimal notation. We denote the Sbox and its inverse by S and $S^{-1}$, respectively.

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | B | F | 3 | 2 | A | C | 9 | 1 | 6 | 7 | 8 | 0 | E | 5 | D | 4 |

**Table II.1.** The Sbox S of PRINCE

**Liner layer:**

The linear layer uses a matrix $M$ ($M = SR \circ M'$, $SR$ is shift rows operation) or $M'$ and is called $M$- or $M'$-mapping. In the linear layer, the 64-bit state is multiplied with $M$ or $M'$, both of which are 64 × 64 matrices and built from four 4×4 matrices. These four matrices are given in Fig. II.3.

$$M_0 = \begin{bmatrix} 0&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix}, M_1 = \begin{bmatrix} 1&0&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix}, M_2 = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{bmatrix}, M_3 = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \end{bmatrix}$$

**Fig. II.3.** Base matrices of $M'$

Where these matrices are used to construct two matrices $\hat{M}^{(0)}$ and $\hat{M}^{(1)}$ of size 16 × 16, as shown below in Fig. II.4. Then, the 64 × 64 matrix $M'$ is constructed as a block diagonal matrix with $\hat{M}^{(0)}$, $\hat{M}^{(1)}$, $\hat{M}^{(1)}$, $\hat{M}^{(0)}$ as its diagonal blocks. Note that $M'$ is an involution matrix, namely, $M'\,M' = I$ is the identity matrix.

$$\hat{M}^{(0)} = \begin{bmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{bmatrix}, \hat{M}^{(1)} = \begin{bmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{bmatrix}$$

**Fig. II.4.** Diagonal matrices of $M'$

The $M$-mapping is the composition of the $M'$-mapping and a permutation SR, i.e. $M = SR \circ M'$. $SR$ behaves like the AES shift rows and permutes the 16 nibbles of the state as $(a_0, a_1, \cdots, a_{15}) \rightarrow (a_0, a_5, \cdots, a_{11})$, where the subscripts are changed according to Table II.2. The inverse of SR is denoted by $SR^{-1}$.

| $a_i$-input | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a_i$-output | 0 | 5 | 10 | 15 | 4 | 9 | 14 | 3 | 8 | 13 | 2 | 7 | 12 | 1 | 6 | 11 |

**Table II.2.** The *SR* operation of PRINCE

**$k_1$ and $RC_i$ addition:**

In $k_1$ -add step, the 64-bit state is xored with the 64-bit subkey $k_1$.

In the $RC_i$ -add step, a 64-bit round constant is xored with the state. We define the constants in Table II.3. in hex notation. Note that, for all $0 \leq i \leq 11$, $RC_i \oplus RC_{11-i}$ is the constant $\alpha = c0ac29b7c97c50dd$, $RC_0 = 0$ and that $RC_1, \ldots, RC_5$ and $\alpha$ are derived from the fraction part of $\pi = 3.141\ldots$..

| | |
|---|---|
| $RC_0$ | 0000000000000000 |
| $RC_1$ | 13198a2e03707344 |
| $RC_2$ | a4093822299f31d0 |
| $RC_3$ | 082efa98ec4e6c89 |
| $RC_4$ | 452821e638d01377 |
| $RC_5$ | be5466cf34e90c6c |
| $RC_6$ | 7ef84f78fd955cb1 |
| $RC_7$ | 85840851f1ac43aa |
| $RC_8$ | c882d32f25323c54 |
| $RC_9$ | 64a51195e0e3610d |
| $RC_{10}$ | d3b5a399ca0c2399 |
| $RC_{11}$ | c0ac29b7c97c50dd |

**Table II.3.** Round Constants

From the fact that the round constants satisfy $RC_i \oplus RC_{11-i}=\alpha$ and that $M'$ is an involution, we deduce that the core cipher is such that the inverse of $\text{PRINCE}_{core}$ parameterized with k is equal to $\text{PRINCE}_{core}$ parameterized with $(k \oplus \alpha)$. We call this property of $\text{PRINCE}_{core}$ the *α-reflection property*. It follows that, for any expanded key $(k_0 \,||\, k_0' \,||k_1)$,

$$D_{(k_0||k_0'||k_1)}(\cdot) = E_{(k_0'||k_0||k_1 \oplus \alpha)}(\cdot)$$

where $\alpha$ is the 64-bit constant, $\alpha = c0ac29b7c97c50dd$. Thus, for decryption one only has to do a very cheap change to the master key and afterwards reuse the exact same circuit.

# III. Attacking PRINCE

The differential attack strategy was introduced in 2013 by Ling Song, Lei Hu [2], which is a well-known attack as it states that it breaks the PRINCE cipher in at least eight faults.

## III.1. Fault Model

Although PRINCE may not be implemented in a round-based fashion, we assume an attacker can typically predict when a particular round happens and induce a nibble fault at a specific round. Moreover, the time that certain events take place can often be determined by analyzing a suitable side channel leakage. Furthermore, we assume that an attacker can repeat the experiments with the same plaintext and key without applying external physical effects.

In the remaining part of this thesis, a 16-nibble state $X$ is represented with $(X_0, X_1, \cdots, X_{15})$ and we always denote a right ciphertext by C and its corresponding faulty ciphertext by C* for the same plaintext and key.

Before going to the details of Fault Model, we split the 16 nibbles of the state of PRINCE into four groups numbered from 1 to 4 as depicted in Fig. III.1.

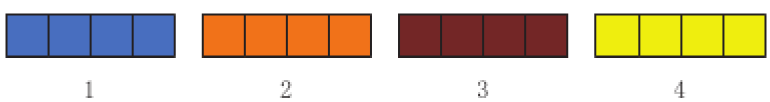



**Fig. III.1.** Splitting nibbles into four groups

The fault model exploits the diffusion property of the diffusion layer $M$ of the cipher.

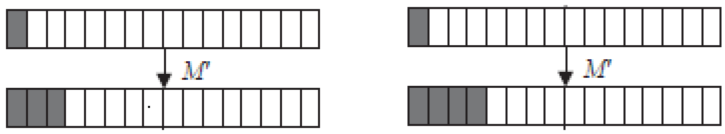
$$M = SR \circ M'$$

**Diffusion property of the M′-mapping:**

Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and corresponding output of the M′-mapping.

First, the $M'$-mapping diffuses the nibbles within groups. If only a certain group of $X$ has nonzero nibbles, then only the same group of $Y$ has nonzero nibbles. Hence the $M'$-mapping of the 64-bit state can be regarded as four small separate mappings $M_1'$, $M_2'$, $M_3'$, and $M_4'$, each of which diffuses the nibbles of the corresponding group.

Second, the $M'$-mapping achieves an almost-MDS property. If $X$ has only one nonzero nibble, say $X_2$ (belongs to Group 1), $Y$ will have at most four nonzero nibbles, all of which are located in the same group (Group 1). Precisely speaking, if the Hamming weight of $X_2$ is greater than 1, then all the four nibbles of Group 1 of $Y$ are nonzero; otherwise, exactly three of them are nonzero as shown in Fig. III.2.




(a) (b)

**Fig. III.2.** (a) and (b) illustrates diffusion property of $M'$ for 1-bit and N>1-bit fault

**Diffusion property of the SR:**

Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and corresponding output of the *SR* operation. If $X$ has a group of four nonzero nibbles, then $Y$ will still have four non-zero nibbles, each of which is located in a different group, i.e., *SR* diffuses the nibbles over groups. $SR^{-1}$ also follows the similar property as shown in Fig. III.3.

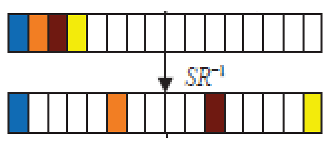



**Fig. III.3.** Diffusion property of $SR^{-1}$

## III.2. Attack Strategy

### Attack at the 11th Round

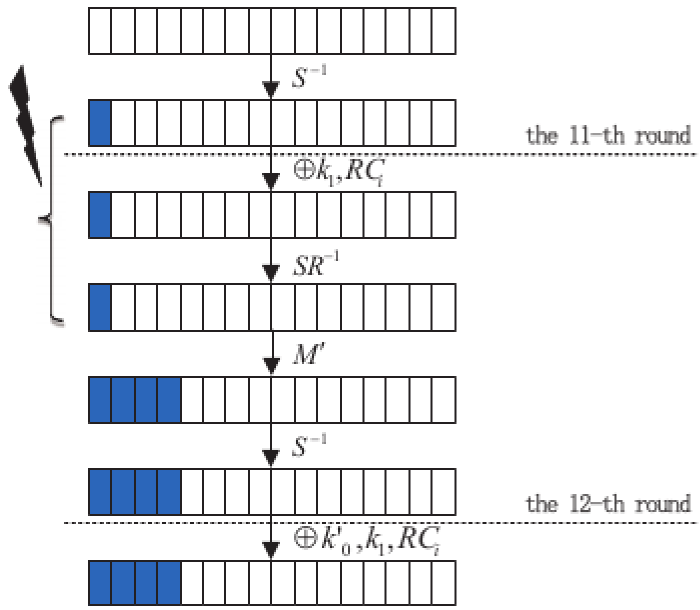



**Fig. III.4.** Attack at the 11th round of PRINCE

First let us consider the scenario when there is a nibble disturbance at the 11th round.

Assume we get a right ciphertext $C$ and its corresponding faulty ciphertext $C^*$ for the same plaintext and key. The fault can happen at any position of the 16 nibbles. For the sake of simplicity, we take the first nibble as a faulty nibble and analysis for other positions are the same.

As illustrated in Fig. III.4., the fault injected in the first nibble during the Sbox substitution of 11th round influences only the first group of the 16 nibbles of the final ciphertext due to the diffusion property of M′-mapping.

In this context, first four nibbles $C_0, C_1, C_2, C_3$ and $C_0^*, C_1^*, C_2^*, C_3^*$ are known, and so is the fact that the bitwise XOR differences of them comes from a single nibble induced by the fault.

Let us look into the first nibble. The $C_0$ and $C_0^*$ are known. Given the input difference of the first Sbox $\Delta_0^{in}$, with the knowledge of the differential distribution table of the $S^{-1}$ of PRINCE (see Table. III.1) in mind, the first nibble of $K = k_0' \oplus k_1$ will be limited to one of 0, 2, or 4 choices by the following equation:

$$S(C_0 \oplus K_0 \oplus (RC_{11})_0) \oplus S(C_0^* \oplus K_0 \oplus (RC_{11})_0) = \Delta_0^{in}$$

To get information about all the first four-nibble of $K = k_0 \oplus k_1$ , we can guess ($\Delta_0^{in}$, $\Delta_1^{in}$, $\Delta_2^{in}$, $\Delta_3^{in}$), the input difference of the first four nibbles of Sbox and then search the subkey information. Before searching, it is necessary to check whether the guesses satisfy the following two conditions which we call the $M'$-Mapping Conditions.

- Nonzero $\Delta_i^{in}$s are valid differences that can lead to the right output differences.
- The preimage of ($\Delta_0^{in}$, $\Delta_1^{in}$, $\Delta_2^{in}$, $\Delta_3^{in}$) under the corresponding submapping of M′-mapping has only one nonzero nibble.

A guess cannot be called a right guess until it passes the $M'$-mapping Conditions. Below a right guess's four-nibble preimage under the corresponding submapping of $M'$-mapping is denoted by P.

After $K = k_0 \oplus k_1$ has been recovered, the last round can be peeled off, and the attack is repeated on the reduced cipher to reveal $k_1$.

| $\Delta in$ | $\Delta out$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 4 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 2 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 2 |
| 3 | 0 | 0 | 4 | 0 | 4 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 4 | 0 | 0 | 0 | 2 |
| 5 | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 0 | 2 | 0 | 4 | 0 | 2 | 0 | 0 | 0 |
| 6 | 0 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 4 | 2 |
| 7 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 2 | 2 |
| 8 | 0 | 4 | 2 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 4 | 0 | 0 |
| A | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 2 |
| B | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 2 | 0 |
| C | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 2 | 0 | 2 | 2 |
| D | 0 | 0 | 4 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 2 | 2 | 0 |
| E | 0 | 0 | 2 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 2 |
| F | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 2 | 2 | 2 |

**Table III.1.** Difference Distribution Table (DDT) of the $S^{-1}$ used by PRINCE

### Attack at the 10th Round

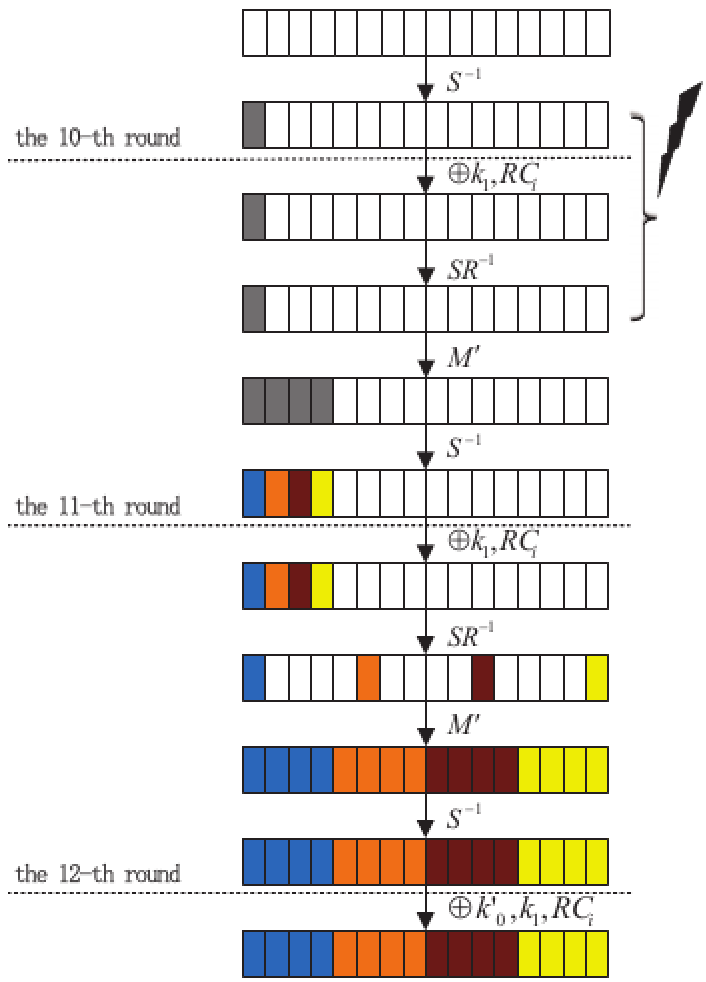



**Fig. III.5.** Attack at the 10th round of PRINCE

In this subsection, fault is induced at the beginning of the 10th round. Set the first nibble to be the faulty nibble again (analysis for other positions remains same), as depicted in Fig. III.5.

Suppose that the induced fault difference has a Hamming weight greater than 1 and the opposite case will be discussed later. As demonstrated by Fig. III.5, the difference remains the same until it goes into the *M′*-mapping of the 11th round. The *M′*-mapping spreads the difference to the whole group, and then the four nibble differences are changed by the $S^{-1}$ (marked with different colors in Fig. III.5). The $SR^{-1}$ of the 12th round splits the four nibble differences into different groups, making each group have one and only one nonzero nibble of difference. After that, *M′*-mapping propagates differences within groups, resulting full difference in the ciphertext.

Given a pair $(C, C^*)$ whose fault difference propagation follows the pattern depicted in Fig. III.5, the analysis is given below.

- For group $i$, $1 \leq i \leq 4$, guess $(\Delta_{4i}^{in}, \Delta_{4i+1}^{in}, \Delta_{4i+2}^{in}, \Delta_{4i+3}^{in})$. For those that satisfy the $M'$-Mapping conditions, store $(P_i, (\Delta_{4i}^{in}, \Delta_{4i+1}^{in}, \Delta_{4i+2}^{in}, \Delta_{4i+3}^{in}))$ in Table $T_i$, where $P_i$ is the four-nibble preimage of the corresponding guess.
- After we get such four tables, search four-nibble subkey values using the items in Table $T_i$ , $1 \leq i \leq 4$ as we did in the previous section.
- Using the $P_i$s in four tables $T_i$, $1 \leq i \leq 4$, check whether the concatenations of $P_1 \,||\, P_2 \,||\, P_3 \,||P_4$ satisfy the *SR* Condition, which is defined as the four nonzero nibbles need to gather together in a single group after the *SR* operation. For those concatenations that pass the *SR* Condition, record $(P_1, P_2, P_3, P_4)$ in table *D*.
- To get candidates of 64-bit key, concatenate the four-nibble subkey values suggested by $(P_1, P_2, P_3, P_4)$, the items of *D*.
- Inject more faults and repeat the previous steps to reduce the space of the 64-bit key.

For the fault difference with Hamming weight equal to 1, less information can be obtained, since the fault difference propagates to only three nibbles within the same group after the $M'$-mapping of 11th round. The following $SR^{-1}$ operation then scatters the three nonzero nibbles into different groups, resulting differences in only three groups of nibbles in the ciphertext.

# IV. <u>Controlling Fault Propagation</u>

As we can observe that the attack exploited the fault propagation scheme of the PRINCE cipher, we can impede the attack by controlling the fault propagation in the cipher.

The following layers of the cipher can be modified to control fault propagation in PRINCE:

- Diffusion layer
- Confusion layer

## IV.1. Linear Diffusion Layer of PRINCE

The linear layer uses a matrix $M$ ($M = SR \circ M'$, $SR$ is shift rows operation) or $M'$. In the linear layer, the 64-bit state is multiplied with $M$ or $M'$, both of which are $64 \times 64$ matrices.

From the construction of the $M'$ matrix, we can observe that each input bit can affect three output bits. All the three affected output bits lie in a different nibble, so one input bit can affect three nibbles. So, we have to make changes in this layer such that the output bit affected by the mounted fault can be decreased. Its simplified representation is shown in Fig. IV.1.

```
def Mcap0(self, data):
    ret = BitArray(length = 16)
    ret[ 0] = data[4] ^ data[ 8] ^ data[12]
    ret[ 1] = data[1] ^ data[ 9] ^ data[13]
    ret[ 2] = data[2] ^ data[ 6] ^ data[14]
    ret[ 3] = data[3] ^ data[ 7] ^ data[11]
    ret[ 4] = data[0] ^ data[ 4] ^ data[ 8]
    ret[ 5] = data[5] ^ data[ 9] ^ data[13]
    ret[ 6] = data[2] ^ data[10] ^ data[14]
    ret[ 7] = data[3] ^ data[ 7] ^ data[15]
    ret[ 8] = data[0] ^ data[ 4] ^ data[12]
    ret[ 9] = data[1] ^ data[ 5] ^ data[ 9]
    ret[10] = data[6] ^ data[10] ^ data[14]
    ret[11] = data[3] ^ data[11] ^ data[15]
    ret[12] = data[0] ^ data[ 8] ^ data[12]
    ret[13] = data[1] ^ data[ 5] ^ data[13]
    ret[14] = data[2] ^ data[ 6] ^ data[10]
    ret[15] = data[7] ^ data[11] ^ data[15]
    return ret
```

**Fig. IV.1.** Simplified representation of $\hat{M}^{(0)}$

## IV.1.A. Attempted Modification to base matrices

To reduce fault propagation, we introduce changes to the base matrices of $M'$, to obtain new diffusion properties. Modified base matrices are shown in Fig. IV.2.

$$M_0 = \begin{matrix} 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{matrix} \qquad M_1 = \begin{matrix} 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{matrix} \qquad M_2 = \begin{matrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \end{matrix} \qquad M_3 = \begin{matrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{matrix}$$

**Fig. IV.2.** Modified base matrices of $M'$

This modification is being suggested by noticing the pattern in the current base matrices of $M'$. In default base matrices, the diagonals have value 1 as a selection of three positions out of four ($^4C_3 = 4$), and the rest have value 0. This pattern contributes to the property shown in Fig. IV.1. So, to reduce the effect on output bits of the diffusion layer, we choose a similar pattern, which is selecting one position out of four ($^4C_1 = 4$) for value 1, and others 0. Now each input bit can affect only one output bit. This makes the $M'$ operation just a bitwise operation on the input bits, as shown in Fig. IV.3.

```
def Mcap0new(self, data):
    ret = BitArray(length = 16)
    ret[ 0] = data[0]
    ret[ 1] = data[5]
    ret[ 2] = data[10]
    ret[ 3] = data[15]
    ret[ 4] = data[12]
    ret[ 5] = data[1]
    ret[ 6] = data[6]
    ret[ 7] = data[11]
    ret[ 8] = data[8]
    ret[ 9] = data[13]
    ret[10] = data[2]
    ret[11] = data[7]
    ret[12] = data[4]
    ret[13] = data[9]
    ret[14] = data[14]
    ret[15] = data[3]
    return ret
```

**Fig. IV.3.** Simplified representation of modified $\hat{M}^{(0)}$

Due to this modification, the differential branch number and linear branch number of the diffusion layer $M$ of the PRINCE decreases.

Branch Number:

The differential branch number measures the diffusion power of a permutation. The differential branch number of a linear diffusion layer *D* is defined as:

$$\beta_d(D) = min_{x!=0} \{wt(x) + wt(Mx)\}$$

The linear branch number measures resistance against linear cryptanalysis. The linear branch number of a linear diffusion layer D is defined as:

$$\beta_l(D) = min_{x!=0} \{wt(x) + wt(M^T x)\}$$

where *wt(x)* is the hamming weight of *x*.

## Results

Using the modified base matrices to construct M′, the current attack strategy on PRINCE fails because the number of residual keys search space increases exponentially.

However, due to the decrease of differential branch number, the number of active boxes decreases, which is not favorable as per the cryptography standards. Also, the reduction in linear branch number decreases the resistance against linear cryptanalysis.

So, we cannot use these modifications as the vulnerability of PRINCE toward cryptanalysis increases.

## IV.1.B. Exhaustive search of base matrices

In this section, we will first describe and mention the different search spaces for the base matrices based on Permutation and Linear equivalence class and also the results of this search.

Permutation Equivalence Class - Let $P_i$ and $P_o$ be two bit permutation matrices. Then the matrix $M'_{new}$ defined by the following transformation,

$$M'_{new} = P_o M' P_i ,$$

belongs to the permutation equivalence set of $M'$, $M'_{new} \in PE(M')$.

Linear Equivalence Class - Let $L_i$ and $L_o$ be two invertible boolean matrices. Then the matrix $M'_{new}$ defined by the following transformation

$$M'_{new} = L_o\, M'\, L_i\,,$$

belongs to the linear equivalence set of $M'$; $M'_{new} \in LE(M')$.

## Results

Permutation Equivalence Class: Search space size ≈ $10^{11}$ cases

- After applying modifications from this class, for each base matrix, there was no change in the diffusion property as compared to original matrices.

Linear Equivalence Class: Search space size ≈ $10^{13}$ cases

- After applying modifications from this class, we found some set of base matrices in which diffusion property was changed. However, the number of XOR operations required to calculate output bits was also increased (many folds), so to maintain the low latency feature of PRINCE cipher, we cannot use those base matrices.

## IV.2. Recursive Diffusion Layer as a Substitute

A diffusion layer $D$ with $s$ words $x_i$ as the input, and $s$ words $y_i$ as the output is called a recursive diffusion layer if it can be represented in the following form:

$$D : \begin{cases} y_0 = x_0 \oplus F_0(x_1, x_2, \ldots, x_{s-1}) \\ y_1 = x_1 \oplus F_1(x_2, x_3, \ldots, x_{s-1}, y_0) \\ \vdots \\ y_{s-1} = x_{s-1} \oplus F_{s-1}(y_0, y_1, \ldots, y_{s-2}) \end{cases}$$

where $F_0, F_1, \ldots, F_{s-1}$ are arbitrary functions, and $\oplus$ is bitwise XOR operation [3].

Recursive diffusion layers with the maximal branch number can be obtained in which $F_i$'s are composed of one or two linear functions and a number of XOR operations.

## Results

To increase resistance of PRINCE to cryptanalysis we choose the functions $(F_0, F_1, \ldots, F_{s-1})$ such that branching number of diffusion layer is maximum.

By keeping maximum differential branching number, we increase the permutation of the input bits which leads to more fault propagation instead of fault masking which is undesirable.

Hence, recursive diffusion layer cannot be used as a substitute for linear diffusion layer in PRINCE.

## IV.3. Confusion Layer of PRINCE

## IV.3.A. PRINCE Sbox

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | B | F | 3 | 2 | A | C | 9 | 1 | 6 | 7 | 8 | 0 | E | 5 | D | 4 |

**Table IV.1.** Default PRINCE Sbox

In order to ensure the security of the resulting design, an Sbox S:$F_2^4 \rightarrow F_2^4$ for the PRINCE-family has to fulfill the following criteria.

1. The maximal probability of a differential is 1/4.
2. There are exactly 15 differentials with probability 1/4.
3. The maximal absolute bias of a linear approximation is 1/4.
4. There are exactly 30 linear approximations with absolute bias 1/4.
5. Each of the 15 non-zero component functions has algebraic degree 3.

## IV.3.B. Different attack scenarios

**Scenario 1:**

While attacking 10th round, if the attacker mounts a 1-bit fault on a nibble in the input, then at the output of the diffusion layer we will get three faulty nibbles of the 1-bit fault and one fault-free nibble in the same group as shown in Fig IV.3.(a) [also, REFER Table IV.2 for more details]. Now, this output becomes input to the confusion layer of 10th round. After applying S-1, the output will have three faulty nibbles, but the count of their bit fault corresponding to each nibble may change. A nibble having a 1-bit fault can be mapped to a nibble having a 3-bit fault at the output or may remain 1-bit fault [REFER Table VI.3 for more details]. For example, in Fig IV.4.(a), only blue nibble remains with 1-bit fault at the output of the confusion layer of the 10th round. Now in the 11th round, this blue nibble on passing through the diffusion layer will output three faulty nibbles and one fault-free nibble. As we already had a fault-free nibble at the input of the 11th round, this nibble will lead to a fault-free group at the end of the 12th round.

In these two ways, we can get fault-free nibbles at the output of the final round. Each fault-free nibble in the faulty ciphertext contributes 24 (nibble size is 4-bit) times the number of predictions in the residual keyspace.

**Scenario 2:**

While attacking the 10th round, if the attacker mounts an *N*>1-bit fault on a nibble, then all the nibbles in that group will be faulty at the output of the diffusion layer as shown in Fig. IV.3.(b) [also, REFER Table IV.2 for more details]. Similar to scenario 1, at the output of the confusion layer in this round, these faulty nibbles can be mapped having a 1-bit fault [REFER Table IV.3 for more details], The mapped 1-bit fault nibble will lead to a fault-free nibble at the end of the 12th round as shown in Fig IV.3.(b).

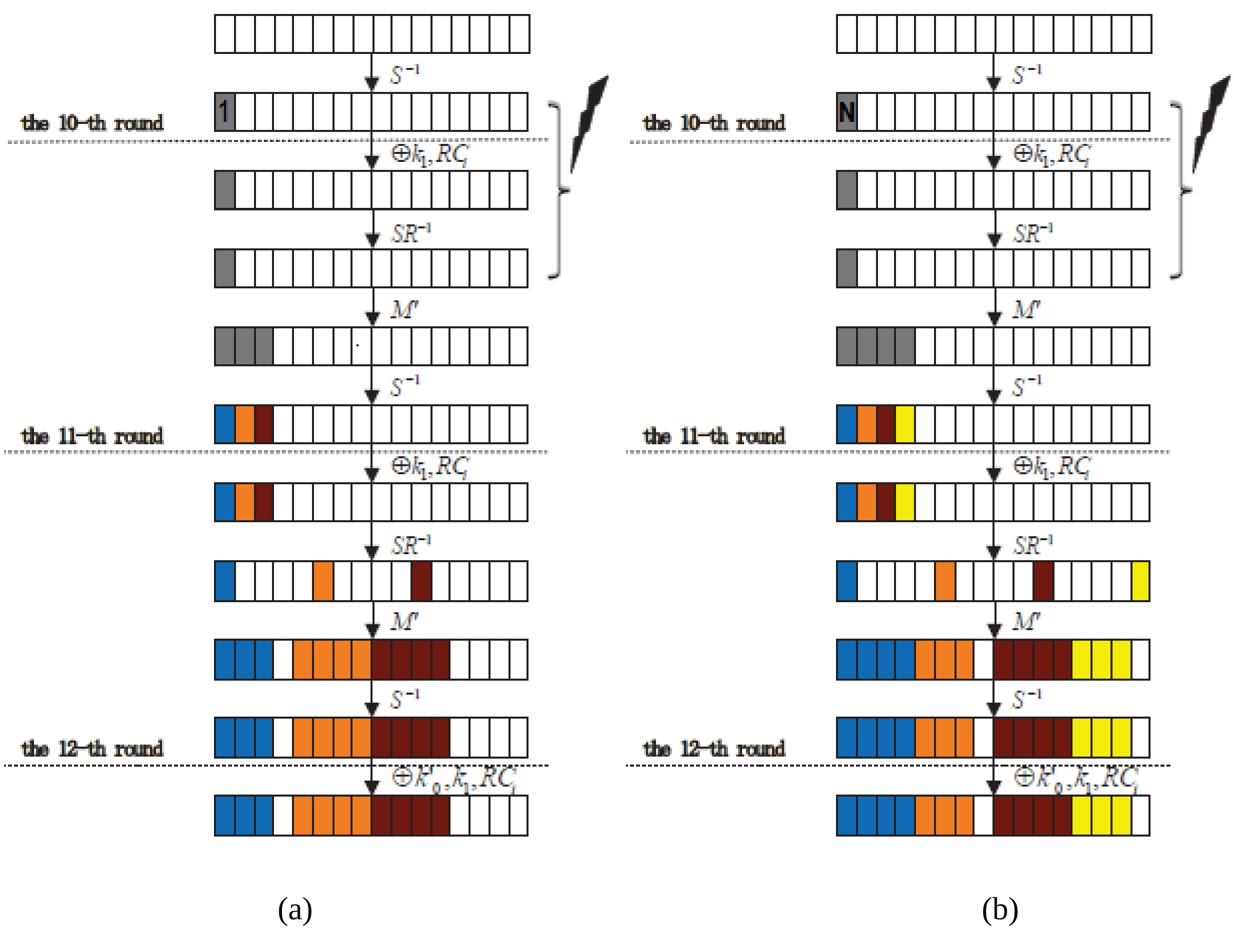



**Fig. IV.4** Fault propagation scenarios: (a)1-bit fault, (b)N>1-bit fault

| Bit fault on a nibble | Output of diffusion layer (faulty nibble's group) |
|---|---|
| 1-bit Fault | 3 Nibbles of 1-bit fault and 1 fault-free nibble |
| 2-bit Fault | 2 Nibbles of 1-bit fault, 2 Nibbles of 2-bit fault |
| 3-bit Fault | 3 Nibbles of 2-bit fault, 1 Nibble of 3-bit fault |
| 4-bit Fault | 4 Nibbles of 3-bit fault |

**Table IV.2**. Diffusion property of $M'$

| Mapping | Cases |
|---|---|
| 1-bit to 1-bit | 30 |
| 2-bit to 1-bit | 20 |
| 3-bit to 1-bit | 14 |
| 4-bit to 1-bit | 0 |

**Table IV.3.** Cases of *N*-bit to 1-bit mapping of fault after applying $S^{-1}$

As the attack is already not feasible for a 1-bit fault (due to vast residual key search space), we find such a Sbox where we increase the chances of an *N*-bit fault ($N$ = 2,3,4) at the input of $S^{-1}$ to map to a 1-bit fault at the output.

## IV.3.C. Proposed Sbox

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S[x] | 9 | d | 3 | 0 | a | 7 | 1 | 6 | e | b | 4 | 5 | 2 | c | f | 8 |

**Table IV.4.** Proposed Sbox for PRINCE

The proposed Sbox satisfies all the property of PRINCE-family Sbox. It also ensures increased chances of mapping *N*-bit fault (*N* = 2,3,4) to 1-bit is increased [REFER to Table IV.4.].

| **Δin\Δout** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 2 |
| 2 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 4 |
| 3 | 0 | 2 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 2 | 0 | 0 | 2 |
| 4 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 |
| 5 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 2 | 2 | 0 |
| 6 | 0 | 0 | 0 | 2 | 2 | 0 | 2 | 2 | 2 | 4 | 0 | 0 | 0 | 0 | 2 | 0 |
| 7 | 0 | 4 | 0 | 0 | 2 | 2 | 2 | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 4 | 2 | 0 | 0 | 2 |
| 9 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 2 | 0 | 0 |
| a | 0 | 0 | 4 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 2 |
| b | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 0 | 0 |
| c | 0 | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 4 | 0 | 2 | 0 |
| d | 0 | 2 | 4 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 2 | 0 |
| e | 0 | 2 | 2 | 2 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 2 | 0 | 0 | 2 | 0 |
| f | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 4 | 2 | 4 |

**Table IV.5.** Difference Distribution Table (DDT) of the proposed $S^{-1}$

| **Mapping** | **Cases** |
|---|---|
| 1-bit to 1-bit | 12 |
| 2-bit to 1-bit | 20 |
| 3-bit to 1-bit | 32 |
| 4-bit to 1-bit | 0 |

**Table IV.6.** Cases of *N*-bit to 1-bit mapping of fault after applying $S^{-1}$

## IV.3.D. Finding proposed Sbox

We used strict hill climbing approach to find the proposed Sbox. We have first defined a desired $S^{-1}$ distribution table (following the constraints maintained below) as the destination for the strict hill climbing.

$$wt(S[x] \oplus S[x \oplus \alpha]) = 1, wt(\alpha) = 3$$

$wt(x)$ denotes hamming weight of $x$, $S$ represents the Sbox and $\oplus$ denotes XOR operation.

There are 4 values of 1-bit fault (0001,0010,0100,1000) and 4 values of 3-bit fault (0111,1011,1101,1110) in 4-bit word, so we uniformly distributed all 64 cases in the DDT, mapping all 3-bit fault to 1-bit [REFER to Table IV.7] and then filled the remaining places randomly keeping properties of DDT in mind [REFER to Table IV.8].

| **Δin\Δout** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **a** | **b** | **c** | **d** | **e** | **f** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **2** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **3** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **4** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **5** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **6** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **7** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **9** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **a** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **b** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **c** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |
| **d** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **e** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **f** | 0 | 0 | 0 | _ | 0 | _ | _ | _ | 0 | _ | _ | _ | _ | _ | _ | _ |

**Table IV.7**. Ideal $S^{-1}$ Difference distribution table

| Δin\Δout | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 2 | 2 | 2 | 2 | 2 |
| **2** | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 4 | 0 | 0 | 0 | 2 | 2 | 0 | 2 | 0 |
| **3** | 0 | 0 | 0 | 2 | 0 | 4 | 4 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 4 | 0 | 0 | 0 | 2 | 2 | 4 |
| **5** | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 4 | 4 | 2 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | 2 | 0 | 4 | 2 |
| **7** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 2 | 4 | 2 | 0 | 2 |
| **9** | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 4 | 0 | 0 |
| **a** | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 2 |
| **b** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **c** | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 4 | 0 | 2 | 2 | 0 | 2 | 2 |
| **d** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **e** | 0 | 4 | 4 | 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **f** | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 2 | 2 | 2 |

**Table IV.8.** Randomly Selected $S^{-1}$ Difference distribution table

So the Table IV.8. becomes the destination $S^{-1}$ table for the hill climbing approach and we our starting point is the distribution table of the Sbox given in Table IV.9.

| **x** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S[x]** | 5 | D | 7 | F | 1 | 9 | 3 | B | 4 | C | 6 | E | 0 | 8 | 2 | A |

**Table IV.9.** Starting Sbox for hill climbing

## Steps to follow

1. Randomly select two values of *S[x]* and swap them.
2. For this new Sbox generate $S^{-1}$ distribution table.
3. Calculate the euclidean distance between the destination and current distribution table.
4. If current distance is less than the previous distance (before swap) :we accept this swap
    a. else revert it.
5. If destination is achieved stop
    a. else goto step 1.

It is not always possible to reach the destination table (as it is randomly selected), we use intermediate results which are most favorable in the scenario.

## Results

Due to the increase in the number of cases where a 3-bit fault nibble is mapped to 1-bit, residual outer key search space for proposed Sbox increases, when fault induced is 3,4-bit, as shown in Table IV.10. On the other hand, residual outer key search space for proposed Sbox decreases when fault induced is 1,2-bit.

The Average residual outer key search space, shown in Table IV.10, is for 5000 iterations having random key and random plain text with fault induced in 1 nibble.

| Induced bit fault (on the input of $10^{th}$ round ) | Residual Outer Key Search Space (#fault = 1) | |
|---|---|---|
| | Default | Proposed |
| 1-bit | $2^{39.1286349468}$ | $2^{34.902531887/8}$ |
| 2-bit | $2^{28.70839134}$ | $2^{25.7504353339}$ |
| 3-bit | $2^{26.0810274461}$ | $2^{26.8074160974}$ |
| 4-bit | $2^{25.5043490616}$ | $2^{28.1624849227}$ |

**Table IV.10.** Residual outer key search space of default and proposed Sbox

## V. Conclusion

Proposed S-box increases the security of PRINCE cipher against higher bit faults (3-bit and 4-bit), which are easier to induce as compared to lower bit faults due to the requirement of high precision equipment.

If the attacker wants to induce fault on the input of 11th round, to breach the proposed security, then the effort of inducing fault becomes 4 times higher as compared to10th round fault mounting.

# VII. References


1. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın : PRINCE – A Low-latency Block Cipher for Pervasive Computing Applications, ASIACRYPT 2012: Advances in Cryptology – ASIACRYPT 2012, pp. 208-225, Springer, Heidelberg (2012)

2. Ling Song, Lei Hu : Differential Fault Attack on the PRINCE Block Cipher, LightSec 2013: Lightweight Cryptography for Security and Privacy, pp. 43-54, Springer, Heidelberg (2013)

3. Mahdi Sajadieh, Mohammad Dakhilalian,Hamid Mala, and Pouyan Sepehrdad : Recursive Diffusion Layers for Block Ciphers and Hash Functions, SAC 2012: Selected Areas in Cryptography, pp. 355-371, Springer, Heidelberg (2012)

4. Sumanta Sarkar, Habeeb Syed : Bounds on Differential and Linear Branch Number of Permutations, ACISP 2018: Information Security and Privacy, pp. 207-224, Springer, Heidelberg (2018)

5. Markku-Juhani O. Saarinen : Cryptographic Analysis of All 4 × 4-Bit S-Boxes, SAC 2011: Selected Areas in Cryptography, pp. 118-133, Springer, Heidelberg (2011)

6. Kishan Gupta, Indranil Ray : On Constructions of MDS Matrices from Companion Matrices for Lightweight Cryptography, CD-ARES 2013: Security Engineering and Intelligence Informatics, pp. 29-43, Springer, Heidelberg (2013)

7. Robert S, Ledley : THE INVERSE OF A BOOLEAN MATRIX, Published 1965 DOI:10.21236/ad0724782

8. Boneh, D., DeMillo, R. A., Lipton, R. J.: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37-51. Springer, Heidelberg (1997)

9. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513-525. Springer, Heidelberg (1997)

10. Bar-el, H., Choukri, H., Naccache, D., Tunstal, M., Whelan, C.: The Sorcerers Apprentice Guide to Fault Attacks. Proceedings of the IEEE, 94(2), 370-382 (2006)