Turk. J. Math. Comput. Sci.
11(2)(2019) 58–73
© MatDer
http://dergipark.gov.tr/tjmcs
http://tjmcs.matder.org.tr

MATDER

FeW: A Lightweight Block Cipher

Manoj Kumar[1,*], SK Pal[1], Anupama Panigrahi[2]

[1]*Scientific Analysis Group, DRDO, Metcalfe House Complex, Delhi-110 054, INDIA.*
[2]*Department of Mathematics, University of Delhi, Delhi-110 007, INDIA.*

Received: 23-08-2018 • Accepted: 04-04-2019

Abstract. In this paper, we propose a new lightweight block cipher *FeW* which encrypts plaintext in the blocks of 64-bit using 80/128 bits key to produce 64-bit ciphertext. We also propose a new structure namely *Feistel-M structure* by admixture of Feistel and 4-branch generalised Feistel structures. This new structure significantly contributes to enhance the security margins of our design against the basic cryptanalytic attacks like differential, linear and impossible differential attacks. Security analysis signifies that *FeW* has enough security margins against these cryptanalytic attacks and it can resist any key recovery attack beyond 17 rounds with the complexity better than 2^{64}.

2010 AMS Classification: 94A60, 11T71, 14G50.

Keywords: Feistel structure, generalised Feistel structure, lightweight cryptography, substitution permutation network.

1. Introduction

Lightweight cryptography [24, 32] has emerged to an immense research area in the field of cryptography. Researchers espied to this area around the beginning of 21st century to cater the demand of cryptographic algorithms having less implementation area and energy. Due to advancements in technology, industry started using cryptography for securing daily life products and there were specific demands from product developers to provide security features in tiny and hand-held devices viz. unlocking devices of cars. This was the time when Rijndael [12] was selected as AES and there was a scope for lightweight ciphers for specific applications like RFID tags and sensor networks. Lightweight block ciphers like PRESENT [8], RECTANGLE [46], PICARO [31], HIGHT [19], LBlock [45], TWINE [41], SIMON and SPECK family [1], TEA [44], DES Light weight variant [27] and other lightweight designs [10, 17, 18, 23, 36, 37, 39] with different design constructions [25] have been published in the last 15 years. International Organisation for Standardization and International Electrotechnical Commission has specified PRESENT and CLEFIA suitable for Lightweight Cryptography (ISO/IEC 29192-2:2012). Popularity of PRESENT inspired cryptographic community to design many more lightweight block ciphers. Majority of designs are proposed for hardware oriented applications and they do not perform equally well in software environments. There is an emerging demand of security in critical situations like wireless sensor networks and cloud computing which require software oriented lightweight block ciphers [28]. We propose a secure and efficient lightweight block cipher suitable for these kinds of environments.

*Corresponding Author
Email addresses: manojkumar@sag.drdo.in (M. Kumar), skpal@hqr.drdo.in (SK. Pal), anupama.panigrahi@gmail.com (A. Panigrahi)

In comparison to SPN [35] structure, Feistel structure [38] seems a better choice for designing a new lightweight cipher as it does not require inversion of the round function and S-box. Feistel based cipher runs with fewer computations, since half of the input block is processed through round function in each round. While, SPN based cipher processes the full input block through round function in each round. Therefore, our choice is Feistel structure with SPN type round function to design an efficient lightweight block cipher. Some earlier designs have also used similar kind of operations in their round function as we have used in *FeW*. For instance, SMS4 [13] block cipher consists of cyclic shifts and XOR operations on 32-bit words and generalised Feistel based design CLEFIA [37] uses two different round functions. We use two different functions inside the round function. In addition, we also use a function to swap the least significant byte between two 16-bit Feistel branches and two different combinations of cyclic shift and XOR operations.

The rest part of the paper is organized in the following manner. In section 2, we describe design of *FeW* in detail. In section 3, we present the security evaluation against some basic cryptanalytic attacks. Finally, its performance is compared with other designs in section 4.

Notations:

- P_m, C_m	:64-bit input plaintext and ciphertext blocks
- MK-80	:80-bit user supplied key
- MK-128	:128-bit user supplied key
- RK_i	:16-bit subkey extracted from MK
- K_i	:32-bit subkey for round i as $RK_{2*i} \| RK_{2*i+1}$ (for i=0 to 31)
- rF	:Round function
- WF_1	:Weight Function 1 used inside rF
- WF_2	:Weight Function 2 used inside rF
- $\oplus$	:Bitwise exclusive-OR operation
- $\lll n$	:Left cyclic shift by n bits
- $\ggg n$	:Right cyclic shift by n bits
- $[i]_2$	:Binary representation of integer i
- $\|$	:Concatenation of two n-bit strings
- &	:Bitwise And between two n-bit strings
- $B \leftarrow A$	:A is transformed to B

2. Lightweight Block Cipher: *FeW*

We describe detailed design specifications; encryption algorithm and key schedule of lightweight block cipher *FeW* in this section:

2.1. **Feistel-M Structure.** Feistel Structure was proposed by Horst Feistel. It divides the input block in two equal parts: First half (FH) and Second half (SH). The round function is applied on FH and its output is XORed with SH to get new FH for the next round and old FH becomes new SH for next round. Generalized Feistel structure processes the input block by dividing it into N equal parts. If we fix the value of N equals to 4 then it is referred as 4-branch generalised Feistel structure.

We devise a novel mixing approach between the two Feistel branches of 4-branch generalised Feistel structure and name it as *Feistel-M structure* due to the extra mixing operation used. Mixing operation is carried out inside the round function rF by swapping the least significant byte between the two 16-bit branches. We use this structure to design the lightweight block cipher *FeW*. This admixture enhanced the security margins of our design against the basic cryptanalytic attacks. We use two different combinations of cyclic shift and XOR operations on 16-bit words inside the round function. Application of cyclic shift and XOR operations provides better software efficiency.

2.2. **Design Specifications.** *FeW* takes 64-bit plaintext as input and generates the same size of ciphertext using 80/128 bits key. It consists of total 32 rounds. Design of *FeW* is based on *Feistel-M structure* which is shown in Fig. 1. Its round function processes 32-bit word as two 16-bit branches with an extra mixing operation between the two branches. The round function rF comprises of two different functions WF_1 and WF_2 which are applied on two 16-bit branches respectively. However, one can visualize it broadly as a balanced Feistel based design also.

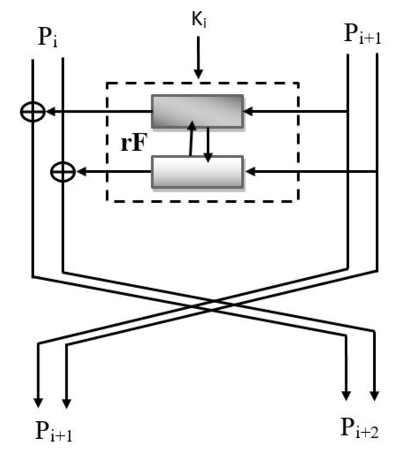

FIGURE 1. one round *FeW*

2.3. **Encryption Algorithm.** First, 64-bit plaintext P_m is divided into two 32-bit halves namely left half P_0 and right half P_1. Plaintext P_m is expressed as a concatenation of two 32-bit words P_0 and P_1 :

$$(P_0 \parallel P_1) \leftarrow P_m$$

Then, following steps are applied to obtain the 64-bit ciphertext C_m:

a) Apply rF on P_{i+1} & K_i and XOR it with P_i to produce P_{i+2} (i=0 to 31): $P_{i+2} \leftarrow (P_i \oplus rF(P_{i+1}, K_i))$
b) Apply swap function on the output of last round: $(C_0 = P_{33}, C_1 = P_{32}) \leftarrow (P_{32}, P_{33})$

Finally, we obtain the 64-bit ciphertext C_m as a concatenation of two 32-bit words C_0 and C_1 as follows:

$$C_m \leftarrow (C_0 \parallel C_1)$$

2.4. **Round Function** rF**.** It takes two inputs: $X_i(= P_{i+1})$ & key K_i and returns Y_i as output (Fig. 2):

$$rF : \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$$

We use two different weight functions WF_1 and WF_2 inside rF, which are described below in detail. Each function takes 16-bit as input and produces 16-bit as output. In each round, rF is applied on X_i and K_i as follows:

i. $(X_i \oplus K_i) \leftarrow (X_i, K_i)$
ii. $(C_{(8)} \parallel D_{(8)} \| E_{(8)} \parallel F_{(8)}) \leftarrow (X_i \oplus K_i)$
iii. $A_{(16)} \parallel B_{(16)}$, where $A_{(16)} = (C_{(8)} \parallel F_{(8)})$ and $B_{(16)} = (E_{(8)} \parallel D_{(8)})$

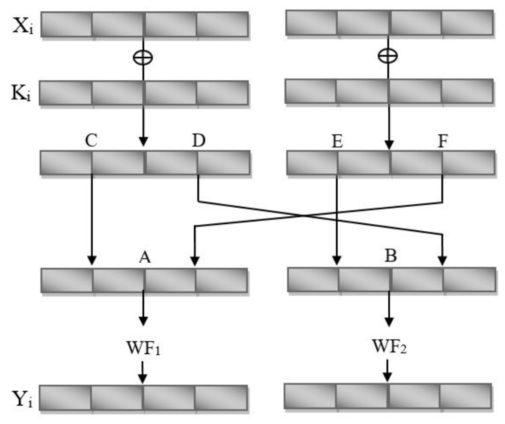

FIGURE 2. Round Function

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	2	E	F	5	C	1	9	A	B	4	6	8	0	7	3	D

Table 1. S-Box

A & B are processed through WF_1 and WF_2 respectively. Finally, Y_i is the concatenation of two 16-bit outputs G & H from WF_1 and WF_2, respectively. Y_i is XORed with 32-bit word P_i to produce P_{i+2} as follows:

$$P_{i+2} \leftarrow P_i \oplus rF(X_i, K_i) \text{ , where } X_i = P_{i+1}$$

$$i.e.\ P_{i+2} \leftarrow P_i \oplus Y_i \text{ , where } Y_i = rF(X_i, K_i)$$

2.4.1. *Weight Function WF_1.* It processes 16-bit input A to produce 16-bit output G as follows:
$WF_1 : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ i.e. $G \leftarrow WF_1(A)$, where $A = (A_0 \parallel A_1 \parallel A_2 \parallel A_3)$
WF_1 applies S-box (S) 4 times in parallel as non-linear operation and produces U. Thereafter, it applies a combination of cyclic shifts and XOR operations on U as linear mixing operation L_1:

(1) $U \leftarrow (U_0 \parallel U_1 \parallel U_2 \parallel U_3)$ where $U_0 \leftarrow S(A_0); U_1 \leftarrow S(A_1); U_2 \leftarrow S(A_2); U_3 \leftarrow S(A_3)$
(2) $G \leftarrow L_1(U)$ where $L_1(U) = U \oplus U \lll 1 \oplus U \lll 5 \oplus U \lll 9 \oplus U \lll 12$

2.4.2. *Weight Function WF_2.* It takes 16-bit input B to produce 16-bit output H as follows:
$WF_2 : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ i.e. $H \leftarrow WF_2(B)$, where $B = (B_0 \parallel B_1 \parallel B_2 \parallel B_3)$
WF_2 applies S-box (S) 4 times in parallel as non-linear operation and produces V. Thereafter, it applies a different combination of cyclic shifts and XOR operations on V as linear mixing operation L_2:

(1) $V \leftarrow (V_0 \parallel V_1 \parallel V_2 \parallel V_3)$ where $V_0 \leftarrow S(B_0); V_1 \leftarrow S(B_1); V_2 \leftarrow S(B_2); V_3 \leftarrow S(B_3)$
(2) $H \leftarrow L_2(V)$ where $L_2(V) = V \oplus V \lll 4 \oplus V \lll 7 \oplus V \lll 11 \oplus V \lll 15$

2.5. **S-Box.** There is an application of single 4-bit S-box (Table 1) in encryption algorithm, decryption process and key expansion of *FeW*-80 and *FeW*-128. This was also used in the design of block cipher HummingBird2 [15]. It is categorized as golden S-box in the cryptographic analysis [11] of all 4×4 S-boxes [34].

2.6. **Key Expansion Algorithm.** There is a k-bit user supplied key which is expanded to obtain the round subkeys for each round. Key expansion algorithm is the only difference between the two versions of lightweight block cipher *FeW* i.e. *FeW*-80 and *FeW*-128.

2.6.1. *Key Schedule for MK-80.* We store user supplied 80-bit key MK-80 in a key register MK as:

$$MK = k_0k_1k_2k_3...........k_{78}k_{79}.$$

We obtain round subkey RK_0 by extracting leftmost 16 bits from the current contents of MK and update MK in the following steps (for i =1 to 63):

(1) $MK \lll 13$
(2) $[k_0k_1k_2k_3] \leftarrow S[k_0k_1k_2k_3]$;
$[k_{64}k_{65}k_{66}k_{67}] \leftarrow S[k_{64}k_{65}k_{66}k_{67}]$;
$[k_{76}k_{77}k_{78}k_{79}] \leftarrow S[k_{76}k_{77}k_{78}k_{79}]$
(3) $[k_{68}k_{69}k_{70}k_{71}k_{72}k_{73}k_{74}k_{75}] \leftarrow [k_{68}k_{69}k_{70}k_{71}k_{72}k_{73}k_{74}k_{75}] \oplus [i]_2$
(4) Leftmost 16 bits from current contents of MK are stored as round subkey RK_i

2.6.2. *Key Schedule for MK-128.* First, we store user supplied 128-bit key MK-128 in a key register called MK as:

$$MK = k_0k_1k_2k_3 \dots k_{127}.$$

We extract the round subkey RK_0 as leftmost 16 bits from current contents of MK and update the register MK in the following steps (for i = 1 to 63):

(1) $MK \lll 13$
(2) $[k_0k_1k_2k_3] \leftarrow S[k_0k_1k_2k_3]$
$[k_4k_5k_6k_7] \leftarrow S[k_4k_5k_6k_7]$
$[k_{112}k_{113}k_{114}k_{115}] \leftarrow S[k_{112}k_{113}k_{114}k_{115}]$
$[k_{124}k_{125}k_{126}k_{127}] \leftarrow S[k_{124}k_{125}k_{126}k_{127}]$

(3) $[k_{116}k_{117}k_{118}k_{119}k_{120}k_{121}k_{122}k_{123}] \leftarrow [k_{116}k_{117}k_{118}k_{119}k_{120}k_{121}k_{122}k_{123}] \oplus [i]_2$
(4) Leftmost 16 bits from current contents of MK are stored as round subkey RK_i.

2.7. **Decryption Algorithm.** *FeW* is a balance *Feistel-M* based design, therefore its decryption algorithm does not require inversion of the round function. Round subkeys are used in the reverse order which is the main difference between the encryption and decryption algorithms.

First, we divide the 64-bit ciphertext C_m into two halves C_0 and C_1. We have 64-bit input ciphertext C_m as a concatenation of two 32-bit words:

$$(C_0 \parallel C_1) \leftarrow C_m$$

(1) Apply rF on C_{i+1} & K_{31-i} and XOR it with C_i to give C_{i+2} ($i = 0$ to 31) :

$$C_{i+2} \leftarrow C_i \oplus rF(C_{i+1} \oplus K_{31-i})$$

(2) Finally, swap the output in last round: $(P_0 = C_{33}, P_1 = C_{32}) \leftarrow (C_{32}, C_{33})$

We obtain the 64-bit plaintext as: $P_m \leftarrow (P_0 \parallel P_1)$.

3. Security Analysis

Block Ciphers [9,24] are among the most analysed cryptographic primitives. A large variety of cryptanalytic attacks can be applied on block ciphers. We provide security estimates of our design against some basic cryptanalytic attacks in this section.

3.1. **Differential Cryptanalysis.** Differential attack [4] is one of the most basic cryptanalytic attack applied on block ciphers which exploits the occurrences of high probability differences in the input and output of a cipher. Linear components of a cipher produces the output differences with probability 1, while this is not the case for non-linear components (e.g. S-box). Therefore, we analyse the non-linear components of a cipher and use the probabilistic relations between the input and output differences of these components (S-box). We join several high probability one round differential trails to form high probability (equals to p) differential trails for r rounds. These differentials are used to recover the round subkeys from outermost rounds. For all possible input differences to the S-box, we find the distribution of output differences and collect these in the form of a difference distribution table (DDT).

3.1.1. *Difference Distribution Table:* Output difference from the S-box is denoted by $\triangle V$ and input difference is represented by $\triangle U$. We count the number of occurrences of each output difference ($0 \leq \triangle V \leq 15$) corresponding to a particular input difference $\triangle U$. This exercise is repeated for all possible input differences ($0 \leq \triangle U \leq 15$) to get the 16×16 DDT. We run algorithm 1 in order to get table 2.

```
**For:** $\triangle U$=0 to 15
**For:** $\triangle V$=0 to 15
**For:** X = 0 to 15
    If $S[\triangle U \oplus X] \oplus S[X] = \triangle V$
    Then $DDT[\triangle U][\triangle V] + +$
**Output:** $DDT[\triangle U][\triangle V]$
```

Algorithm 1. Difference Distribution Table

The maximum differential probability for a non-zero input difference producing an output difference in a single S-box application of *FeW* is $4/16 = 2^{-2}$ (Table 2). We need approximately p^{-1} chosen plaintext pairs to mount the differential attack on a block cipher, where p is the probability of the differential trail [7]. Assuming that an attacker finds a differential trail with one active S-box (i.e. S-box with non-zero input difference value) in each round, so the number of active S-boxes in full round differential trail will be 32 . Therefore, the probability of 32-round trail will become $(2^{-2})^{32}$ i.e. 2^{-64}. So, we need 2^{64} chosen plaintext pairs for a full round differential distinguisher. This condition ensures that even if there is only one active S-box in each round, still we will require 2^{64} chosen plaintexts (full codebook) to distinguish *FeW* from random permutation using differential attack.

Branch number of a function is defined by Rijmen [33] and Kanda [20] for SPN based designs and Feistel based designs with SPN type round function. We define the branch number of linear permutation layer and find out the

OD →																
ID ↓	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	2	0	0	0	2	0	0	2	0	2	2	4	2
2	0	0	0	2	0	2	0	0	0	0	2	4	2	4	0	0
3	0	2	2	2	2	0	2	2	2	0	0	0	0	2	0	0
4	0	0	0	2	0	4	2	0	0	0	0	2	0	0	2	4
5	0	0	2	2	2	2	0	0	0	0	0	4	4	0	0	0
6	0	0	0	2	4	0	2	0	2	2	0	2	0	0	0	2
7	0	2	0	0	0	0	2	4	4	2	0	0	0	0	2	0
8	0	0	0	0	0	0	2	2	0	4	4	0	2	2	0	0
9	0	2	0	2	2	2	2	2	0	2	0	2	0	0	0	0
A	0	2	0	0	4	0	2	0	0	2	0	0	2	2	0	2
B	0	2	2	0	0	0	0	0	2	0	4	2	0	0	4	0
C	0	0	4	0	0	2	0	2	2	2	0	0	2	0	0	2
D	0	2	2	0	0	2	2	0	2	0	2	0	2	0	2	0
E	0	2	4	2	0	0	0	0	0	2	2	0	0	0	2	2
F	0	2	0	0	2	2	0	2	2	0	0	0	0	4	0	2

Table 2. Difference Distribution Table

differential & linear branch number for these layers below. We use similar techniques as in [22] and [40] to show the resistance of *FeW* to differential and linear attacks.

Definition 3.1. Let $X = (x_0 \parallel x_1 \parallel x_2 \parallel x_3)$ be a $4n$-bit word and size of each x_i is n-bit. We count the number of $i^{'}s$ such that $x_i (i = 0 \ to \ 3)$ is not equal to zero. We denote this count as the Hamming weight of X i.e. Hw(X).

Definition 3.2 (Branch Number)**.** Let X be a 16-bit input to the function f where X is written as concatenation of 4-bit nibbles x_0, x_1, x_2 and x_3. We define the branch number of the function $f : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ by $\beta(f)$ as follows:

$$\beta(f) = \min_{(X \neq 0, X \in \{0,1\}^{16})} (\text{Hw(X)} + \text{Hw(f(X))}).$$

Definition 3.3. Differential Branch number β_d of a linear layer L is defined as:

$$\beta_d(L) = \min_{(\triangle X \neq 0, X_1, X_2 \in \{0,1\}^{16})} (\text{Hw}\triangle(X) + \text{Hw}(L(\triangle X)))$$

where $\triangle X = X_1 \oplus X_2$ is input difference to the linear layer L and $L(\triangle X) = L(X_1) \oplus L(X_2)$ is output difference. In case of *FeW*, differential branch number of the linear layers L_1 and L_2 is 5, which is explained in section 3.1.2.

Definition 3.4. Let $S : U \rightarrow V$ is a bijective function where U and V are 4-bit nibbles. We observe the difference distribution table (DDT) of the S-box (Table 2) which reflects the number of occurrences for all possible input ($\triangle U$) and output ($\triangle V$) differences. If the input to the DDT of S-box is a non-zero nibble then it certainly outputs a non-zero nibble which implies that:

$$\text{Hw}(\triangle U) = \text{Hw}(\triangle V).$$

Result 1: We observe the DDT of S-box and find that any two inputs to S-box with non-zero difference certainly produce two outputs with non-zero difference. Therefore, number of non-zero nibbles in the input to round function is same as the number of non-zero nibbles in the input to linear layer.

3.1.2. *Differential Branch Number of FeW.* If a function takes n-branch input and returns n-branch output, then maximum value of the branch number for this function can be $n + 1$. Diffusion layers with maximum value of branch number provides optimal security against differential and linear attacks. In case of *FeW*, linear layers L_1 and L_2 are applied on 16-bit branches. We tested the combination of cyclic shifts and XOR for all possible values of cyclic shifts and all possible 16-bit non-zero inputs using algorithm 2.

There are four different values of (i, j, k, l) which give the maximum value of the branch number. These four values are: (1,5,9,12), (3,7,11,12), (4,5,9,13) and (4,7,11,15). We have used two values out of these four values as our linear layers L_1 and L_2.

```
For: $i = 1$ to 15
For: $j = i + 1$ to 15
For: $k = j + 1$ to 15
For: $l = k + 1$ to 15
For: $\triangle X = 1$ to $2^{16} - 1$
    $L(\triangle X) = \triangle X \oplus (\triangle X \lll i) \oplus (\triangle X \lll j) \oplus (\triangle X \lll k) \oplus (\triangle X \lll l)$;
    $\beta_p(L) = \mathrm{Hw}(\triangle X) + \mathrm{Hw}(L(\triangle X)$;
If $\beta_p(L) \geq 5$, for all $\triangle X$;
Return: $\beta_d(L) = 5$ and (i, j, k, l)
```

Algorithm 2. Differential Branch Number

Definition 3.5 (Active S-box)**.** An S-box corresponding to a non-zero input difference is called an active S-box.

Definition 3.6 (Active Round Function)**.** We term a round function as active if it contains at least one active S-box. In other words, a round function which contributes to a differential trail is an active round function otherwise we call it a passive round function.

Definition 3.7 (Inverse of Linear Layer)**.** For a 16-bit input W, we define the inverse of a linear function L by L^{-1} such that $L^{-1}(L(W)) = W$. We use two different combinations L_1 and L_2 of cyclic shifts and XOR operations as linear layers and their inverse is defined as follows:

$$\begin{aligned} L_1^{-1}(W) &= W \oplus W \ggg 1 \oplus W \ggg 4 \oplus W \ggg 5 \oplus W \ggg 7 \oplus W \ggg 8 \oplus W \ggg 9 \\ &\quad \oplus W \ggg 10 \oplus W \ggg 12 \oplus W \ggg 14 \oplus W \ggg 15 \\ L_2^{-1}(W) &= W \oplus W \ggg 1 \oplus W \ggg 2 \oplus W \ggg 4 \oplus W \ggg 6 \oplus W \ggg 7 \oplus W \ggg 8 \\ &\quad \oplus W \ggg 9 \oplus W \ggg 11 \oplus W \ggg 12 \oplus W \ggg 15 \end{aligned}$$

Theorem 3.8. *Let $(\triangle P_i \parallel \triangle P_{i+1})$ be the 64-bit input difference to the i^{th} round where $\triangle X_i (= \triangle P_{i+1})$ is the 32-bit input difference and $\triangle Y_i$ is the 32-bit output difference to the round function rF. We obtain the following relationship between the input and output differences for any three consecutive rounds i^{th}, $(i+1)^{th}$ and $(i+2)^{th}$ of FeW:*
$\triangle X_i \oplus \triangle X_{i+2} = \triangle Y_{i+1}$

Proof. Follows trivially by observing Fig. 3. □

Theorem 3.9. *If $\triangle X_i \oplus \triangle X_{i+2}$ is not equal to zero, then any three consecutive rounds of FeW have at least 5 active S-boxes.*

Proof. We denote the linear transformation layers (L_1 and L_2) in round function rF by L. We use Theorem 3.8 and Definition 3.7 with linearity of L to get the following relation:

$$\triangle X_i \oplus \triangle X_{i+2} = \triangle Y_{i+1} = L(L^{-1}(\triangle Y_{i+1}) = L(\triangle L^{-1}(Y_{i+1})).$$

Applying inverse of L on the output to round function at i^{th} round, we get the same number of non-zero nibbles as there are in the input to round function. Therefore, we have the following relation using Definition 3.4 and Result 1:

$$\mathrm{Hw}(\triangle X_{i+1}) = \mathrm{Hw}(\triangle L^{-1}(Y_{i+1})).$$

We know the relation between the number of non-zero nibbles in two binary strings α and β [20]:

$$\mathrm{Hw}(\alpha) + \mathrm{Hw}(\beta) \geq \mathrm{Hw}(\alpha \oplus \beta).$$

Using this relation, we get:

$$\mathrm{Hw}(\triangle X_i) + \mathrm{Hw}(\triangle X_{i+2}) \geq \mathrm{Hw}(\triangle X_i \oplus \triangle X_{i+2}).$$

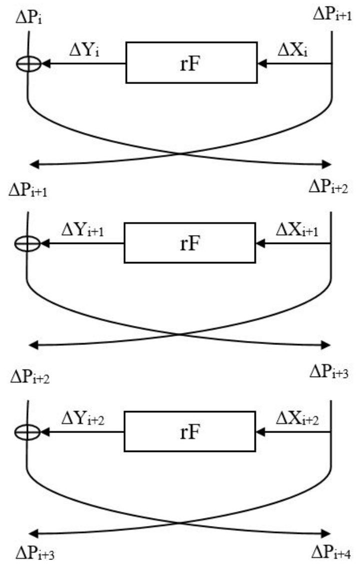

Figure 3. Differential Trail

Using equations (2.1), (2.2), (2.3) & definition of $\beta_d(L)$, we get the following relation which asserts that any 3 consecutive rounds will have at least 5 differentially active S-boxes if $\triangle X_i \oplus \triangle X_{i+2} \neq 0$

$$\begin{aligned}
\text{Hw}(\triangle X_i) + \text{Hw}(\triangle X_{i+1}) + \text{Hw}(\triangle X_{i+2}) &= \text{Hw}(\triangle X_i) + \text{Hw}(\triangle X_{i+2}) + \text{Hw}(\triangle X_{i+1}) \\
&= \text{Hw}(\triangle X_i) + \text{Hw}(\triangle X_{i+2}) + \text{Hw}(\triangle L^{-1}(Y_{i+1})) \\
&\geq \text{Hw}(\triangle X_i \oplus \triangle X_{i+2} \oplus \triangle L^{-1}(Y_{i+1})) \\
&= \text{Hw}(L(\triangle L^{-1}(Y_{i+1})) \oplus \triangle(L^{-1}(Y_{i+1})) \\
&= 5[\text{ since } \beta_d(L) = 5 \text{ for } L_1 \text{ and } L_2].
\end{aligned}$$ □

Theorem 3.10. *Any four consecutive rounds of* FeW *(i^{th} to $i+3^{rd}$ round) have at least three differentially active round functions.*

Proof. (Refer Fig. 3) Let us assume that round function in i^{th} round is differentially passive then input to this round will be of the form:

$$(\triangle P_i \neq 0) \,||\, (\triangle P_{i+1} = 0) \quad i.e. \quad \triangle X_i = 0 \;\&\; \triangle Y_i = 0.$$

Whereas $\triangle X_i = 0$ implies that input to round function consists all zero nibbles, therefore output will also contain all zero nibbles. As a result, there will be no active S-box in this round. So, the number of active S-boxes in this round is zero and input to $(i + 1)^{th}$ round will be of the form:

$$(\triangle P_{i+1} = 0) \,||\, (\triangle P_{i+2} = \triangle P_i \neq 0)$$

i.e. input to round function in $(i + 1)^{th}$ round is $\triangle X_{i+1} \neq 0$ and output is $\triangle Y_{i+1} \neq 0$. We consider that only one nibble of $\triangle X_{i+1}$ is non-zero. Therefore, there will be at least one active S-box in this round. We obtain input to $(i+2)^{nd}$ round of the form:

$$(\triangle P_{i+2} = \triangle P_i \neq 0) \,||\, (\triangle P_{i+3} \neq 0).$$

This asserts that input and output to round function in $(i + 2)^{nd}$ round are $\triangle X_{i+2} \neq 0$ and $\triangle Y_{i+2} \neq 0$ respectively. Since $\triangle X_{i+2} = \triangle P_{i+3}$ and $\triangle P_{i+3} = \triangle Y_{i+1}$, one gets $(\triangle X_{i+2} = \triangle Y_{i+1})$. As, branch number of Linear layers L_1 and L_2 is 5, we get minimum 4 non-zero nibbles in $\triangle Y_{i+1}$ corresponding to one non-zero nibble in $\triangle X_{i+1}$. This ensures that $\triangle X_{i+2}$ consists of 4 non-zero nibbles and the minimum number of active S-boxes in this round are 4. If the round function in $(i + 3)^{rd}$ round is passive (i.e. $\triangle X_{i+3} = 0$), then input to this round should be of the form:

$$(\triangle P_{i+3} = \triangle P_i \neq 0) \,||\, (\triangle P_{i+4} = 0).$$

To get $\triangle P_{i+4} = 0$, the following expression should be satisfied:

$$\triangle P_i \oplus \triangle Y_{i+2} = 0 \ i.e. \ \triangle P_i \oplus rF(\triangle P_{i+3}) = 0.$$

Finally, we get $\triangle P_i \oplus rF(rF(\triangle P_{i+2})) = 0$ which is represented in terms of input to i^{th} round by $\triangle P_i \oplus rF(rF(\triangle P_i)) = 0$. We searched this relation for all possible 32-bit input differences (i.e. 2^{32} values with $\triangle P_i \neq 0$) using a C programme. This relation was not satisfied for any non-zero value of $\triangle P_i$, which proves the theorem. □

We now provide crude estimates to prove the resistance of full round *FeW* to differential attack. It is shown by providing a lower bound on the number of active S-boxes in any 27 round differential characteristic. The following theorem assures the resistance of full round *FeW* against the differential attack.

Theorem 3.11. *Any differential characteristic for 27 rounds of FeW has minimum 45 active S-boxes and hence the probability of this differential characteristic is* 2^{-90}.

Proof. We use the fact that any three rounds of *FeW* has at least 5 active S-boxes. Therefore, $27(= 3 \times 9)$ rounds will have at least $45(= 5 \times 9)$ active S-boxes. So, the maximum probability of any single 27 round differential trail is $(2^{-2})^{45} = 2^{-90}$. If we use 27 round trail to recover subkeys of 32 round *FeW*, it will require 2^{90} chosen plaintexts which is more than the amount of available data. This theorem ensures that full round *FeW* is secure enough against differential attack. □

We assumed that *FeW* is broadly a Feistel based cipher to prove the theorems 3.8 to 3.11. We did not consider the effect of mixing operation in these results. Now we consider the *Feistel-M* structure which shows the actual impact of mixing operation on security bounds against differential attack. We use branch-and-bound based algorithm given by M. Matsui [30] to search the best differential trail. We get the following results by searching the best differential trails of *FeW*.

Round Index	Input Difference	Swap LSB $(A\|\|B)$	S-box (DDT)	Linear Layer $(L_1\|\|L_2)$	Probability $(-log2)$
1	0200 0000 0000 0000	0000 0000	0000 0000	0000 0000	0
2	0000 0000 0200 0000	0200 0000	0d00 0000	b7cb 0000	2
3	0200 0000 b7cb 0000	b700 00cb	a700 00fe	022b 8066	11
4	b7cb 0000 002b 8066	0066 802b	00f4 a0aa	b792 fa00	23
5	002b 8066 0059 fa00	0000 fa59	0000 46c5	0000 c166	34
6	0059 fa00 002b 4100	0000 412b	0000 fedb	0000 5e00	43
7	002b 4100 0059 a400	0000 a459	0000 463b	0000 4100	54
8	0059 a400 002b 0000	0000 002b	0000 00be	0000 a404	58
9	002b 0000 0059 0004	0004 0059	0005 00b1	5aaf 5b7c	65
10	0059 0004 5a84 5b7c	←	Output Difference		

Table 3. Optimal Differential Trail

We need a differential trail with probability greater than 2^{-64} to mount a successful differential attack on *FeW*. We did not find any differential trail with probability greater than 2^{-65} for 9-round *FeW*. We can use 8 round differential trail (Table 3) with probability 2^{-58} to attack reduced round *FeW*. We can recover some of the subkey bits from the outermost rounds of the trail using differential attack and the remaining subkey bits are found using exhaustive trials.

We guess the subkey bits corresponding to active S-boxes only. There is a 4-bit subkey to guess corresponding to one active S-box. Since any three round differential trail will have at least 5 active S-boxes, so we need to guess at least 20 subkey bits in any three additional outer rounds of the trail. If we add 3 rounds on the top and 3 rounds on the bottom of the trail, then we can get 40 subkey bits of 14 round *FeW*. If we allow to add 3 more rounds on the top or bottom, then we can recover 60 subkey bits of 17 round *FeW*. Therefore, we estimate that it is not possible to mount any useful differential attack on *FeW* beyond 17 rounds. Although, we can apply differential attack up to a maximum of 17 rounds by guessing 2^{60} possible values of 60-bit subkey.

If we remove the mixing operation used in the round function of *FeW*, then any 22 round differential trail will consist of 31 active S-boxes. Thus, we can recover the subkeys for the 31 rounds of *FeW*. However, the number of rounds resisting the differential attack is reduced to 17 after using the *Feistel-M structure*.

3.2. **Multiple Differential and Structure Attacks.** We find a high probability differential trail and use it to recover the secret key in the classical form of a differential attack. The main impediment in mounting a differential attack on the greater number of rounds of a cipher is the data complexity. In order to reduce the data complexity, multiple differential [6] and structure attacks [43] are devised. These variants have been proved very effective in reducing the data complexity [43] and covering the greater number of rounds than the differential attack for lightweight block cipher PRESENT. We can cluster many differential trails with different input and output differences in a multiple differential attack. In structure attack, we try to find a collection of all possible differential trails with the different input differences but the same output difference to form a differential [26]. The probabilities of all the trails belonging to the differential is combined to get a lower bound on the probability of this differential which becomes the upper bound for the complexity of the attack.

S. No.	# Differential Trails	Probability ($-log2$)
1	32	65
2	128	66
3	128	67
4	44	68
5	132	69
6	1202	70
7	12365	71
8	71590	72

TABLE 4. Multiple Differential Trails

We searched for all differential trails with probability between 2^{-65} and 2^{-72} based on the branch-and-bound search algorithm for 9-round *FeW* (Table 4). We find that it is not possible to construct a structure using these trails and it cannot be used to mount a structure attack with reduced complexity. So, we did not find any useful structure which can be used to cluster the differential trails in a differential and reduce the complexity of the attack. We believe that clustering of the differential trail is not possible beyond 9 rounds in case of *FeW*.

3.3. **Differential Factors in *FeW*.** Differential factor [42] α of a S-box results in invariant output difference β when possible input pairs of S-box are XORed with some non-zero value α. Presence of Differential factors in S-box significantly reduces the time complexity of differential attack. S-box used in *FeW* is free from Differential factors which is guaranteed by the Conjecture [42] "A differential 4-uniform S-box S has a differential factor for the input difference β if and only if the β-*th* column of the DDT table of S consists of only zeros and fours". Since there does not exist any column in DDT of S (Table 2) which consists of only zeros and fours, so the differential factors cannot be applied on *FeW* to reduce the complexity of the differential attack.

3.4. **Impossible Differential Cryptanalysis.** Impossible Differential attack [3,21] is an extension of basic differential attack. This attack has been widely applied and given best results on some ciphers like CLEFIA etc. Zero probability differential trails are used in this attack while differential attack make use of high probability differential trails. We use impossible differentials to recover the key material by sieving out the keys suggesting the impossible differential from the list of all possible keys. The key (or keys) still remaining in the list are the candidates for the correct key. We process the 64-bit input in four groups G1, G2, G3 and G4, where each group consists of four 4-bit nibbles. We observe the following four facts from DDT (Table 2) and differential branch number β_d:

(1) A nibble with zero input difference to DDT of S-Box outputs a zero difference (i.e. $0 \rightarrow 0$) with probability 1.
(2) A nibble with some non-zero input difference α ($1 \leq \alpha \leq 15$) to DDT of S-Box certainly outputs some non-zero difference β ($1 \leq \beta \leq 15$) (i.e. $\alpha \rightarrow \beta$) with probability 1
(3) Any group with all zero nibbles as input difference to linear layer returns a zero difference (i.e. $0000 \rightarrow 0000$) with probability 1.
(4) Any group with non-zero difference in 1,2,3,4 nibbles as input to linear layer produces at least 4,3,2,1 (respectively) nibbles with non-zero difference.

#R	**Forward Differential**				**#R**	**Backward Differential**			
	G1	G2	G3	G4		G1	G2	G3	G4
0	$\alpha 000$	0000	0000	0000	6	0000	0000	0000	000α
1	0000	0000	$\alpha 000$	0000	5	000α	0000	0000	0000
2	$\alpha 000$	0000	$\Gamma\theta\zeta\eta$	0000	4	$\Gamma\prime\theta\prime\zeta\prime\eta\prime$	0000	000α	0000
3	$\Gamma\theta\zeta\eta$	0000	****	****	3	****	****	$\Gamma\prime\theta\prime\zeta\prime\eta\prime$	0000

Table 5. Impossible Differential Trail

We construct 6 round impossible differential trail (Table 5) by proving the contradiction between a 3-round forward differential trail and a 3-round backward differential trail. These differential trails occur with probability 1. We get a 3-round forward differential trail in the following way:

(1) Set non-zero difference α to first nibble in G1 and the remaining nibbles in G1 and all nibbles in G2, G3 and G4 are assigned a 0 difference.
(2) Any group with 0 difference outputs a 0 difference with probability 1 (see Fact 3 and Result 1). We get the difference α positioned to the first nibble in G3 and 0 differences in the remaining nibbles after first round.
(3) In second round, we get some non-zero difference β with probability 1 after applying DDT of S-box on α (see Fact 2). We have one nibble in G3 with non-zero difference β as input to linear layer which returns 4 nibbles with some non-zero difference $(\Gamma, \theta, \zeta, \eta)$ (Fact 4 and β_d). After second round, we get first nibble in G1 as α and G3 as all non-zero nibbles $(\Gamma, \theta, \zeta, \eta)$.
(4) In third round, two rightmost nibbles of G3 & G4 are swapped and we get two non-zero nibbles in each of G3 and G4. We get the output from round function as all * values (i.e. any value) in G3 and G4 with at least three nibbles with non-zero difference in each of the group G3 and G4 (see Fact 4). After this round, G1 becomes $((\Gamma, \theta, \zeta, \eta)$, all nibbles of G3 and G4 becomes * values while all nibbles of G2 remains with 0 difference.

In a similar way, we get a backward differential trail for 3 rounds with G1 and G2 being * values, G3 as $(\Gamma\prime, \theta\prime, \zeta\prime, \eta\prime)$ and all nibbles of G4 as 0 difference. We get a contradiction for G2 between forward and backward differential trails at third round. Forward trail consists of 0 differences in G2 while backward differential trail consists of 3 or 4 nibbles with non-zero difference. We get the following 6-round impossible differential trail using miss-in-the-middle technique:

$$(\alpha 000000000000000)6R \nrightarrow (000000000000000\alpha)$$

We find the values of secret key using the impossible differentials by adding the rounds on the top and bottom of the trail. We choose the plaintext pairs with the input difference suggested by the trail and get the corresponding ciphertext pairs. We process the ciphertext and guess the key bits involved in the additional rounds and sieve out the wrong keys suggesting the impossible difference. We know that any 3 rounds of *FeW* have at least 5 active S-boxes. So, we need to guess 20 subkey bits corresponding to these 5 active S-boxes in 3 rounds and 40 subkey bits corresponding to 10 active S-boxes in 6 rounds. If we add 3 rounds on the top and 6 at the bottom of the trail, then we need to guess all possible values of 60-bit subkey. We can apply this attack on 15 rounds of *FeW* out of total 32 rounds.

3.5. **Linear Cryptanalysis.** Linear cryptanalysis proposed by Matsui [14, 29] have been proven most effective on DES. We can show the resistance to linear attack similar to the differential attack. This is a known plaintext attack and it exploits the probabilistic relation involving the plaintext, ciphertext and subkey bits which are bounded away from half. These relations are used to find the values of key bits. Some bit positions in the plaintext and ciphertext are tapped which is known as input and output mask value. Linear Branch number [20] is used to count the minimum number of active S-boxes in a trail, so we first define the linear branch number of a linear layer.

Definition 3.12. Linear Branch number β_l of a linear function L is defined as:

$$\begin{aligned}\beta_l(L) &= \min_{(\Gamma q \neq 0, \Gamma q \in \{0,1\}^{16})} (\mathrm{Hw}(L^*(\Gamma q) + \mathrm{Hw}(\Gamma q)) \\ &= \min_{(\Gamma q \neq 0, \Gamma q \in \{0,1\}^{16})} (\mathrm{Hw}((\Gamma t) + \mathrm{Hw}(\Gamma q))\end{aligned}$$

where Γq is an output mask value and Γt is an input mask value of the linear layer L. Function L^* is the linear function of the mask values concerned to L.

In case of *FeW*, linear branch number of the linear layers L_1 and L_2 used in rF is 5. We have applied L^* on all possible 16-bit mask values Γq and calculated $L^*(\Gamma q)$. For each non-zero mask value Γq and the corresponding value

```
**For:** $\Gamma t$=0 to 15
**For:** $\Gamma q$=0 to 15
**For:** A = 0 to 15
    If ($parity[A\&\Gamma t] = parity[S[A]\&\Gamma q]$)
    Then $LAT[\Gamma t][\Gamma q] + +$
**Output:** ($LAT[\Gamma t][\Gamma q] - 8$)
```

Algorithm 3. Linear Approximation Table

$L^*(\Gamma q)$, we count the minimum number of non-zero nibbles in the input and output using a computer programme and this count comes out to be 5. It is observed from the linear approximation table of S-box (Table 6) that non-zero mask value always outputs a non-zero mask value with bias ($\varepsilon = p^{-\frac{1}{2}}$) and the zero mask value always returns a zero mask value with bias $\frac{1}{2}$. Therefore, the number of non-zero branches as input to the round function is same as the number of non-zero branches as input to the liner layer. We estimate the minimum number of active S-boxes in a trail and use it to prove the resistance of *FeW* from linear attack.

3.5.1. *Linear Approximation Table.* We denote an output mask value by Γq and input mask value by Γt. We construct a parity check array for all 4-bit values (i.e. t to 15) as follows:

$$parity[16] = \{0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0\}.$$

We find the number of matches between the linear equation represented in hexadecimal as Γt and the sum of the output bits represented in hexadecimal as Γq. We get the linear approximation table (LAT) by subtracting 8 from $LAT[\Gamma t][\Gamma q]$ and dividing it by 16 gives the linear approximation bias. We run algorithm 3 to construct Table 6:

In case of *FeW*, linear branch number of the linear layers L_1 and L_2 used in rF is 5. We have applied L^* on all possible 16-bit mask values Γq and calculated $L^*(\Gamma q)$. For each non-zero mask value Γq and the corresponding value $L^*(\Gamma q)$, we count the minimum number of non-zero nibbles in the input and output using a computer program and this count comes out to be 5. It is observed from the linear approximation table of a S-box (Table 6) that non-zero mask value always outputs a non-zero mask value with bias ($\varepsilon = p^{-\frac{1}{2}}$) and the zero mask value always returns a zero mask value with bias $\frac{1}{2}$. Therefore, the number of non-zero branches as input to the round function is same as the number of non-zero branches as input to the liner layer. We estimate the minimum number of active S-boxes in a trail and use it to prove the resistance of *FeW* from linear attack.

$\Gamma t \rightarrow$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\Gamma q \downarrow$																
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-2	2	2	-2	0	0	0	4	-2	-2	2	2	0	4
2	0	2	0	2	0	-2	0	-2	2	0	2	0	2	4	2	-4
3	0	2	2	0	2	4	-4	2	-2	0	0	-2	0	2	2	0
4	0	2	-2	0	-2	0	0	2	0	2	2	-4	-2	0	-4	-2
5	0	-2	-4	-2	0	2	0	-2	0	2	0	-2	0	-2	4	-2
6	0	0	-2	-2	2	-2	-4	0	-2	2	0	4	0	0	-2	-2
7	0	-4	0	0	4	0	0	0	2	-2	-2	-2	-2	2	-2	-2
8	0	0	0	-4	0	0	0	-4	-2	-2	2	-2	2	2	-2	2
9	0	0	2	2	-2	2	0	-4	-2	2	-4	0	0	0	-2	-2
A	0	2	0	-2	0	-2	0	2	0	-2	-4	-2	4	-2	0	-2
B	0	2	-2	0	-2	0	-4	-2	4	-2	-2	0	-2	0	0	2
C	0	-2	-2	0	-2	4	0	2	2	0	0	2	4	2	-2	0
D	0	2	0	2	4	2	0	-2	2	0	2	0	2	-4	-2	0
E	0	4	-2	-2	2	2	-4	0	0	0	-2	2	-2	2	0	0
F	0	0	4	4	0	0	0	0	4	4	0	0	0	0	0	0

Table 6. Linear Approximation Table

Theorem 3.13. *Any linear characteristic for 27 rounds of* FeW *has a minimum 45 active S-boxes and hence the maximal bias of this 27 round linear trail is* 2^{-92}.

Proof. We again consider *FeW* broadly a Feistel structure based design. Branch number of linear layers L_1 and L_2 of *FeW* is 5 and the maximal bias [29] of the S-box is 2^{-2} (Table 6). We start with all zero nibbles in left half and the right half contains non-zero nibbles. We assume that there is at least one nibble as non-zero mask value in right half Γq (Fig. 4). Round function returns a zero mask value corresponding to all zero mask values, so first round trail consists zero active S-box. In second round, output mask value Γq corresponds to the input mask value Γt and this becomes output mask value for the next round. We get Γq as input mask value and there are minimum 5 active S-boxes in these two rounds (see β_l). This shows that any 3 rounds linear trail of *FeW* has minimum 5 linearly active S-boxes. Using Matsui‘s Piling-up lemma [29], we get maximal bias for any 3 round linear trail as:

$$\epsilon_3 = 2^4 \times (2^{-2})^5 = 2^{-6}.$$

Similarly, we get the maximal bias for any $27 (= 3 \times 9)$ round linear trail as:

$$\epsilon_{27} = 2^8 \times (2^{-6})^9 = 2^{-46}.$$

If we assume that linear attack is applied on full round *FeW* using 27 round differential trail then the amount of known plaintext/ciphertext data required will be of order $2^{92} (\sim \epsilon^{-2})$ which is greater than the available data. Any kind of cryptanalytic attack on block ciphers is considered successful if it recovers the key using the available data, whereas 2^{64} is the upper limit of data available in case of *FeW*.

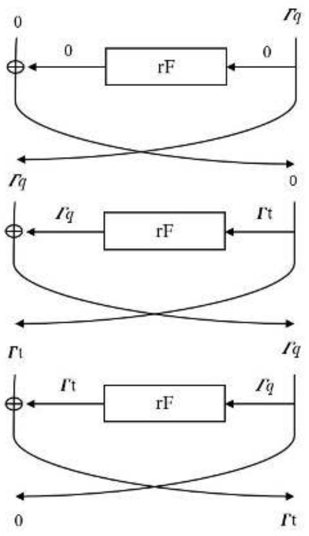

FIGURE 4. Linear Trail

□

3.6. **Related Key Cryptanalysis.** We exploit the weaknesses in the key schedule of a block cipher by a related key attack [2] and slide attack [5]. There is a practice to use a round dependent constant to protect the subkeys from the slide attacks. Therefore, we have used a round dependent constant so that round subkeys cannot be “slid” easily. For each 16-bit subkey generation, we apply a non-linear S-box three times to mix the current contents of the key register MK. After 13 rounds, all subkey bits become a non-linear function of the user supplied key. As a result, there is no subkey bit which can be expressed as a linear function of the user supplied key bits after 13 rounds.

3.7. **Statistical Tests.** NIST Statistical Test Suite, SP800-22 [16] is a widely used application for testing the output bit sequence. It consists of 15 different tests which test the random nature of the output bit sequence generated from a cipher. We have generated the bit sequence from lightweight block cipher *FeW* using two different modes of operation, output feedback mode and counter mode. For each mode, we have taken 100 different random 64-bit plaintexts and encrypted these using 100 different random 80-bit keys. For each mode, we have generated 100 files and each file contains 10^7 bits sequence. We have collected the experimental results of the both type of data in the table 7 which shows the P-value and the pass percentage of the files for each test. We observe that output bit sequence generated using *FeW* is totally random in nature.

Statistical Test	Output Feedback mode		Counter mode	
	P-Value	Proportion	P-Value	Proportion
Frequency	0.911413	98	0.816537	99
Block Frequency (m=64)	0.045675	98	0.275709	100
Cumulative Sum-Forward	0.759756	98	0.851383	99
Cumulative Sum-Backward	0.955835	99	0.955835	99
Runs	0.030806	100	0.911413	99
Longest Runs of Ones	0.554420	100	0.867692	100
Rank	0.455937	100	0.249284	98
Universal	0.455937	99	0.249284	99
Approximate Entropy	0.419021	100	0.955835	100
Serial	0.437274	100	0.514124	98
Serial	0.514124	99	0.595549	100
Linear Complexity	0.637119	97	0.455937	98

Table 7. Statistical Results

3.8. **Avalanche Effect.** Avalanche effect quantifies the effect of changing the bits in plaintext or key on the ciphertext bits. It states that if we change 1 bit in the plaintext or key, then there must be approx. 50 percent change in the ciphertext bits. We show the effect of changing the 1 bit in plaintext for 1040 executions of *FeW*. We observe that there is approximately 50 percent change on an average in the ciphertext bits. Similarly, we have performed this exercise for 1053 executions for 1 bit change in the key. The detailed results are shown in Fig. 5 where series 1 represents the result for change in plaintext bits and series 2 refers to change in the key bits.

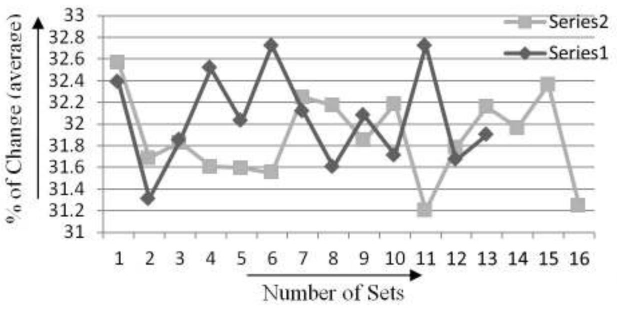

Figure 5. Avalanche Effect

3.9. **Performance Evaluation.** We designed *FeW* focusing its application in software oriented environments. So, we made used those operations which are considered to be lightweight when implemented in software viz. cyclic shifts and XORs. We use single 4×4 S-box in the encryption, decryption and key expansion algorithms. Linear operations L_1 and L_2 are applied on 16-bit branches which are two different combinations of cyclic shifts and XOR operations. This type of linear layer is approximately two times efficient in software in comparison to a linear layer comprising the bitwise mixing operations. PRESENT and RECTANGLE use bitwise mixing operations in its linear layers while we use cyclic shifts and XOR in the linear layer of *FeW*. Therefore, *FeW* performs better in software implementations. We describe the parameters of PRESENT, RECTANGLE and *FeW* in Table 8.

We compare throughput of *FeW* with PRESENT and RECTANGLE on a Windows based PC with the following configuration: Intel Core 2 Duo E8400 @ 3.00 GHz processor with 1 GB RAM. We encrypt five files with different

Cipher	PRESENT	RECTANGLE	*FeW*
Block Size (bit)	64	64	64
Key Size (bit)	80	80	80
# Rounds	31	25	32

TABLE 8. Parameters

sizes in KB (87.8, 166, 244, 400, 556) using these ciphers in output feedback mode and measure the throughput of each cipher (Table 9). This comparison indicates that *FeW* is significantly efficient than lightweight block ciphers PRESENT and RECTANGLE in software performance.

	Throughput (MB/Sec)		
File Size (MB)	PRESENT	RECTANGLE	*FeW*
0.085742	0.352849	0.496193	0.734723115
0.162109	0.381075	0.553652	0.84232353
0.238281	0.395029	0.574726	0.889109142
0.390625	0.403997	0.590961	0.92546525
0.542969	0.403904	0.599369	0.938742652

TABLE 9. Performance Comparison

4. Conclusion

We proposed a secure and efficient lightweight block cipher *FeW* by devising a novel mixing approach between the Feistel and 4-branch generalised Feistel structures. We named this structure as *Feistel-M structure* which is used for the first time in the design of a block cipher. We analysed the security of *FeW* against basic cryptanalytic attacks which proves that it has enough security margins to counter these attacks. We believe that any cryptanalytic attack cannot be applied on *FeW* to recover the secret key beyond 17 rounds. There is a large variety of attacks which can be applied on block ciphers. In future, we will analyse it against other advanced cryptanalysis methods and we will use it to design other cryptographic primitives.

Acknowledgement

We would like to thank Director, SAG and Ms. Pratibha Yadav for their valuable guidance and continuous support to work in this direction. We would also like to thank anonymous reviewers who have invested their precious time to suggest the improvements in the paper.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

References

[1] Beaulieu, R., Shors, D., Smith, J., Clark, S.T., Weeks, B., Wingers, L., *The SIMON and SPECK families of lightweight block ciphers*, Cryptology ePrint Archive, Report **2013/404**(2013). 1

[2] Biham, E., *New types of cryptanalytic attacks using related keys*, EUROCRYPT'93, LNCS, **765**(1994), 398–409. 3.6

[3] Biham, E., Biryukov, A., Shamir, A., *Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials*, EUROCRYPT'99, LNCS, **3027**(1999), 12–23. 3.4

[4] Biham, E., Shamir, A., *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, **4(1)**(1991), 372. 3.1

[5] Biryukov, A., Wagner, D., *Slide attacks*, In: Knudsen, L.R. (ed.) FSE 1999 LNCS, **1636**(1999), 245–259. 3.6

[6] Blondeau, C., Gerarad, B., *Multiple differential cryptanalysis: Theory and practice*, In: Jaux, A. (ed.) FSE 2011. LNCS, **6733**(2011), 35–54. 3.2

[7] Bogdanov, A., Analysis and Design of Block Cipher Constructions, PhD thesis, 2009. 3.1.1

[8] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C., *Present: An ultra-lightweight block cipher*, CHES 2007, LNCS, **4727**(2007), 450–466. 1

[9] Bogdanov, A., Rijmen, V., *Linear Hulls with correlation zero and linear cryptanalysis of block ciphers*, Cryptology ePrint Archive, Report **2011/123**(2011). 3

[10] Cannire, C., Dunkelman, O., Knezevi, M., *Katan and Ktantana family of small and efficient hardware-oriented block ciphers*, CHES 2009, LNCS, **5747**(2009), 272–288. 1

[11] Carlet, C., Ding, C., *Nonlinearities of S-boxes*, Finite fields and their applications, **13(1)**(2007), 121–135. 2.5

[12] Daemen, J., Rijmen, V., The Design of Rijndael, Springer-Verlag, 2002. 1

[13] Diffie, W., Ledin, G. (translators), *SMS4 encryption algorithm for wireless networks*, Cryptology ePrint Archive, Report **2008/329**(2008). 1

[14] Emami, S., Ling, S., Nikolic, I., Pieprzyk, J., Wang, H., *The Resistance of PRESENT-80 against related key differential attacks*, Cryptology and Communications, **Sep. 2014**(2014), 171–187. 3.5

[15] Engels, D., Saarinen, M.J.O., Schweitzer, P., Smith, E. M., *The Hummingbird-2 Lightweight Authenticated Encryption Algorithm*, RFID Sec 2011, 7th Workshop on RFID Security and Privacy, 26-28, Amherst, Massachusetts, USA, 2011. 2.5

[16] Gallagher, P. (Director), A Statistical Test for Random and Pseudorandom Number Generators for Cryptographic Application, Apr, 2010. 3.7

[17] Gong, Z., Nikova, S., Law, Y. W., *KLEIN: A New Family of Lightweight Block Ciphers*, RFID Sec 2011, LNCS Vol. 7055, 1-18, 2011. 1

[18] Guo, J., Peyrin, T., Poschmann, A., The LED Block Cipher, Cryptographic Hardware and Embedded Systems CHES 2011, LNCS, 2011. 1

[19] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S., *HIGHT: A new block cipher suitable for low-resource device*, CHES 2006, LNCS, **4249**(2006), 46–59. 1

[20] Kanda, M., Practical Security Evaluation against Differential and Linear Cryptanalysis for Feistel Ciphers with SPN Round Function, SAC 2000, LNCS 2012, 324-338, Springer-Verlag, 2001. 3.1.1, 3.1.2, 3.5

[21] Kim, J., Hong, S., Sung, J., Lee, C., Lee, S., *Impossible differential cryptanalysis for block cipher structure*, INDOCRYPT 2003, LNCS, **2904**(2003), 82–96. 3.4

[22] Kim, T., Kim, J., Hong, S., Sung, J., *Linear and differential cryptanalysis of reduced SMS4 block cipher*, Cryptology ePrint Archive, Report **2008/281**(2008). 3.1.1

[23] Knudsen, L.R., Leander, G., Poschmann, A., Robshaw, M.J.B., *PRINTcipher: A block cipher for IC printing*, In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010, LNCS, **6225**(2010), 16–32. 1

[24] Knudsen, L., Robshaw, M.J.B., Block Cipher Companion, Book Springer, 2011, ISBN 978-3-642-17341-7. 1, 3

[25] Kumar, M., Pal, S.K., Yadav, P., *Mathematical constructs of lightweight block ciphers-A survey*, American Jr. of Mathematics and Sciences, **2(1)(2013)**, ISSN No: 2250-3102. 1

[26] Lai, X., Massey, J.L., Markov Ciphers and Differential Cryptanalysis, In: Davis D.W. (ed.) EUROCRYPT 1991. LNCS, Vol. 547, 17-38. Springer, Heidelberg, 1991. 3.2

[27] Leander, G., Paar, C., Poschmann, A., *New Lightweight DES Variants*, FSE 2007, LNCS, **4593**(2007), 196-210. 1

[28] Matsuda, S., Moriai, S., *Lightweight cryptography for the cloud: Exploit the power of Bitslice Implementations*, CHES 2012, LNCS, **7428**(2012), 408-425. 1

[29] Matsui, M., Linear Cryptanalysis Method for DES Cipher, Advances in Cryptology EUROCRYPT 1993, LNCS, Vol. 765, 386-397, Springer-Verlag, 1994. 3.5, 3.5.1

[30] Matsui, M., On Correlation between the Order of S-Boxes and the Strength of DES, In: De Santis, A. (ed.), EUROCRYPT 1994, LNCS, Vol. 950, 366-375, Springer, 1995. 3.1.2

[31] Piret, G., Roche, T., Carlet, C., *PICARO- A block cipher allowing efficient higher order side channel resistance*, ACNS 2012, LNCS, **7341**(2012), 311-328. 1

[32] Poschmann, A.Y., Lightweight Cryptography: Cryptographic Engineering for a Pervasive World, PhD thesis 2009. 1

[33] Rijmen, V., Cryptanalysis and design of iterated block cipher, PhD Thesis, 1997. 3.1.1

[34] Saarinen, M.O., *Cryptographic analysis of all 4x4 bit S-boxes*, Cryptology ePrint Archive, Report **2011/218**(2011). 2.5

[35] Shannon, C. E., *Communication theory of secrecy systems*, Bell Systems Technical Journal, (1949), 656-715. 1

[36] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T., *Piccolo: An ultra-lightweight block cipher*, CHES 2011, LNCS, **6917**(2011), 342-357. 1

[37] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T., *The 128-bit block cipher CLEFIA*, (Extended Abstract) FSE 2007, LNCS, **4593**(2007), 181-195. 1

[38] Sorkin, A., *LUCIFER: A cryptographic algorithm*, Cryptologia, **8(1)**(1984), 22-35. 1

[39] Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J., *SEA: A scalable encryption algorithm for small embedded applications*, CARDIS 2006, LNCS, **3928**(2006), 222-236. 1

[40] Su, B., Wu, W., Zhang, W., *Differential cryptanalysis of SMS4 block cipher*, Cryptology ePrint Archive, Report **2010/62**(2010). 3.1.1

[41] Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E., Twine: A Lightweight, Versatile Block cipher, ECRYPT Workshop on Lightweight Cryptography, 2011, http://www.uclouvain.be/crypto/ecryptlc11/static/post proceedings.pdf. 1

[42] Tezcan, C., Ozbudak, F., *Differential factors: Improved attacks on SERPENT*, Cryptology ePrint Archive, Report **2014/860**(2014). 3.3

[43] Wang, M., Sun, Y., Tischhauser, E., Preneel, B., *A model for structure attacks, with applications to PRESENT and serpent*, In: Canteaut, A. (ed.) FSE 2012, LNCS, **7549**(2012), 49-68. 3.2

[44] Wheeler, D., Needham, R., *TEA, a tiny encryption algorithm*, FSE 1994, LNCS, **1008**(1995), 363-366. 1

[45] Wu, W., Zhang, L., *LBlock: Lightweight block cipher*, Cryptology ePrint Archive, **2011/345**(2011). 1

[46] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I., *RECTANGLE: A bit-slice ultra-lightweight cipher suitable for multiple platforms*, Cryptology ePrint Archive, Report **2014/084**(2014). 1