Chapter 4

Advance Attacks

Cipher design should resist basic cryptanalysis which includes linear and differential cryptanalytic. It is also necessary cipher should show good resistance against advance attacks like Biclique attack, Zero correlation attack and Meet In The Middle attack. These kinds of attacks decide the cipher strength. In this section, we are presenting the Biclique, Meet In The Middle and Zero correlation attack which we have mounted on our proposed lightweight designs. Biclique attack is considered to be the theoretical attack. In this section, we have also shown the analysis and the method which we have adopted to mount these attacks.

4.1 Biclique Attack

Biclique attack is applicable on block ciphers as well as on hash functions. Biclique attack was firstly applied on hash functions by Khovratovich et.al. The Biclique attack was applied on block ciphers by Bogdanov et.al. The Biclique attack is based on the meet-in-the-middle-attack (MITM) [21]. Biclique attacks are popularly known for having broken full AES [2][3] and full IDEA[77]. It is the best publicly known single key attack on AES. The computational complexity for AES128, AES192 and AES256 is $2^{126.1}$, $2^{189.7}$ and $2^{254.4}$ respectively [2][3]. A Biclique attack is applied on the target subcipher, and is used to improve the efficiency of computations.

f subcipher that maps the internal state $S$ to ciphertext $C : f_K(S)$. $f$ associates $2^d$ internal states $S_j$ to $2^d$ ciphertext $C_i$ with $2^{2d}$ keys $K[i,j]$.

$$K[i,j] = \begin{pmatrix} K[0,0] & \cdots & K[0,2^d-1] \\ \vdots & \ddots & \vdots \\ K[2^d-1,0] & \cdots & K[2^d-1,2^d-1] \end{pmatrix}$$

The 3-tuples \{Ci, Si, K[i,j]\} is called a d-dimensional biclique if

$$C_i = f_{K[i,j]}(S_j) \quad (1)$$

In other words $K[i,j]$ maps $S_j$ to ciphertext $C_i$, and vice versa.
4.1.1 Procedure to build Biclique

1] The intermediate stage $S_0$, the ciphertext $C_0$ and the key $K[0,0]$ are chosen in such a way that,

$$K[0,0]$$

$$S_0 \xrightarrow{f} C_0$$ (2)

A Key is used to map the internal state and ciphertext by applying the f function.

2] $2^d \times 2^d$ keys are chosen such that

- The 1$^{st}$ keys set is selected that satisfies below differential requirement over function ‘f’ with respect to the base computation:

$$\Delta^k_i$$

$$\Delta^k_i \xrightarrow{f} \Delta_i$$ (3)

- The 2$^{nd}$ keys set is selected that satisfies below differential requirement over function ‘f’ with respect to the base computation:

$$\nabla^k_j$$

$$\nabla^k_j \xrightarrow{f} 0$$ (4)

- The keys are chosen such that the trails of $\Delta_i$ and $\nabla_j$ do not share common active S-boxes i.e. the trails are independent.

3] The trails don’t share common active S-boxes i.e. nonlinear component, so the trails can be combined to get:

$$\Delta^k_i \xrightarrow{f} \Delta_i \oplus \nabla^k_j \xrightarrow{f} 0 = \nabla^k_j \xrightarrow{f} \Delta_i$$ (5)

4] After performing XOR operation on equation (1) and (5), we get:

$$K[0,0] \oplus \Delta^k_i \oplus \nabla^k_j \xrightarrow{f} C_0 \oplus \Delta_i$$ (6)

5] It is a minor matter to see that

$$S_j = S_0 \oplus \nabla_j$$
\[ C_i = C_0 \oplus \Delta_i \]

\[ K[i,j] = K[0,0] \oplus \Delta_i^k \oplus \psi_i \]

After putting these values in equation (6), we will get:

\[ K[i,j] \]

\[ S_j \rightarrow \rightarrow C_i \]

(7)

This is the way how Biclique is constructed on any cipher but there are some practical limitations while constructing a Biclique with the above technique. The permutation layer plays a very important role while constructing the Biclique.

4.1.2 Biclique cryptanalysis procedure

1] The keys are grouped into key subsets of size $2^{2d}$ and the key in the set is indexed as $K[i,j]$ in matrix size $2^{2d} \times 2^{2d}$. The attacker applies MITM attack by splitting cipher into two subcipher $f$ and $g$ mentioned in section 4.1.3. $K[i,0]$ and $K[0,j]$ are the set of keys for each subcipher.

2] The attacker mounts the Biclique for each set of $2^{2d}$ keys. The Biclique of $d$-dimension maps the $S_j$ to $C_i$ using Keys $K[i,j]$. The procedure of building the Biclique is mentioned in section 4.1.2.

3] $2^{2d}$ possible ciphertext $C_i$ goes through the decryption oracle to deliver matching plaintext $P_i$.

4] The attacker chose $S_j$ and $P_i$ and performed a MITM attack over $f$ and $g$ mentioned in section 4.1.3.

4.1.3 Meet-in-the-middle attack (MITM)

The $2^d$ intermediary values from plaintext $P_i$

\[ P_i \rightarrow \rightarrow v_i \]

(8)

The $2^d$ intermediary values from ciphertext $S_j$
A matching pair \( v_i = v_j \) gives a key candidate \( K[i,j] \).

### 4.1.4 Computational Complexity or Complexity of Key Recovery

The complexity of the Biclique attack is calculated as follows:

\[
C_{\text{full}} = 2^{N-2d} [C_{\text{biclique}} + C_{\text{precomp}} + C_{\text{recomp}} + C_{\text{falsepos}}]
\]

Where

- \( C_{\text{biclique}} \) is the complexity of creating a single Biclique.
- \( C_{\text{precomp}} \) is the complexity of the pre-computation.
- \( C_{\text{recomp}} \) is the complexity of the re-computation. \( C_{\text{recomp}} \) is depends on the permutation properties of the cipher.
- \( C_{\text{falsepos}} \) is the complexity generated by false positives.
- \( N \) is length of key and \( d \) is the dimension of the Biclique attack.

### 4.1.5 Biclique Attack on PRESENT Cipher

PRESENT [8][9] is a SPN network cipher that supports 64-bit block size and 80/128-bit key size. PRESENT-80 accepts a 64-bit plaintext \( P = (P_{63}, P_{62}, \cdots, P_0) \) and the 80-bit key \( K = (R_{79}, R_{78}, \cdots, R_0) \) as input and generates a 64-bit ciphertext \( C = (C_{63}, C_{62}, \cdots, C_0) \). Similarly, PRESENT-128 accepts a 64-bit plaintext \( P = (P_{63}, P_{62}, \cdots, P_0) \) and the 128-bit secret key \( K = (R_{127}, R_{126}, \cdots, R_0) \) as input values and generates a 64-bit ciphertext \( C = (C_{63}, C_{62}, \cdots, C_0) \). It has a total of 31 rounds, composed of three main operations which are mentioned below:

1] addRoundKey,

2] sBoxLayer

3] pLayer

addRoundKey operation performs bit wise XOR operation between the current state output and the current roundsubkey \( (R_k) \) where \( 1 \leq i \leq 31 \). sBoxLayer is a non-linear layer that performs a substitution operation. The substitution operation is performed by using the 4 × 4 S-box mentioned in Table 4.1. This layer provides the confusion property to the cipher [85].
Table 4.1 S-box of the PRESENT cipher

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[b]</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The pLayer performs bitwise permutation to provide the diffusion property to the cipher. Bitwise permutation is given in Table 4.2. In the end, $R_{k32}$ is used as a post-whitening key.

Table 4.2 P-Layer of the PRESENT cipher

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

1. **80-bit Key Scheduling**

Register $K^i$ stores the user key and it is represented as $R_{k79}, R_{k78}, \ldots, R_{k1}, R_{k0}$. For $i^{th}$ round, the 64-bit round key $RK^i = RK_{63}RK_{62} \ldots RK_0$ consists of the 64 leftmost bits of register $K^i$

$K^i = R_{k79}, R_{k78} \ldots R_{k1}, R_{k0}$

$RK^i = RK_{63}RK_{62} \ldots RK_0 = R_{k79}, R_{k78} \ldots, R_{k16}$

Register $K^i = R_{k79}, R_{k78} \ldots R_{k1}, R_{k0}$ is updated as follows

1. $[R_{k79}R_{k78} \ldots R_{k1}R_{k0}] = [R_{k18}R_{k17} \ldots R_{k20}R_{k19}]$

2. $[R_{k79}R_{k78}R_{k77}R_{k76}] = S[R_{k79}R_{k78}R_{k77}R_{k76}]$

3. $[R_{k19}R_{k18}R_{k17}R_{k16}R_{k15}] = [R_{k19}R_{k18}R_{k17}R_{k16}R_{k15}] \oplus [\text{round-counter-i}]$

The operations that are used to update the whole 80-bit key are:

1] Left circular shift by 61-bit.
2] Nonlinear layer operation is performed on \( R_{k_79} R_{k_78} R_{k_77} R_{k_76} \).

3] The round counters value ‘i’ is XORed with \( R_{k_19} R_{k_18} R_{k_17} R_{k_16} R_{k_15} \).

2. 128-bit Key Scheduling

Register \( K^i \) stores the user key and it is represented as \( R_{k_127}, R_{k_126}, \ldots, R_{k_1}, R_{k_0} \). For \( i^{th} \) round, the 64-bit round key \( R_{K^i} = R_{K_127} R_{K_126} \ldots R_{K_{164}} \) consists of the 64 leftmost bits of register \( K^i \).

\[ K^i = R_{k_127}, R_{k_126}, \ldots, R_{k_1}, R_{k_0} \]

\[ R_{K^i} = R_{K_63} R_{K_62} \ldots R_{K_0} = R_{k_127}, R_{k_126}, \ldots, R_{k_{64}} \]

128-bit key register \( K^i \) is updated as follows:

1. \[ [R_{k_{127}} R_{k_{126}} \ldots R_{k_1} R_{k_0}] = [R_{k_{66}} R_{k_{65}} \ldots R_{k_{68}} R_{k_{67}}] \]

2. \[ [R_{k_{127}} R_{k_{126}} R_{k_{125}} R_{k_{124}}] = S[R_{k_{127}} R_{k_{126}} R_{k_{125}} R_{k_{124}}] \]

3. \[ [R_{k_{123}} R_{k_{122}} R_{k_{121}} R_{k_{120}}] = S[R_{k_{123}} R_{k_{122}} R_{k_{121}} R_{k_{120}}] \]

4. \[ [R_{k_{66}} R_{k_{65}} R_{k_{64}} R_{k_{63}} R_{k_{62}}] = [R_{k_{66}} R_{k_{65}} R_{k_{64}} R_{k_{63}} R_{k_{62}}] \oplus [\text{round-counter-i}] \]

The operations that are used to update the complete 80-bit key are:

1] Left circular shift by 61-bit.

2] Nonlinear layer operation is performed on \( R_{k_{127}} R_{k_{126}} \ldots R_{k_{121}} R_{k_{120}} \).

3] The round counters value ‘i’ is XORed with \( R_{k_{66}} R_{k_{65}} R_{k_{64}} R_{k_{63}} R_{k_{62}} \).

4.1.6 Biclique cryptanalysis of PRESENT-80

In [78], the author explains a Biclique attack on PRESENT-80 with 4 dimensions from rounds 28 to 31. Partial secret keys used in \{R_{K^{28}}, R_{K^{29}}, R_{K^{30}}, R_{K^{31}}\} are:

\[ R_{K^{28}} = (R_{k_{51}}, R_{k_{50}}, \ldots, R_{k_0}, R_{k_{79}}, R_{k_{78}}, \ldots, R_{k_{68}}) \]

\[ R_{K^{29}} = (R_{k_{70}}, R_{k_{69}}, \ldots, R_{k_7}) \]

\[ R_{K^{30}} = (R_{k_9}, R_{k_8}, \ldots, R_{k_0}, R_{k_{79}}, R_{k_{78}}, \ldots, R_{k_{26}}) \]

\[ R_{K^{31}} = (R_{k_{28}}, R_{k_{27}}, \ldots, R_{k_0}, R_{k_{79}}, R_{k_{78}}, \ldots, R_{k_{45}}) \]
(Rk_{58}, Rk_{57}, Rk_{56}, Rk_{55}) and (Rk_{37}, Rk_{36}, Rk_{35}, Rk_{34}) gives Bicliques for the attack on the PRESENT-80 [78]. Subkeys (Rk_{58}, Rk_{57}, Rk_{56}, Rk_{55}) are considered to construct the $\Delta_i$-differential and (Rk_{37}, Rk_{36}, Rk_{35}, Rk_{34}) subkeys are considered to construct the $\nabla_j$-differential. Let, f be a sub-cipher from round 28 to round 31. As can be seen from Fig. 4.1, the $\Delta_i$-differential affects the 23-bits of the ciphertext. So, the data complexity does not exceed $2^{23}$. This figure is taken from paper [78].

The attack procedure on the full PRESENT-80 is as follows:

1. Pre-computation
1] The attacker computes the most significant 4-bits of the output value of round 15 by using \( P_i \) and \( K[i,0] \) in the forward direction and stores it as \( v_i \). [78].

2] The attacker computes the most significant 4-bits of the input value of round 16 by using \( S_j \) and \( K[0,j] \) in the forward direction and stores it as \( v_j \).

2. Computation in the backward direction
   An attacker would calculate \( v_j \) from \( S_j \) and \( K[i,j] \) for all values of \( i \) and \( j \), and stores them in memory. Re-computations are accomplished according to the red/blue lines shown in Fig.4.2 b). This fig is taken from the paper [78].

3. Computation in the forward direction
   An attacker would calculate \( v_j \) from \( P_i \) and \( K[i,j] \) for all values of \( i \) and \( j \), and stores them in memory. Re-computations are accomplished according to the red/blue lines shown in Fig.4.2 a). This figure is taken from the paper [78].

![Fig.4.2 a): Re-computation in forward direction](image)
Fig. 4.2 b): Re-computation in the backward direction

The total computational complexity of PRESENT-80 is computed as follows:

\[ C_{\text{total}} = 2^{N-2d} (C_{\text{biclique}} + C_{\text{precomp}} + C_{\text{recomp}} + C_{\text{falsepos}}) \]

Where \( N = 80 \) and \( d = 4 \).

1] \( C_{\text{biclique}} = 2^{d+1} \) (Number of rounds in Biclique / total no of rounds)

\[ C_{\text{biclique}} = 2^{4.63} \approx 2^{4+1} \times (3/31) \]

2] \( C_{\text{precomp}} = 2^d \) (Number of rounds in pre-computation / total no of rounds)

\[ C_{\text{precomp}} = 2^3 \approx 2^4 \times (28/31) \]

3] \( C_{\text{recomp}} = 2^{2d} \) (Number of active S-box in pre-computation / total no of active S-box)

\[ C_{\text{recomp}} = 2^7 \approx 2^{2.4} \times (22.31/31) \]

4] \( C_{\text{falsepos}} = 2^{2d - \text{no. of matching bits}} = 2^4 \approx 2^{(2\times4)-4} \)

\[ C_{\text{total}} = 2^{79.86} \approx 2^{80-2.4} (2^{1.63} + 2^{3.85} + 2^{7.53} + 2^4) \]
4.2 Zero-Correlation Linear Cryptanalysis of Block Cipher

Linear cryptanalysis and differential are important techniques to compute the security of block ciphers [20]. Zero-correlation linear cryptanalysis is a novel extension of linear cryptanalysis. Zero-correlation linear cryptanalysis is based on linear approximations with a correlation value of exactly zero. Zero-correlation linear attack [22] is developed by Bogdanov et al. in 2012. Zero-correlation linear cryptanalysis is similar to the impossible differential cryptanalysis in the field of linear cryptanalysis.

Consider a function $f: \mathbb{F}_n^2 \rightarrow \mathbb{F}_m^2$ and let the input of the function be $a \in \mathbb{F}_n^2$. A linear approximation with an input mask $m$ and an output mask $n$,

$$ a \rightarrow m \cdot x \oplus n \cdot f(a). $$

The probability of linear approximation is

$$ p(m;n) = Pr(m \cdot x \oplus n \cdot f(a) = 0) = Pr(a) = 0 $$

And the correlation can be given as

$$ cf(m;n) = 2p(m;n) - 1. $$

Linear cryptanalysis is based on linear approximations with correlation away from zero. And the number of known plaintexts required in linear cryptanalysis is inversely proportional to square of correlation. While in Zero-correlation linear cryptanalysis is based on linear approximations with correlation zero for all sub keys.

4.2.1 Extension of Linear Cryptanalysis

Numerous extensions of linear cryptanalysis have been presented which usually describe linear approximations with high correlation:

- Multiple linear approximations with the same key mask [79]. (Kaliski and Robshaw 1994)
- Several linear approximations with the same input and output masks. (Nyberg 1994)
- Multiple independent linear approximations [80]. (Biryukov et al. 2004)
- Multidimensional linear approximations [81]. (Hermelin et al. 2009)
1. Zero-Correlation Linear Cryptanalysis of Feistel Type Block Cipher

Feistel ciphers are based on three basic operations:

1. XOR-operation,
2. Branching operation,
3. Key-dependent F-function φ.

Fig. 4.3 represents balanced a Feistel structure and Fig. 4.4 represents basic operations in the Feistel structure.

Linear approximations over these operations obey three major rules

Lemma 1] XOR approximation
Either the three linear selection patterns at an XOR ⊕ are equal or the correlation over ⊕ is exactly zero.

Lemma 2] Branching approximation
Either the three linear selection patterns at a branching point • sum up to 0 or the correlation over • is exactly zero.
Lemma 3] Permutation approximation

Over a permutation $\varphi$, if the input and output selection patterns are neither both zero nor both nonzero, the correlation over $\varphi$ is exactly zero.

The understanding of Lemma 1 to 3 is as follows. If in a linear trail

1. the selection patterns at XOR $\oplus$ are not equal,
2. the selection patterns at branching point • do not sum up to zero or
3. the selection patterns at permutation F are neither both zero nor both nonzero,

Then this linear trail contains an incompatible pair of adjacent selection patterns.

2. Zero-Correlation Linear Hulls for Feistel Ciphers

If the F-functions of the Feistel-type block cipher is invertible, then the linear hulls have zero correlation for $a \neq 0$.

$(a,0) \not\leftrightarrow (0,a)$ for 5 rounds of balanced Feistel ciphers,

Fig.4.5: Zero-correlation linear hull $(a,0) \not\leftrightarrow (0,a)$ over 5 rounds of balanced Feistel cipher

Where, a and b are nonzero values.

4.2.2 The Matrix Method

Numerous tools have been described for finding the statistical distinguisher. These tools help to examine algorithms systematically. The Matrix method is one of them; it is based on the “Meet-In-The-Middle” approach to find impossible differential characteristics [21]. Meet-In-The-Middle technique computes impossible differential characteristics with two differential paths having a probability of one that causes a contradiction in the middle. Linear approximation trails of input and output masks are followed in the middle rounds and analyzed with no linear characteristics with non-zero-correlation. The same matrix method is also used for finding zero-correlation linear approximation [22].

65
1. Matrix Method for Finding Linear Approximation with Correlation Zero

The linear masks applied to the words can be of the following five types:

1. Zero mask denoted by 0,
2. An arbitrary non-zero mask denoted by 0,
3. Non-zero mask with a fixed value a,
4. The exclusive-or of a fixed non-zero mask a and an arbitrary non-zero mask, denoted by a,
5. Any other mask is denoted by *.

Table 4.3 Arithmetic Rules Multiplication by 0, 1 and 1F

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

The matrix shows how a linear mask of each output word is affected by the linear mask of an input word. Arithmetic rules for multiplication and addition are given in the Table 4.3 and 4.4, respectively.

Table 4.4 Arithmetic Rules Addition between Two masks

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>0</th>
<th>a</th>
<th>a</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>a</td>
<td>a</td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>a+b</td>
<td>a+b</td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
4.2.3 Zero-Correlation Linear Approximation for 14-rounds of LBlock

Description of LBlock

Following notations are used for describing the algorithm of the Lblock:

1. $\text{Ski} \mapsto$ 32-bit round key
2. $\ll i \mapsto$ i-bit left circular shift
3. $|| \mapsto$ concatenation of two binary strings
4. $\text{Li}||\text{Ri} \mapsto$ the output of the i-round of LBlock

The Lblock is a Feistel block cipher with 32 rounds. The Lblock supports a block size of 64-bit and a key size of 80-bit. The Lblock uses eight different S-boxes and permutation is based on nibble wise. Single step of Lblock is represented in Fig. 4.5 and its F-function is represented in Fig. 4.6.

![Fig. 4.6: Single round of Lblock](image-url)
Let, $P$ be a 64-bit plaintext such that $L_0 || R_0$. The encryption can be given as

For $i = 1, 2, \cdots, 31$, do

$$R_i = L_{i-1}$$

$$L_i = F(L_{i-1}, SK_i) \oplus (R_{i-1} \ll 8)$$

$L_{32} = L_{31},$

$R_{32} = F(L_{31}, SK_{32}) \oplus (R_{31} \ll 8)$

$C = L_{32} || R_{32}.$

**Key schedule**

A user supplied 80-bit key is stored in the key register $K$. For the $i^{th}$ round, the left most 32-bit key is extracted from the key register and stored as the round key $SK_i$. Then the key register is updated as follows

For $i = 1, 2, \cdots, 31$, update the key register $K$ as follows:

1. $K \ll\ll 29$
2. \([k_{79} \ k_{78} \ k_{77} \ k_{76}] = s_9[k_{79} \ k_{78} \ k_{77} \ k_{76}]\)

3. \([k_{75} \ k_{74} \ k_{73} \ k_{72}] = s_8[k_{75} \ k_{74} \ k_{73} \ k_{72}]\)

4. \([k_{50}k_{49}k_{48}k_{47}k_{46}] \oplus [i]_2\)

5. Output the leftmost 32 bits of current content of register K as round subkey \(SK_{i+1}\).

The linear approximation has zero correlation if the input mask has one non-zero nibble in \(L_r\) and out mask has non-zero nibble in \(R_{r+14}\). For example \((000a0000||00000000) \rightarrow (00000000||0b000000)\), where a and b are non-zero values. The contradiction occurs at round 7.

Fig. 4.8: Zero-correlation linear approximation for 14-round LBlock
In this section, we have presented the details of the advance attacks which we have mounted on our proposed cipher designs. Specifically Biclique, Meet In The Middle, and zero correlation attack is studied in detail and we have successfully shown the cipher design resistance against these types of attacks.