MITM and Differential Fault Attack on ULBC

1st Shreyasi Ghorai
Department of Mathematics
West Bengal State University
Kolkata, India
shreyasighorai5@gmail.com

2nd Mrinal Nandi
Department of Statistics
West Bengal State University
Barasat, India
mrinal.nandi1@gmail.com

***Abstract*—ULBC is a SPN based block cipher, operates 64 bit state and use 128 bits key. Here we present meet-in-the-middle (MITM) attack on ULBC. MITM attack strategy proposed by Demirci and Selcuk. In this paper we partition cipher ULBC in two halves and separate key space by two independent set and observe matching between encryption of first half with decryption of second half. By this method, called MITM attack, we can reduce the key space for exhaustive search. Basic fault analysis of ULBC requires 192 faulty ciphertext to detect full key register. Also, we provide another fault analysis method of ULBC, which requires only average 57 faulty ciphertext to retrieve master key. Here we assume that we can induce nibble fault in after or before substitution layer to any rounds. MITM and Differential fault attack particularly exploits weakness like dependency, linearity of designing key schedule.**

***Index Terms*—ULBC, MITM Attack, Fault attack.**

I. Introduction

Ultra-light block cipher (ULBC) is (substitution permutation network) SPN based block cipher. Due to substitution box (S-box), hardware implementation of SPN may be complicated. ULBC is used in IoT application such as health care, smart cities, military purpose in recent time. Many devices are connected through a single network for transferring data is called IoT. IoT needs lightweight cryptographic algorithm to provide a secure cipher for low resource application. Lightweight block ciphers are less costly and needs less space to apply. Using these types of lightweight block ciphers, we can makes our planet smart.

Diffie and Hellman first introduced cryptographic attack meet-in-the-middle (MITM) in 1997 [1]. MITM is a chosen plaintext attack. Here we setup a specific set of events, which can occur with probability one. Also generates a 'sieving set' consisting all possible outcomes of this specific event. By this manner key candidates not producing any of this event are eliminated [2]. Basic idea of MITM attack is as follows, let E_k be encryption function of any block cipher with SPN structure and n be block size. Also let $K = K_1||K_2$ be key, where K_1 and K_2 are independent. We can separate full cipher in two halves. If V is intermediate state and (P, C) be any plaintext-ciphertext pair then $V = F_{K_1}(P) = F^{-1}_{K_2}(C)$, for independent key guess. Thus $C = F_{K_2}(V) = F_{K_2}(F_{K_1}(P)) = F_K(P)$ [3]. Here $F_{K_1}(P)$ will be forward chunk and $F^{-1}_{K_2}(C)$ will be backward chunk. They meet in several matching points; V is one of them. MITM attack clearly defined by its path. All the values of forward and backward chunks can be computed independently [4].

Basic Assumption of MITM attack is that few bits v of internal state (IS) may be computed from plaintext by using K_1 these same bits may also be computed from the corresponding ciphertext with K_2. Now intersecting partial key $K_c = K_1 \cap K_2$, information of the key common to K_1 and K_2, where $K_1{}'$ be the part of K_1 not in K_c and $K_2{}'$ be the part of K_2 not in K_c. In this attack, few bits of Initial Structure (IS) l are computed from the left by guessing $k_1 \in K_1$. Then same bits r computed from right by guessing $k_2 \in K_2$. Now checking $l = r$, valid key candidates can be determined [5].

Side channel attack is one of most powerful attacks for getting secret information of a cipher implementation in hardware environment. Differential and linear cryptanalysis provides weakness of cryptographic algorithm, whereas side channel attack stands on the information gathered from physical implementation, fault injection, power consumption and timing information. Thus, Differential fault analysis (DFA) is most important side channel attacks. The attacker can obtain a faulty ciphertext injected by a random fault in one round through encryption process for most of DFAs. Using criteria of differential cryptanalysis, last round key can be detected.

A. Related Works:

Kazumaro Aoki and Yu Sasaki propose two techniques of pre-image attack, one is splice and cut and another is partial matching. In splice and cut, first and last steps are considered as consecutive steps. Then divide attack target in such a way that each chunk (forward or backward) includes one message word independent of other chunk. This kind of message word is called 'neutral word'. Partial matching technique execute only one word instead of all words [6].

Now choice of two subsets of key bits is not always possible. A. Bogdanov and C. Rechberger introduces a new attack called 3-subset-Meet-in-the-Middle Attack [7]. Here they remove some restrictions of choosing key bits. The goal of this method is reducing key space and searching right key in reduced space. This attack is based on chosen plaintext attack. This kind of attack consists of two phases called precomputation phase and key elimination phase. In first phase i.e., precomputation phase, we can construct a 'seiving set', the possible outcomes of a specific event are elements of this set. In second phase

i.e., key elimination phase those key candidates, don't satisfy conditions of sieving set must be eliminated.

MITM attack is effectively applied on block ciphers namely Hierocrypt-3 [8], PRESENT [26], KATAN [5], GIFT [18]. To reduce key search space From 2^{128} for 128 bit key register, MITM is used. Also, attacker can partially guess the key.

Being a nonlinear part of block cipher, the input-output differentials of S-box can be used to detect the secret key if input contains its information [9]. Differential Fault Attack (DFA) can be effectively applied to the block ciphers CLEFIA [10], PRESENT [11], GIFT [11], HIGHT [12], FEW [9], ITUbee [13], PIPO [14], KASUMI [15], LED-64 [16]. Lim et al. propose a DFA logic on Lightweight block cipher PIPO, which shows 64 faulty ciphertext is sufficient to recover the secret key. They also demonstrate EM-FI attack, which is fatally vulnerable in the real device, as PIPO without Fault Injection (FI) countermeasure does not provide security against DFA [17].

The Implementation cost of an S-box of the block cipher PRESENT is 22 units, whereas cost of an S-box of the block cipher GIFT and ULBC is 16 units. But the differential probability of PRESENT and ULBC is same equals 2^{-2}, whereas the differential probability of GIFT is $2^{-1.415}$. Total of 32 nibble faults are required for PRESENT and 64 nibble faults are required for GIFT to recover the original key.

B. Our Contribution:

In this paper we are trying to focus on MITM attack on our designed cipher ULBC. We have designed ULBC for reducing implementation cost of iterated SPN cipher [19]. In that paper Ghorai and Nandi take care of Bad output Good input property (BOGI), which is proposed by Banik et al. [18] with more rounds. We have discussed two Fault attack method using differential characteristics and explain detailed process of retrieving round keys.

II. ULBC Block Cipher: Specification and Design Rationale

This section proposes our new SPN based ultra-lightweight block cipher [19].

A. Specification

The block length of this cipher is 64 bits. In each round 3 steps are performed, namely, sub cell, permutation bit, add round key. The cipher accepts 64-bit plaintext to generate ciphertext. Then all steps are performed one by one.

□ **Sub Cell:** To minimize area we have used 4-bit S-box. 4-bit to 4-bit S-box $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ of our cipher is given by table I.

Here 64-bit $p_0p_1p_2 \dots p_{63}$ input is taken and divided into 16 parts ($x_0x_1 \dots x_{15}$), each part contains 4-bits. Then apply S-box function to each part and get the output of Sub cell.

□ **Bit Permutation:** The bit permutation used in our cipher is given in the table II, where each bit is taken as i.

□ **Key register update:** In every round after extracting round key, key register must have to updated. 128 bits state key is divided into 8 parts, each part contains 16 bits i.e., $statekey = k_0k_1 \cdots k_7$. After extracting leftmost 32 bits as round key, key register is updated as follows:

$$k_0||k_1|| \cdots ||k_7 = k_2||k_3||. \cdots ||k_7|| \cdots ||k_0 >> 2||k_1 << 4$$

where$>> i$ is an i bit right rotation and $<< i$ is an i bit left rotation within 16 bits.

$k_0 = U_0V_0 = u_0u_1 \cdots u_7v_0v_1 \cdots v_7$
$k_1 = U_1V_1 = u_8u_9 \cdots u_{15}v_8v_9 \cdots v_{15}$
$k_2 = U_2V_2 = u_{16}u_{17} \cdots u_{23}v_{16}v_{17} \cdots v_{23}$
$\vdots$
$k_i = U_iV_i =$
$u_{8*i}u_{(8*i)+1} \cdots u_{(16*i)-1}v_{8*i}v_{(8*i)+1} \cdots \ v_{(16*i)-1}.$

for $i = 1, 2, \dots, 7$.

□ **Key Schedule:**

As encryption algorithm is publicly available, key schedule algorithm and secret key is very crucial in designing block cipher. Round keys have to be independent, otherwise, it cannot resist key schedule attack. According to National Institute of Standards and Technology (NIST), USA Gov., minimum key length to resist brute force attack is 112 bits. We choose 128 bits key register.

Here we specify adding round key and round constants. 64 bit key $K = k_ik_{i+1}$ extracted from 128 bit key state. If cipher state is $\{s_i\}_{i=0}^{63}$, adding round keys is given by

$$s_{4i+1} = s_{4i+1} \oplus u_i \quad \text{for} \quad i = 0, 1, \dots, 15$$

$$s_i = s_i \oplus 1 \quad \text{for} \quad i = 62$$

$$s_{4i+3} = s_{4i+3} \oplus v_i \quad \text{for} \quad i = 0, 1, \dots, 15.$$

Now 6-bit round constant ($c = c_0c_1c_2c_3c_4c_5$) is xored in the following bits

$$s_2 = s_2 \oplus c_0$$

$$s_6 = s_6 \oplus c_1$$

$$s_{10} = s_{10} \oplus c_2$$

$$s_{14} = s_{14} \oplus c_3$$

$$s_{18} = s_{18} \oplus c_4$$

$$s_{22} = s_{22} \oplus c_5$$

where round constants are given by the table III.

III. Basic Meet-in-Middle Attack

Basic MITM attack proposed that partition all the key bits in two independent subsets. If partial transformation of R round block cipher denoted as $\varphi_{i,j}$, which begins from round i and ends in round j. $\varphi_{1,\alpha}(p) = \varphi_{\alpha+1,R}^{-1}(c)$, where p is plaintext and c is ciphertext. This is the basic objective. So many pair of p and c may satisfy the equation, but only one pair is required, i.e., large number of collisions may find during finding of meaningful collision that is required. This meaningful collision is called golden collision [28].

TABLE I
S-BOX OF ULBC

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

TABLE II
PERMUTATION OF ULBC

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P[i]	1	16	35	50	49	0	19	34	33	48	3	18	17	32	51	2
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P[i]	5	20	39	54	53	4	23	38	37	52	7	22	21	36	55	6
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P[i]	9	24	43	58	57	8	27	42	41	56	11	26	25	40	59	10
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P[i]	13	28	47	62	61	12	31	46	45	60	15	30	29	44	63	14

TABLE III
ROUND CONSTANTS OF DIFFERENT ROUNDS OF ULBC

i	1	2	3	4	5	6	7	8	9	10	11
constants	0xF	3xC	3xA	3x9	1xE	1xD	1xB	1x7	2x7	2xB	2XD
i	12	13	14	15	16	17	18	19	20	21	22
constants	2xE	3x3	3x5	3x6	3xD	3xF	3xE	1xF	2xF	3x7	3xB

In every round, we observe that 32 bits from 128 bits master key xored to state bits of ULBC. Subkey bits derived from k_0, k_1 used in first round. In second round, subkey bits are derived k_2, k_3. Similarly, k_4, k_5 are used in round 3 and k_6, k_7 are used in round 4. If we analyze 6-round attack of ULBC, we can observe that k_6, k_7 are not used in first three rounds. Also k_4, k_5 are not used in next three round (starting from round 4). Thus we can construct two independent parts of master key for each values of k_0, k_1, k_2, k_3.

We can extend number of rounds by using various technique. Then select 8-bits from k_6, k_7 and 8-bits from k_2, k_3 as neutral bits. We can describe round keys of 15 rounds in table IV.

After round 4, all k_i's are rotated according to key schedule algorithm. k_i^F is used for forward computation and k_i^B is used for backward computation.

From Fig. 1, it is clear that forward and backward computation starts from round 4 and round 6 respectively, while the initial structure consists only three rounds. k_2 and k_3 never occurs from round 7 to round 9, thus computation can be performed independently. k_6 and k_7 never occur from round 3 to round 1. Thus, backward computation does not depend on the values of k_6 and k_7. Here we use the technique splice and cut. Hence independent computation will be continued till round 13. Next from round 10, we process partial computation and partial match (PM) in Fig. 2, which will help us to filter out key candidate, which does not follow partial matching. Let S_i^I denotes initial state of round i, S_i^S denote output after applying S-box in round i, S_i^{Perm} denotes output after bit-permutation in round i. Each state consisting 64-bits will be represented as two dimensional array for the sake of MITM attack. Each element of 4×16 two dimensional array represent 1 bit. Each 4-bit from right is considered as each column from right. Thus S-box will be applied to each column. As 16 S-boxes are there so in two dimensional array 16 columns are there. We use matrix based framework to calculate each state. This matrix representation of round function and state is uniquely determined. Each element in i-th row and j-th column corresponds to $(4j + i)$th element. Each state $b_{63}b_{62} \dots b_0$ can be represented as

$$\begin{bmatrix} b_{60} & b_{56} & \dots & b_4 & b_0 \\ b_{61} & b_{57} & \dots & b_5 & b_1 \\ b_{62} & b_{58} & \dots & b_6 & b_2 \\ b_{63} & b_{59} & \dots & b_7 & b_3 \end{bmatrix}.$$

We choose $0, 1, 2, 3$ bit positions of k_6 and k_7 as neutral bits, whenever forward computation is ongoing. Also, we select $8, 9, 10, 11$ bit positions of k_2 and k_3 as neutral bits, whenever backward computation continues. According to key schedule of ULBC, these bits are xored. Now total 16 bits are fixed as neutral bits. The values in bit positions $1, 5, 9, 13, 33, 37, 41, 45$ of state S_4^{PB} are corresponds to neutral bit positions of k_6^F and k_7^F. After applying S-box, we can compute S_5^S, where first and third slice are affected if we take 16 bits as one slice. Now first four S-boxes will affect 1st, 5th, 9th, 13th S-boxes, third slice will affect 3rd, 7th, 11th, 15th S-boxes. After permutation effected bit positions are marked as blue.

Now by observing key schedule algorithm, we can fix 24 bits of S_6^{Perm} in bit positions 0, 2, 4, 6, 8, 10, 12, 14, 17, 21, 25, 29, 32, 34, 36, 38, 40, 42, 44, 46, 49, 53, 57, 61. As k_2^B and k_3^B only will affect 17, 21, 25, 29, 49, 53, 57, 61-bit positions, we can compute k_2^B and k_3^B. By backward computation, we can observe that 2nd and 4th slice will get affected. Thus, from diagram it is clear that forward and backward computations consisting round 4 to 6, never have common bit positions, hence these computations are independent.

A. Partial matching

Main objective of this partial matching is with independent key setup reducing the searching key space. We have some

TABLE IV
SELECTION OF KEY BITS

Round	1	2	3	4	5	6	7	8	9
Sub-key	$\{k_0, k_1\}$	$\{k_2^B, k_3^B\}$	$\{k_4, k_5\}$	$\{k_6^F, k_7^F\}$	$\{k_0, k_1\}$	$\{k_2^B, k_3^B\}$	$\{k_4, k_5\}$	$\{k_6^F, k_7^F\}$	$\{k_0, k_1\}$
Round	10	11	12	13	14	15			
sub-key	$\{k_2^B, k_3^B\}$	$\{k_4, k_5\}$	$\{k_6^F, k_7^F\}$	$\{k_0, k_1\}$	$\{k_2^B, k_3^B\}$	$\{k_4, k_5\}$			

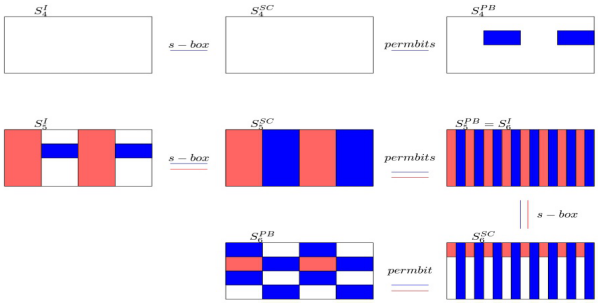

Fig. 1. 3-round initial structure for 15 round mitm attack of ULBC

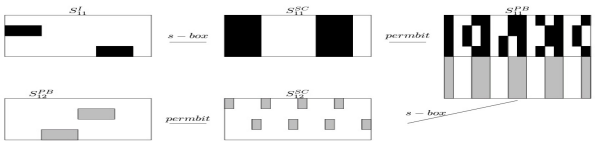

Fig. 2. Partial matching corresponding to MITM

plaintext-ciphertext pair. We know some intermediate states. By dividing whole key space into smaller ones and encrypting first half, decrypting second half, attacker can find out matching pairs. Now by guessing all possible values of neutral key bits, we can check if matching occurs or not. Thus, we partially can guess the key. Meet in the middle or matching in the middle occurs from round 10 to round 12. Whenever k_2^B, k_3^B occurs as xoring key candidate. For 2^8 choices of neutral bits of k_6^F, k_7^B. We search for matched pairs and store all 2^8 values of k_6^F, k_7^F with corresponding 2^8 values S_{12}^{Perm}.

As 8, 9, 10, 11 bits are neutral bits of k_2 and k_3 in round 6, after 2-bits right rotation of k_2, neutral bit positions of k_2 will be 10, 11, 12, 13. After 4-bit left rotation of k_3, neutral bit position of k_3 will be 4, 5, 6, 7.

Now neutral bit positions of k_6 and k_7 in round 4 are 0, 1, 2, 3. In round 12, after double right rotation, neutral bits of k_6 are 4, 5, 6, 7. After double left rotation, neutral bits of k_7 are 8, 9, 10, 11.

As explained in partial matching diagram in Fig. 2, 8 unknown bits of subkey hides 32 bits of S_{12}^I in forward computation. Attacker can compute the remaining bits of S_{12}^I. Backward computation hides 32 bits and remaining bits of S_{12}^I can be easily computed by attacker. Both chunks overlap in 24 bits value.

a) **Data Complexity:** 2^8 values of subkey are selected for forward computation and 2^8 values of subkey are selected for backward computation. This 2^{112} values of subkeys are not associated as neutral bits. Since forward and backward chunk match in 24 bits, so $2^8 \cdot 2^8 \cdot 2^{-24} = 2^{-8}$ key candidates remains for verifying. Thus for 2^{112} different values of subkey, total $2^{112} \cdot 2^{-8} = 2^{104}$ key candidates survives for exhaustive search. Memory complexity of this attack is 2^8 and time complexity is $2^{112} \cdot 2^8 + 2^{104} \approx 2^{120}$.

IV. Differential Fault Attack

Differential Fault Attack (DFA) is a cryptanalysis method to extract information about secret key by using differences between correct and faulty ciphertext. We have analysed DFA attack on ULBC by using permutation criteria and differential property. If one fault is injected in a S-box, we observe that this fault will spread into four S-boxes in next round. Now by taking differences of faulty and correct ciphertext, we can get information about fault location. Here by observing solution corresponding to DDT table, we also can identify round key. Fault attack method is first proposed by Boneh et al. [20] and differential fault attack is first proposed by E. Biham and A. Samir based on fault attack and differential attack [21]. This cryptanalytic method has been applied to PRESENT [23], GIFT [24], AES [22], SIMON [25] etc.

A. Structural Properties of ULBC

If $\Delta\alpha \in \mathbb{F}_2^4$ and $\Delta\beta \in \mathbb{F}_2^4$ be input and output differences respectively, number of solution of the equation

$$S(x) \oplus S(x \oplus \Delta\alpha) = \Delta\beta \tag{1}$$

is given in DDT table. Let this number of solutions is denoted as $DDT(\Delta\alpha, \Delta\beta)$.

Now the probability of equation (1) has no solution i.e., $DDT(\Delta\alpha, \Delta\beta) = 0$ is 0.621. Probability of equation (1) has two solution i.e., $DDT(\Delta\alpha, \Delta\beta) = 2$ is 0.282. Now probability of equation (1) has 4 solution i.e., $DDT(\Delta\alpha, \Delta\beta) = 4$ is 0.093. Probability of equation (1) has 16 solution is 0.004.

Each nibble is diffused to four different nibbles of separate group, if 4 S-boxes form a group from right to left through permutation of ULBC. Thus, by knowing about S-box differences, we have to track fault occurring bit position by analyzing permutation explicitly.

B. Basic Differential Fault Attack

In Basic Differential Fault Attack, fault is injected in only one bit of intermediate state at the beginning of last rounds S-box layer. A single bit fault is required here. The main idea of this attack is to inject single bit fault in $S^k N_j$ for k-round cipher, where N_j is nibble j. Let $S^{k*} N_j$ be faulty state and $S^k N_j$ be correct state of j-th nibble and d be difference of them. Then the value of d will be $0, 1, 2, 8$. From different values of d, different ciphertext will be generated as d is input difference of Substitution layer of kth round. If C and D are correct and faulty ciphertext respectively corresponding

to fault in jth nibble. $\Delta D^k = P\text{-}layer^{-1}(C \oplus D)$ provides difference in jth nibble after k-th rounds substitution layer. From two or three values of jth nibble difference $\Delta D^k N_j$, for different single bit fault on same nibble. All the faulty ciphertexts are generated using same plaintext and same key.

Analysing DDT table, we can get all possible inputs I_1 of jth nibble in k-th round. Now possible input of this output difference will be more than one as 1 is not present in DDT table. Thus, we have to inject another single bit fault and by applying same method, we can calculate another set of possible inputs I_2. If $I_1 \cap I_2$ is singleton set, then this provides exact input to jth nibble in k-th round. Otherwise, we have to inject another bit fault and continue with same algorithm. Now by this method, we can find out all 16 possible nibble inputs to round function in kth round.

Taking this 64-bit input as $P_1, PS(P_1) \oplus C$ provides the last round key. According to round key generation algorithm, key register of ULBC can't be obtained from one round key. 4-round consecutive round keys are needed to know about full key register.

Next aim is to detect $(k-1)$th round key. Inject fault in jth nibble before substitution layer of $(k-1)$th round and use same plaintext and same key to get faulty ciphertext. As kth round is known explicitly, so take input in kth round as C. After using last round key, decrypting faulty ciphertext, we get faulty state after $(k-1)$th round, denote it as D. Then apply same algorithm to find out $(k-1)$th round key of ULBC.

Using same strategy all $k, (k-1), (k-2), (k-3)$th round keys are detected. Hence key register of ULBC is detected.

C. Basic Assumption of Attack

a) **Power of Adversary:** For this attack, adversary should have more power to inject arbitrary fault. Adversary can access of fault injection before or after substitution layer in any round. Adversary can choose any plaintext during encryption process. Adversary can run encryption of plaintext as many times as required.

b) **Experiment limitation:** During experiment if we get same differential in same nibble, no different differential is present through 16 nibbles, discard that faulty ciphertext. It only increases data complexity, but don't provide any new information about key. Same different fault in same nibble can't provide two or more differential corresponding to any nibble. Thus, injection of fault in different nibbles is necessary. Thus, we have tried to collect all independent set of faulty ciphertext, those provides different differential in different nibbles and discard useless faulty ciphertext.

D. The Second DFA Attack

In basic DFA attack, we get each round key by using average 48 faulty ciphertext. As 4 round keys are needed to know full key register, $4 \times 48 = 192$ faulty ciphertexts are required in basic DFA Attack [27]. To reduce complexity and maintain single bit fault in each nibble of last round, we can induce one nibble fault in $(k-2)$th round, for k round cipher. As a result, in $(k-1)$th round, we get single bit fault induced in maximum 4-nibbles and kth round follow the criteria of basic DFA Attack. Since at one experiment we get maximum 16-differential nibble fault, number of required faulty ciphertext will be lesser.

Encryption process of ULBC requires 32 bits round key and 6 bits round constant. Key register consists of 128 bits. Observing key schedule and key register update, if we run ULBC upto 22 rounds, last 4 round keys should be detected. 19th, 20th, 21st and 22nd round keys are required. According to second DFA attack, for detecting 22nd round key inject fault in the beginning of 20th round. For recovery of 21st, 20th and 19th key inject fault in beginning of $19, 18, 17$ rounds respectively.

Attack steps are given in following for introducing concept of attack:

(1) For one experiment fix a plaintext P and corresponding correct ciphertext C, after 22nd round through correct encryption of ULBC. Adversary has no knowledge about key.

(2) Inject one nibble fault in rightmost nibble that generate a faulty ciphertext C^*. For injecting one fault in the beginning of 20th round, some nibbles of round 21 are affected. If fault in state 21 is denoted as ΔC^*_{21}, then $\Delta C^*_{21} = (0, 0, 0, c^*_{12,21}, 0, 0, 0, c^*_{8,21}, 0, 0, 0, c^*_{4,21}, 0, 0, 0, c^*_{0,21})$, where some of $c^*_{12,21}, c^*_{8,21}, c^*_{4,21}, c^*_{0,21}$ are nonzero.

(3) Each nonzero difference of 21st round will propagate to maximum 4 nibbles of 22nd round. If C and C^* be correct and faulty ciphertext respectively, then difference ΔC^* is $C \oplus C^*$.

(4) Since differentials are not affected by key xor, inverse permutation of ΔC^* provides output difference Δy^i_{22} of ith nibble after substitution layer in 22nd round. If corresponding input is x^i_{22} and input difference is Δx^i_{22}, then

$$\Delta y^i_{22} = S(x^i_{22}) \oplus S(x^i_{22} \oplus \Delta x^i_{22}).$$

We can tabulate possible inputs corresponding to each output difference in table V.

Now observing the table V, it is clear that the values of x^i_{22} will be more than one. For this faulty ciphertext, corresponding to each nibble i, set of all possible input denoted as L_i.

(5) Inject fault in another nibble of different groups, and generate faulty ciphertext. Here take 4 consecutive nibbles as one group from any end. So, each state contains 4 groups. Fault in two nibbles of same group can't provide two different values of same nibble. Repeat steps from (2) to (4) to generate another set of possible inputs corresponding to each nibble i, and denote it as M_i. Check $L_i \cap M_i$ is singleton or not corresponding to each i.

(6) If intersection is not singleton, then repeat steps from (2) to (5). Intersection may not be singleton in some cases, as in ULBC there exist two inputs corresponding to 4 output differences. As there are 4 groups in a state, so maximum 4 different output differences can be generated by injecting one nibble fault at a time.

(7) Now observe table VI for identifying these two inputs, provide same 4 output differences. It is clear that 0001 and 0101 provides same output differences. Now there may be ties in this round. We need an extra round to break the tie.

TABLE V
POSSIBLE INPUT CORRESPONDING TO OUTPUT DIFFERENCES OF ULBC

2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0110	0000	0001	1001	0000	0010	1001	0110	0010	0011	0000	0001	0100
0010	0111	0100	0101	1010	0001	0011	1011	0111	0011	0111	0001	0010	1000
	1110	1011	0111	1101	0011	0100		1110	1010	1000	0101	0101	1001
	1111	1100	1001	1110	0100	0110		1111	1011	1100	1101	0110	1100
		1110	1100		0101	1010			1101			1000	
		1111	1101		1000	1011			1111			1010	

TABLE VI
POSSIBLE OUTPUT DIFFERENCES CORRESPONDING TO EACH INPUT

input	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0010	0101	0010	0111	0100	0101	0011	0011	0111	0101	0110	0100	0100	0101	0011	0011
	0100	0111	1000	1000	0111	0111	1000	0101	1100	0110	1000	1000	0101	0110	0100	0100
	0111	1101	1011	1011	1000	1101	1010	1010	1110	1001	1011	1001	1100	1011	0110	1010
	1101	1110	1110	1100	1111	1110	1110	1100	1111	1111	1110	1011	1111	1101	1010	1011

Tie breaking round for unsolved nibble

Mark that nibble N_d which correspond to two inputs. Now if inverse permutation of this nibble corresponds to ith nibble of previous round, inject fault corresponding bit wiring of that nibble after substitution layer. As for example, if first nibble (from right) corresponds to two inputs, then we can inject fault in first nibble 0001 after substitution layer of previous round.

Then calculate the output difference, and check input difference 2 correspond to output difference with set of input T. Finally $T \cap \{0001, 0101\}$ provides unique input value for N_d.

(8) Hence we get round key of 22nd round by calculating $PS(state) \oplus C$, where state is state before substitution layer of 22nd round.

(9) Next aim is to detect 21st round key. Inject fault in jth nibble before substitution layer of 19th round and use same plaintext to get faulty ciphertext. As 22nd round is known explicitly, by decrypting ciphertext through last round, we get faulty ciphertext after 21 round. Now repeat steps (1)-(8) for finding round key in 21st round.

(10) Similarly, 20th and 19th round key can be detected.

We have implemented second differential fault attack for 10 different plaintexts with the ciphertexts given in table VII.

We have implemented second differential fault attack for 10 different plaintexts with the ciphertexts and the number of required faulty ciphertexts needed for different round keys given in table VII.

TABLE VII
EXPERIMENTAL FAULT ATTACK RESULT FOR ULBC

Experiment no.	22 round key	21 round key	20 round key	19 round key
1	18	15	15	17
2.	14	14	14	16
3.	16	13	14	14
4.	15	15	14	13
5.	15	15	15	15
6.	13	12	13	13
7.	14	14	16	15
8.	14	12	14	12
9.	14	13	12	16
10.	16	15	16	15

V. CONCLUSION

In this we paper present some cryptographic attack on our new ULBC. Implimantation cost of our ULBC is lowest among all existing lightweight block cipers. Here we present basic meet in the middle attack on ULBC using splice and cut technique and initial structure technique. After explicitly analyzing splice and cut technique and initial structure technique 15-round meet in the middle (MITM) attack on ULBC is discussed, which is classical one. The data, time and memory complexities of 15-round attack are 2^{64}, 2^{120} and 2^8 respectively.

We have explicitly explained partial matching and complexity analysis in 15-round MITM on ULBC. This article also presents several Differential Fault Attack (DFA) on block cipher ULBC.

Using differential propagation of DDT table and some number of faulty ciphertext, DFA attack logic is designed. We have implemented this fault injection process by C-programming language. By small modification of encryption process of ULBC i.e., injecting nibble fault, we get faulty ciphertext.

Finally, we calculate exact number of required faulty ciphertext to calculate round key. We have checked for 10 different experiments, which ensures that average 57 faulty ciphertexts are needed to retrieve key in second DFA Attack.

REFERENCES

[1] W. Diffie and M. E. Hellman, "Special feature exhaustive Cryptanalysis of the NBS data encryption standard", Computer,10(6) (1977), 74–84.

[2] H. Demirci, A. A. Selçuk, and E. Türe, "A New Meet-in-the-Middle Attack on the IDEA Block Cipher", Springer Berlin heidelberg (2004), 117–129.

[3] X. Dong, J. Hua, S. Sun, Z. Li, X. Wang and L. Hu, "Meet-in-the-Middle Attacks Revisited: Key recovery, Collision and Preimage Attacks", Cryptology ePrint Archive (2021), url: https://eprint.iacr.org/2021/427.

[4] A. Schrottenloher and M. Stevens, "Simplified Modeling of MITM Attacks for Block Ciphers: new (Quantum) Attacks", Cryptology ePrint Archive (2023), url: https://eprint.iacr.org/2023/816.

[5] T. Fuhr and B. Minaud, "Match box meet-in-the-middle attack against KATAN", In Fast Software Encryption: 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014, Revised Selected Papers 21 Springer Berlin Heidelberg, 61–81.

[6] K. Aoki and Y. Sasaki, " Preimage Attacks on One-Block MD4, 63-Step MD5 and More", In: Avanzi, R. M., Keliher, L., Sica, F. (eds) Selected Areas in Cryptography. SAC 2008. Lecture Notes in Computer Science, vol 5381 (2009), 103–119.

[7] A. Bogdanov and C. Rechberger, “A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN”, Cryptology ePrint Archive 2010, 229–240.

[8] A. Abdelkhalek, R. AlTawy, M. Tolba and A. M. Youssef, “Meet-in-the-middle attacks on reduced-round hierocrypt-3”, In International Conference on Cryptology and Information Security in Latin America, August (2015), 187–203.

[9] H. Xiao, L. Wang and J. Chang,“The differential fault analysis on block cipher FeW”, Cybersecurity(2022), url: https://cybersecurity.springeropen.com/articles/10.1186/s42400-022-00130-z.

[10] H. Chen, W. Wu and D. Feng, “ Differential fault analysis on CLEFIA”, In Information and Communications Security: 9th International Conference, ICICS 2007, Zhengzhou, China, December 12-15 (2007) Proceedings 9, 284–295.

[11] H. Luo, W. Chen, X. Ming and Y. Wu. “ General differential fault attack on PRESENT and GIFT cipher with nibble”, IEEE Access (2021), 9, 37697–37706, url: https://ieeexplore.ieee.org/document/9364996.

[12] Y. Lee, J. Kim, J. H. Park and S. Hong, “Differential fault analysis on the block cipher HIGHT”, In Future Information Technology, Application, and Service: FutureTech (2012) Volume 1, Dordrecht: Springer Netherlands, 407–416.

[13] S. Fu, G. Xu, J. Pan, Z. Wang and A. Wang, “ Differential fault attack on ITUbee block cipher”, ACM Transactions on Embedded Computing Systems (TECS) (2016), 16(2), 1–10, url: https://dl.acm.org/doi/abs/10.1145/2967610.

[14] S. Lim, J. Han and D. G. Han,“ Single-byte error-based practical differential fault attack on bit-sliced lightweight block cipher PIPO”, IEEE Access (2022), 10, 67802–67813.

[15] Z. Wang, X. Dong, K. Jia and J. Zhao, “Differential fault attack on KASUMI cipher used in GSM telephony”, Mathematical problems in engineering (2014), 251853.

[16] L. Dong, H. Zhang, L. Zhu, S. Sun, H. Gan and F. Zhang, “Analysis of an optimal fault attack on the LED-64 lightweight cryptosystem”. IEEE Access (2019), 7, 31656–31662.

[17] S. Lim and D. G. Han, “Experimental evaluation of differential fault attack on lightweight block cipher PIPO”, IET Information Security (2023), 17(1), 80–88.

[18] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sin and Y. Todo, “GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption”, In Cryptographic Hardware and Embedded Systems- CHES 2017-19th International Conference, Taipei, Taiwan **2017** (2017), 321–345.

[19] S. Ghorai, N. Dutta and M. Nandi, “ULBC: An Ultra Lightweight Block Cipher”, Journal of advances in Mathematics and Computer Science **38** (2023), 86–100.

[20] D. Boneh, R. A. DeMillo and R. Lipton, “On the importance of eliminating errors in cryptographic computations”, Journal of cryptology **14** (2001), 101–119.

[21] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems”, Advances in Cryptology—CRYPTO’97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17, Springer (1997), 513–525.

[22] C. Giraud, “Dfa on aes”, Advanced Encryption Standard–AES: 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers 4, Springer (2005), 27–41.

[23] N. Bagheri, R. Ebrahimpour and N. Ghaedi, “New differential fault analysis on PRESENT”, EURASIP Journal on Advances in Signal Processing, Springer (2013), 1–10.

[24] H. Luo, W. Chen, X. Ming and Y. Wu, “General differential fault attack on PRESENT and GIFT cipher with nibble”, IEEE Access **9**, 2021, 37697–37706.

[25] J. Zhang, N. Wu, F. Zhou, M. R. Yahya and J. Li, “A novel differential fault analysis on the key schedule of SIMON family”, Electronics **8**, 2019, MDPI, 93,1–13.

[26] A. Canteaut, M. Naya-Plasencia, and B. Vayssiere,“ Sieve-in-the-middle: improved MITM attacks”, In Annual Cryptology Conference (2013, August), Berlin, Heidelberg: Springer Berlin Heidelberg, 222–240.

[27] T. Y. Feng, Y. WEI, J. Shi, J. Cong and Y. Zheng, “Differential fault analysis on lightweight block cipher GIFT”, Journal of Cryptologic Research (2019), 6(3), 324–335.

[28] P. C. Van Oorschot and M. J. Wiener, “Improving implementable meet-in-the-middle attacks by orders of magnitude”, In Annual International Cryptology Conference (1996, August), Berlin, Heidelberg: Springer Berlin Heidelberg, 229–236.