Non-isomorphic Biclique Cryptanalysis and Its Application to Full-Round mCrypton

Mohsen Shakiba [a], Mohammad Dakhilalian [a], Hamid Mala [b]

[a] *Cryptography & System Security Research Laboratory, Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran*

m.shakiba@ec.iut.ac.ir , mdalian@cc.iut.ac.ir

[b] *Department of Information Technology, University of Isfahan, Isfahan, Iran*

h.mala@eng.ui.ac.ir

Abstract Biclique attack, is a new cryptanalytic technique which brings new tools from the area of hash functions to the area of block cipher cryptanalysis. Till now, this technique is the only one able to analyze the full-round AES cipher in a single key scenario. In this paper, we introduce *non-isomorphic biclique* attack, a modified version of the original biclique attack. In this attack we obtain isomorphic groups of bicliques, each group contains several non-isomorphic bicliques of different dimensions. Actually, these bicliques are the results of an asymmetric key partitioning which is done according to two sets of key differences. Using this technique it is possible to get a chance to expand the length of bicliques or mount an attack with less data complexity. We found out the lightweight block cipher mCrypton is an appropriate candidate to be analyzed with this technique and bicliques up to five rounds can be constructed for this block cipher. Furthermore, we use two additional minor techniques, including *pre-computation/re-computation in the bicliques construction* and *early abort technique* in the matching stage, as well as a property observed in the diffusion layer of mCrypton to obtain more improvements for the complexity of our attacks on full-round mCrypton-96 and mCrypton-128.

Keywords Biclique cryptanalysis, Asymmetric key partitioning, Non-isomorphic bicliques, Block ciphers, mCrypton

1. Introduction

After the introduction of the meet-in-the-middle (MITM) attack by Diffie and Hellman in 1977 [6], the area of modern block ciphers cryptanalysis saw another milestone in the early 1990s by the invention of differential [2] and liner [16] attacks. Then, the advanced encryption standard competition, 1997-2001, accelerated the progress of design and cryptanalysis of block ciphers. However, this progress is more evident in the field of design than the field of analysis. Indeed, most of important cryptanalysis methods for the block ciphers such as

impossible differential, differential-linear, square and boomerang attacks have been proposed during the period of the AES competition. On the other hand, thanks to the SHA-3 competition, cryptanalysis of hash functions has grown faster than the block ciphers. This fact has encouraged the adoption of some hash cryptanalysis techniques to enrich the field of block cipher cryptanalysis. Biclique attack, although may be considered as an extension of the meet-in-the-middle attack, is one of these kinds of brilliant techniques which brings new tools from the area of hash functions to the area of block cipher cryptanalysis [10]. This attack was first introduced by some cryptanalysis results on the AES [3]. The authors present two paradigms for key recovery with bicliques, including the *long-biclique* and *independent-biclique*. The obtained results show the best known single-key attacks on the variants of AES. Since its introduction, this technique has been applied to many block ciphers such as SQUARE [14], Piccolo [17], ARIA-256 [4], HIGHT [7], TWINE [5], Present and LED [1]. After that, the notion of *narrow-biclique* was presented in [9] to analyze the long-lasting block cipher IDEA. In fact, they extend the independent-biclique framework to allow for lower data complexity requirements by using available degrees of freedom for limiting the diffusion in spite of high dimension.

In this paper, we introduce a modified version of the original biclique attack. In this technique we define isomorphic groups of bicliques, each of them consists of several non-isomorphic biclques. Based on this fact we call this technique *non-isomorphic bicliquie* attack. Our experiments show that this method in combination with key schedule properties of some block ciphers like Crypton [11] and mCrypton [12] leads to a longer independent bicliques. As it is known, mCrypton is derived from Crypton and they have the same top level structure. But due to its slightly more complicated key scheduling, we selected mCrypton to study the application of non-isomorphic biclique attack. Moreover, two additional techniques, one for increasing the speed of biclique construction and the other, early abort technique, for reducing the complexity of matching check are exploited to reduce the overall time complexity of the attack procedure.

The designers of mCrypton show that this cipher is secure against differential and linear attacks [12]. So far, some attacks have been published on this cipher, including a related-key rectangle attack on 8 rounds of mCrypton-128 [8] and related-key impossible differential attacks on 9 rounds of mCrypton-96 and mCrypton-128 [15]. However, the attack proposed in this paper is the first full-round attack on this block cipher. Obtained results include two attacks on mCrypton-96 using a 4-round and 5-round independent bicliques with the computational complexity $2^{94.09}$ and $2^{93.75}$ encryptions, respectively. Also, we construct a 4-round biclique to mount an attack on mCrypton-128 with the computational complexity $2^{126.05}$ encryptions. As it will be discussed in Section 4.4, in this paper we have used a stricter criterion for the complexity estimation instead of typical criterion which is dependent only to the number of active S-boxes. Thus, considering the typical criterion these computational complexities are decreased at least 0.15 in the exponentials.

This paper is organized as follows: Section 2 provides a brief description of mCrypton and the required preliminaries. A concise review of the biclique attack along with a preview of our contribution is given in Section 3. In Subsection 4.1 we introduce a 4-round biclique for the mCrypton-96. Then Subsection 4.2 describes the asymmetric key partitioning method. For this purpose, first, we define the concept of minimal space generator and a method for its construction. Then a general form for the asymmetric key partitioning and the topology of

constructed bicliques is introduced. Then we customize this general form for the non-overlapping sets. Subsections 4.3 and 4.4 are devoted to the attack procedure and its complexity, respectively. Another attack on mCrypton-96 based on a five-round biclique, is presented in Subsection 4.5. A similar procedure is applied for mCrypton-128 in Section 5. Finally, we conclude the paper in Section 6.

2. Preliminaries and a Brief Description of mCrypton

mCrypton is a 64-bit lightweight block cipher designed for use in low-cost and resource-constrained devices such as RFID tags and sensors in wireless sensor networks. As it is known, this cipher is a redesigned compact version of the Crypton cipher. mCrypton processes 64-bit data block, consisting of 16 nibbles, by representing it as a 4×4 matrix of nibbles as follows:

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix}$$

mCrypton encryption/decryption, independent of its key size, has a 12-round Substitution-Permutation structure, where each round consists of four transformations as follows:

Non-linear Substitution γ: consists of nibble-wise substitution, using four 4-bit S-boxes $S_i, i = 0,1,2,3$. Thus, the nibble located in row $0 \le i \le 3$ and column $0 \le j \le 3$ in array A ($a_{4\times i+j}$) is passed through the S-box $S_{(i+j) \bmod 4}$.

Bit Permutation π: mixes column $0 \le i \le 3$ of array A by bit-wise column permutation π_i. In column $0 \le i \le 3$, for an input column $a = (a_0, a_1, a_2, a_3)^t$ and its corresponding output column $b = \pi_i(a) = (b_0, b_1, b_2, b_3)^t$, we have $b_j = \oplus_{k=0}^{3}(a_k \bullet m_{(i+j+k) \bmod 4})$, $0 \le j \le 3$, where $m_0 = 1110$, $m_1 = 1101$, $m_2 = 1011$ and $m_3 = 0111$ are masking nibbles and '$\bullet$' is the bit-wise AND operation. π transformation is an involution (i.e. $\pi = \pi^{-1}$) and has a differential branch number of 4.

Column-To-Row Transposition τ: moves the nibble in the position (i, j) to the position (j, i).

Key Addition σ: For a round key $K_r = (K_r[0], K_r[1], K_r[2], K_r[3])$, it is a simple bit-wise XOR operation. $K_r[i], i = 0,1,2,3$ is the value of round key in the *i*-th row.

According to the above transformation, *r*-th round of mCrypton, $1 \le r \le 12$, is applied to the input intermediate value *X* by $\sigma_{K_r} \bullet \tau \bullet \pi \bullet \gamma(X)$. An initial *Key Addition* transformation (σ_{K_0}) and a final operation $\tau \bullet \pi \bullet \tau$ are also performed before the first round and after last round, respectively. In the *r*-th round, the

intermediate values after application of transformations γ , π , τ and σ are denoted by x_r^{γ} , x_r^{π} , x_r^{τ} and x_r^{σ} , respectively. Also, the input and the output of *r*-th round is represented by x_r^I and x_r^O , respectively.

Key scheduling of mcrypton-96: the 96-bit internal register $U=(U[0],U[1],U[2],U[3],U[4],U[5])$ is first initialized with the 96-bit user key, then round keys $K_r, 0\le r\le 12$ are computed consecutively as follow (in the following equations, $C[r]$, $0\le r\le 12$, are some constants and *S* denotes a nibble-wise S-box operation):

$$T \leftarrow S(U[0])\oplus C[r], T_i \leftarrow T \bullet M_i$$
$$(i=0,1,2,3, M_0=0xf\ 000, M_1=0x\ 0f\ 00, M_2=0x\ 00f\ 0, M_3=0x\ 000f)$$
$$K_r=(U[1]\oplus T_0, U[2]\oplus T_1, U[3]\oplus T_2, U[4]\oplus T_3)$$
$$U \leftarrow (U[5], U[0]<<3, U[1], U[2], U[3]<<8, U[4])$$

Key scheduling of mcrypton-128: the 128-bit internal register $U=(U[0],U[1],U[2],U[3],U[4],U[5],U[6],U[7])$ is first initialized with the user key, then round keys $K_r, 0\le r\le 12$ are computed consecutively same as the 96-bit version, except that the updating stage is performed as follow:

$$U \leftarrow (U[5],U[6],U[7],U[0]<<3,U[1],U[2],U[3],U[4]<<8).$$

In the following we introduce a property of bit permutation π transformation in mCrypton cipher:

Property 1 Assume that for column i, $0\le i\le 3$ and its four-nibbles input/output a and b we have $b=\pi_i(a)$. Also for column i', $0\le i'\le 3$ and its four-nibbles input/output a' and b' we have $b'=\pi_{i'}(a')$. Obviously, there is $0\le f\le 3$ which $i'=(i+f) \bmod 4$. Thus, if a' is a circular shift of a by s nibbles, i.e. $a'=a<<s$, $0\le s\le 3$, then, the nibbles of b' are also a circular shift of nibbles of b such that:

$$b'=\begin{cases} b>>(s-f) & (s-f)\ge 0 \\ b<<(f-s) & (s-f)<0 \end{cases}.$$

Proof. Suppose $a'=a<<s, 0\le s\le 3$ then $a'_k=a_{(k+s) \bmod 4}, 0\le k\le 3$. Based on π operation, the value of the j th nibble, $0\le j\le 3$ of b is $b_j=\oplus_{k=0}^{3}(a_k \bullet m_{(i+j+k) \bmod 4})$. In the same way, the j' th nibble, $0\le j'\le 3$ of b', where $j'=(j+s-f) \bmod 4$, is:

$$b'_{j'}=\oplus_{k=0}^{3}(a'_k \bullet m_{(i'+j'+k) \bmod 4})=\oplus_{k=0}^{3}(a_{(k+s) \bmod 4} \bullet m_{((i+f)+(j+s-f)+k) \bmod 4}).$$

Assume $k'=(k+s) \bmod 4$, then $b'_{j'}$ could be rewritten as:

$$b'_{j'}=\oplus_{k'=0}^{3}(a_{k'} \bullet m_{(i+j+(s-f)+k'+(f-s)) \bmod 4})=\oplus_{k'=0}^{3}(a_{k'} \bullet m_{(i+j+k') \bmod 4})=b_j.$$ Hence, $b_j=b'_{(j+s-f) \bmod 4}$. So for the whole column we have:

$$b' = \begin{cases} b >> (s-f) & , (s-f) \geq 0 \\ b << (f-s) & , (s-f) < 0 \end{cases}.\square$$

As it was mentioned, the π operation is an involution so the above property holds for the inverse of π operation. Also, π is linear, so the same property holds for the differentials in the input and output of π.

Notations. Throughout this paper we use the following notations:

$\{A\}$: means that A is a collection of sets.

$A_{i,[j_1,j_2,\ldots]}$: indicates the bits $j_1, j_2, \ldots$ of i-th element of A.

$\{A\}_{i,[j_1,j_2,\ldots]}$: indicates the elements with indexes $j_1, j_2, \ldots$ of i-th set of $\{A\}$.

3. Biclique Cryptanalysis and Our Contribution

In this section, first, we review the concept of biclique cryptanalysis as it was introduced in [3] unless we describe biclique construction at the plaintext side instead of ciphertext side. Also we describe this method with an emphasis on independent bicliques. Then, we introduce our contribution in the biclique cryptanalysis.

Biclique Cryptanalysis. Assume block cipher E is composed of three sub-ciphers as $E_K(P) = f_K \circ g_K \circ h_K(P)$. Also the intermediate value $h_K(P)$ is denoted by S. Suppose h connects 2^d plaintexts P_i, $0 \leq i < 2^d$, to 2^d intermediate states S_j, $0 \leq j < 2^d$, with 2^{2d} keys $K[i,j]$. Now, set of P_is, S_js and $K[i,j]$s is called a biclique of dimension d if we can write $S_j = h_{K[i,j]}(P_i)$, $0 \leq i < 2^d$, $0 \leq j < 2^d$.
For performing a biclique attack on the block cipher E, at the first step, the whole key space is partitioned into 2^{k-2d} groups each of 2^{2d} keys. Then for each key group, according to the definition of a biclique, the biclique attack is performed based on the following three major steps:

- Biclique Construction: Build a structure of 2^d plaintexts and 2^d intermediate values such that for each i and j, $0 \leq i < 2^d$, $0 \leq j < 2^d$, the relation $S_j = h_{K[i,j]}(P_i)$ is satisfied.

- Pre-computation: Fix some matching nibbles in the output of sub-cipher g. Ask for the encryption of plaintexts P_i s to obtain the corresponding ciphertexts C_i s. Then compute $f^{-1}_{K[i,0]}(C_i)$ and store the intermediate values. Also, for intermediate values S_j s, compute $g_{K[0,j]}(S_j)$ and store the intermediate values. To reduce the data complexity, some plaintexts are reused in different groups.

- Matching check with re-computation: According to the stored values in the previous stage, ask for re-computation of the matching nibbles for each S_j under $K[i,0]$ and for each P_i under $K[0,j]$. If a $K[i,j]$ be the correct user key, then

obviously, it maps S_j to P_i in the matching nibbles. So, the adversary checks such keys by a trial encryption with a valid plaintext/ciphertext pair. In the other hand if a $K[i,j]$ leads to different values in the matching nibbles, then it is surely incorrect.

According to the definition of biclique, a *d*-dimensional biclique needs to establish 2^{2d} relationships simultaneously. Bogdanov et al. proposes an approach to find a *d*-dimensional biclique from independent related key differentials [3]. The result of this technique says that if we have two related key $\Delta-differential$ and $\nabla-differential$, which share no active non-linear components (S-boxes); $\Delta-differential$ maps an input difference Δ_i to the output difference 0 under key difference Δ_i^K, and $\nabla-differential$ maps the input difference 0 to an output difference ∇_j under key difference ∇_j^K, then an input difference Δ_i maps to an output difference ∇_j under key difference $\Delta_i^K \oplus \nabla_j^K$ for each *i* and *j*.

Our contributiuon. The main structure of the *non-isomorphic biclique* attack is the same as original biclique attack. The most notable difference with the previous approach is the way we partition the key space. Actually, in the original biclique method the key space is partitioned uniformly, so all of the bicliques are of the same dimension. While, in the *non-isomorphic biclique attack*, at first we select two arbitrary sets of key differences which could lead to the bicliques with more length, then an asymmetric (non-uniform) key partitioning is done according to these differences sets. Thus, we obtain isomorphic groups of bicliques, called *biclique groups*, which in each of them there are several non-isomorphic biclques. Note that, this does not mean that all of the bicliques in a biclique group are non-isomorph and it could be some bicliques in a group with the same dimensions. Fig. 1 shows the overall topology of a *biclique group* consists of *m* non-isomorphic bicliques which *m* is smaller or equal to the number of bicliques in the *biclique group*. For a biclique with d_1^i and d_1^i vertices in its two segments, we define its dimension with the pair (d_1^i, d_2^i) and it is expected for such a biclique to cover at most $d_1^i \times d_2^i$ new distinct keys of the key space (if two key differences sets be non-overlapped then such a biclique covers exact $d_1^i \times d_2^i$ new distinct keys). Through this method, depending on the key scheduling algorithm, it is possible to get a chance to expand the length of bicliques or less data complexity. In Section 4.2 we introduce the asymmetric key partitioning method in its general form and also we customize it for non-overlapping key differences sets. Moreover, we improve time complexity of the attack by another two minor techniques including *pre-computation/re-computation* in the biclique construction and *early abort* technique in the matching check stage.

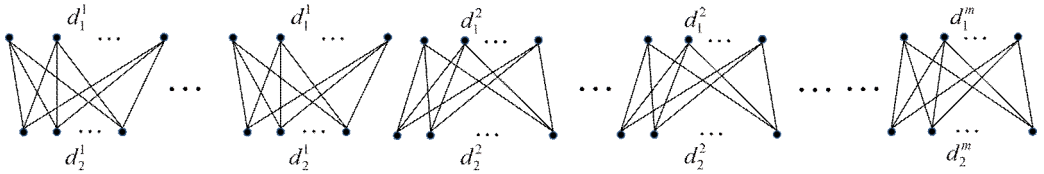

Fig. 1. Overall topology of a *biclique group* consists of *m* non-isomorphic bicliques.

4. Non-isomorphic biclique cryptanalysis of mCrypton-96

In this section, an attack on full-round mCrypton-96 is presented. For this purpose, first, we construct a 4-round independent biclique for the initial rounds and then we introduce the asymmetric key partitioning approach with more details. Next, the attack procedure and complexity analysis are proposed. Finally, we introduce another version of the attack using a 5-round biclique.

4.1. Four round biclique

To construct an independent biclique, we need a more accurate view of the key schedule of mCrypton-96. Table 1 shows the changes of the internal value of $U=(U[0],U[1],U[2],U[3],U[4],U[5])$ register through the 12 rounds of the encryption process.

Table. 1 Updated register ***U*** through the rounds of mCrypton-96

r = 0	U[0]	U[1]	U[2]	U[3]	U[4]	U[5]
r = 1	U[5]	U[0] << 3	U[1]	U[2]	U[3] << 8	U[4]
r = 2	U[4]	U[5] << 3	U[0] << 3	U[1]	U[2] << 8	U[3] << 8
r = 3	U[3] << 8	U[4] << 3	U[5] << 3	U[0] << 3	U[1] << 8	U[2] << 8
r = 4	U[2] << 8	U[3] << 11	U[4] << 3	U[5] << 3	U[0] << 11	U[1] << 8
r = 5	U[1] << 8	U[2] << 11	U[3] << 11	U[4] << 3	U[5] << 11	U[0] << 11
r = 6	U[0] << 11	U[1] << 11	U[2] << 11	U[3] << 11	U[4] << 11	U[5] << 11
r = 7	U[5] << 11	U[0] << 14	U[1] << 11	U[2] << 11	U[3] << 3	U[4] << 11
r = 8	U[4] << 11	U[5] << 14	U[0] << 14	U[1] << 11	U[2] << 3	U[3] << 3
r = 9	U[3] << 3	U[4] << 14	U[5] << 14	U[0] << 14	U[1] << 3	U[2] << 3
r = 10	U[2] << 3	U[3] << 6	U[4] << 14	U[5] << 14	U[0] << 6	U[1] << 3
r = 11	U[1] << 3	U[2] << 6	U[3] << 6	U[4] << 14	U[5] << 6	U[0] << 6
r = 12	U[0] << 6	U[1] << 6	U[2] << 6	U[3] << 6	U[4] << 6	U[5] << 6

Now, we define two sets of key differences Δ_i^K and ∇_j^K, which are used to form a 4-round independent biclique. Both of these sets are directly defined on the user key which is the initial value of register ***U***. The first set includes 61 distinct 96-bits differences as follows:

$$\underset{0\le i\le 60}{\Delta_i^K}:\begin{cases}\Delta U[k]=0, & k=0,1,3,4,5\\ \Delta U[2]=a_i, & \pi_3(a_i<<8)=[0,0,0,0]or[0,0,0,t]or[0,0,t,0]or[0,t,0,0]or[t,0,0,0],\ t\neq 0\end{cases}$$

As it can be seen, in each of these differences, $\Delta U[0]$, $\Delta U[1]$, $\Delta U[3]$, $\Delta U[4]$ and $\Delta U[5]$ are zero and difference $\Delta U[2]<<8$ is chosen to have at most one active nibble after application of π_3. As it is obvious, we have $\pi(0)=0$ and for the sake of simplicity, in the above relation we assume $a_0=0$. The second set includes 16 distinct differences as follows:

$$\underset{0\le j\le 15}{\nabla_j^K}:\begin{cases}\Delta U[k]=0, & k=0,1,2,3,4\\ \Delta U[5]=a_j, & a_{j,[2,3,4,5]}=dec2bin(j),a_{j,[0,1,6,7,8,9,10,11,12,13,14,15]}=0\end{cases}$$

Where, $dec2bin(j)$ corresponds to the binary representation of decimal value j. As it can be seen, in each of these differences $\Delta U[0]$, $\Delta U[1]$, $\Delta U[2]$, $\Delta U[3]$ and $\Delta U[4]$ are zero and the difference $\Delta U[5]$ takes all of the 16 possible values. The round-key differences resulted from the two sets of key differences Δ_i^K and ∇_j^K, are provided in Table 5.a in Appendix B.

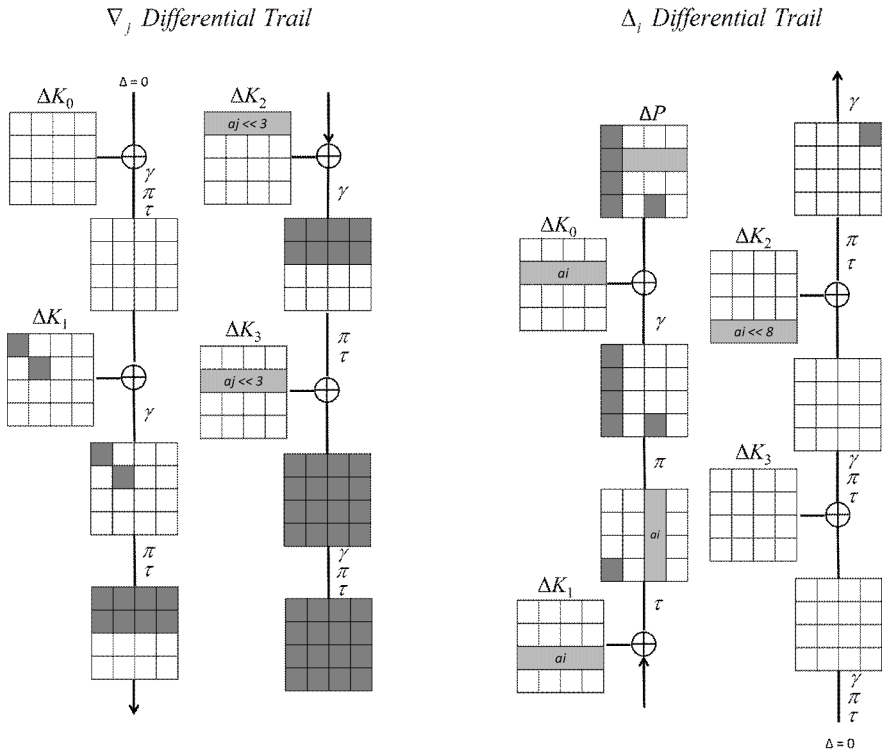

Fig. 2. Differential trails for a four round biclique of mCrypton-96 according to key differences ∇_j^K (left) and Δ_i^K (right) .

Fig. 2 shows that 4-round related key differentials $\Delta-differential$ and $\nabla-differential$ resulted from Δ_i^K and ∇_j^K, share no active S-boxes. So, as it was mentioned before, the constructed biclique is correct.

These differential trails, actually, show the parts of computation of P_i and S_j for each i and j which are different from the *base computation*. $\nabla-differential$ is clear and self-explaining. But, $\Delta-differential$ needs some descriptions. According to ΔK_2 and the definition of Δ_i^K, there are four possible active nibbles in the 4-th column of Δx_2^γ. Also, according to the property 1, it is easy to check that $\Delta x^\gamma_{1,[2,6,10,14]} = \Delta x^\gamma_{2,[3,7,11,15]} << 1$ (Parameters: f=3, s= 2). In Fig. 2 we have selected nibble 3 of Δx_2^γ to be active, hence nibble 14 of Δx_1^γ is also active. Truly, there are four possible $\Delta-differential$ s, which Fig. 2 shows one of them. By examining all of the four possible differential trails, it can be found that the difference of plaintexts is one of the differences in Fig. 3.

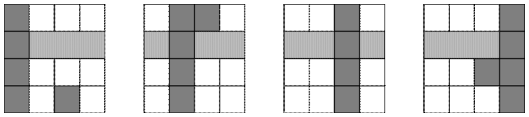

Fig. 3. Possible plaintext differences according to possible Δ_i^K related-key differential trails.

As it can be seen, in three cases we have 5 active nibbles and in one case there is only four active nibbles. Also, according to ΔK_0, ΔP takes at most 61 additional possible differences a_is in three nibbles of the second row. So, the number of possible plaintext differences is $61\times(3\times 2^{4\times 5}+2^{4\times 4})\approx 2^{27.54}$.

Moreover, a sample biclique is constructed by the following four steps:

Step 1: Set an arbitrary 96 bit value for the user key and assume it *base key* $K[0,0]$.

Step 2: Encrypt the plaintext $P_0=0$ under key $K[0,0]$ to compute the intermediate value x_4^τ. Since $K[0,0]\oplus\nabla_0^K=K[0,0]$, assume S_0 is equal to x_4^τ. This step is considered as the *base computation*.

Step 3: For each $1\le i\le 60$ decrypt S_0 under key $K[0,0]\oplus\Delta_i^K$ to obtain the plaintext P_i. Since $K[0,0]\oplus\Delta_0^K=K[0,0]$, so, as it is expected, the plaintext value is P_0 for $i=0$. So we have $P_i \overset{K[0,0]\oplus\Delta_i^K}{\leftrightarrow} S_0, 0\le i\le 60$.

Step 4: For each $0\le j\le 15$ encrypt P_0 under key $K[0,0]\oplus\nabla_j^K$ to obtain the intermediate value x_4^τ and assume it S_j. So we have $P_0 \overset{K[0,0]\oplus\nabla_j^K}{\leftrightarrow} S_j, 0\le j\le 15$.

For such a biclique it is expected that for each $0\le i\le 60$ and $0\le j\le 15$, encryption/decryption process $P_i \overset{K[0,0]\oplus\Delta_i^K\oplus\nabla_j^K}{\leftrightarrow} S_j$ is satisfied. However, the constructed biclique is a biclique with the maximum dimension in a *biclique group*. Actually, according to the asymmetric key partitioning, for an arbitrary biclique in a *biclique group*, it is not necessary that all possible keys $K[0,0]\oplus\Delta_i^K\oplus\nabla_j^K$, $0\le i\le 60$, $0\le j\le 15$, are considered.

4.2. Asymmetric key partitioning

According to the key scheduling of mCrypton, we perform the key partitioning directly to the main key. However, in the proposed method the key partitioning is somewhat different. In the usual method of biclique attacks we construct each biclique for a value of base key, which all of the bicliques are from the same dimensions and the base keys are the certain values of some bits. However, in the proposed method we have the bicliques from different dimensions and also the base keys must be pre-computed. In this section we will describe the proposed key partitioning method but at first, we must define the space generator.

4.2.1 Definition of the minimal space generator

Assume space $\boldsymbol{S}$ is corresponds to a set of all 2^m binary vectors of length m and D is a given subset of $\boldsymbol{S}$. Now, we define a minimal space generator of $\boldsymbol{S}$ based on set D.

Definition 1 Assume G^D is a subset of $\boldsymbol{S}$, which for each element of $\boldsymbol{S}$ like $S_i, 0 \le i \le 2^m - 1$ there is an element $G_k^D \in G^D$, $0 \le k \le |G^D| - 1$ and an element $D_j \in D$, $0 \le j \le |D| - 1$ such that $S_i = G_k^D \oplus D_j$. Then G^D is a space generator of $\boldsymbol{S}$ based on set D. Now, the space generator G^D is minimal, if by removing any element of G^D, the remained G^D is not a space generator any more.

For clarifying, for any D, $G^D = S$ is a trivial space generator of $\boldsymbol{S}$ but may not be a minimal one. In general, for an S and a given set D, the minimal space generator is not necessarily unique. Note that there are some optimal space generators which are minimal space generators with the minimum number of elements. As it can be seen later, smaller cardinality of a minimal space generator, could lead to an attack with less complexity, but in this work we prefer not to encounter such an optimization problem. However, the semi-optimal algorithm in Appendix A is proposed to find a minimal space generator of a known $\boldsymbol{S}$ and a given D. Although the obtained G^D is not certainly optimal, the algorithm is so fast and the results can be easily verified.

After obtaining a space generator G^D, we also want a partitioning of S according to the obtained G^D. In other words, we want to assign each element of S to a unique element of G^D. Obviously, the number of elements of such a space generator could not be less than $2^m / |D|$. Our experiments show that the cardinality of obtained G^D is less than $2^{m+2} / |D|$. Also, except a special case is explained later, this cardinality is usually about $2^{m+1} / |D|$. Hence, for a certain value of i there could be more than one pair (k, j) which $S_i = G_k^D \oplus D_j$. In other words, there are elements of space $\boldsymbol{S}$ which can be generated by more than one element of G^D. Hence, to partitioning $\boldsymbol{S}$, we must consider a mechanism in which each element of $\boldsymbol{S}$ is assigned to a unique element of G^D. For this purpose, according to this fact that in the proposed algorithm elements of G^D are obtained sequentially, for a new element of G^D we assign only those elements of $\boldsymbol{S}$ which are generated by this new element and have not been assigned to the previous selected elements of G^D.

For each element $G_k^D \in G^D$, $0 \le k \le |G^D| - 1$, the assigned unique elements of $\boldsymbol{S}$ are stored in $\{A\}_k^D$, and their corresponding elements of D are stored in $\{B\}_k^D$ (for each element $\{A\}_{k,z}^D \in \{A\}_k^D$ we have $\{A\}_{k,z}^D = G_k^D \oplus \{B\}_{k,z}^D$). Thus, through the algorithm, we obtain a minimal space generator G^D and also we partition the space $\boldsymbol{S}$ with respect to the elements of G^D. Hence we have $\bigcup_{k=0}^{|G^D|-1} \{A\}_k^D = S$ and $\forall_{0 \le k_1, k_2 \le |G^D| - 1} k_1 \ne k_2$, $\{A\}_{k_1}^D \bigcap \{A\}_{k_2}^D = \varnothing$.

A special case of minimal space generator. Assume a set D which the linear combination of each two elements of it is also an element of D, i.e., D is a closed set under XOR operation. The elements of such a set D are actually the code-words of a linear block code of length m over GF(2). Hence, the cardinality of D is certainly a power of 2 and there is an optimal space generator based on set D with the exact cardinality $2^m / |D|$. So, if one is encountered with such a case it is better to use the certain simple methods in linear block codes for obtaining an optimal space generator instead of the proposed algorithm [13]. However, in this paper we do not deal with this special case.

4.2.2 Asymmetric key partitioning and bicliques topology

The goal of key partitioning in a biclique attack is to determine the values of base keys, which the bicliques are constructed based on them. According to the definition of minimal space generator in the previous section, it is easy to describe how the key partitioning is done in the non-isomorphic biclique attack. In the following, without loss of generality we describe it according to key differences sets Δ_i^K and ∇_j^K defined in Section 4.1. At first, we define set D as follow:

$$\underset{0\le i\le 60, 0\le j\le 15}{\forall i,j} \; D_{i\times 16+j} = \Delta^K_{i,[32,\ldots,47,82,\ldots,85]} \oplus \nabla^K_{j,[32,\ldots,47,82,\ldots,85]} \quad (1)$$

As it can be seen, the difference in the other $96-20=76$ bits of $\Delta_i^K \oplus \nabla_j^K$, $0 \le i \le 60$, $0 \le j \le 15$, is always zero. In the first step of key partitioning, we set a user key which has a certain value in these 76 bits and is zero in the other 20 bits. This way we have 2^{76} user keys, each of them indicated by K^b. Now, for each K^b we construct a set of bicliques of different dimensions called *biclique group*. For this purpose, we need the values of base keys for each biclique in a *biclique group*. These base keys are actually the elements of minimal space generator G^D of space $\boldsymbol{S}$ with dimension 20 based on D. Hence, if K^b is the user key of a *biclique group* then we replace 20 bits $[32,\ldots,47,82,\ldots,85]$ in K^b with each $G_k^D \in G^D$, $0 \le k < |G^D|$ to get a new base key. Note that the other 76 bits of all base keys in a biclique group are the same as those bits of K^b. It is obvious that there is no need to re-obtain the minimal space generator G^D for another *biclique groups* (corresponds to a new value of K^b) and it is enough to precompute it only once. Thus, the number of all base keys and consequently the number of all bicliques is $|G^D| \times 2^{76}$.

As it was mentioned before, the dimensions of bicliques in a biclique group are not the same. However, the topology of these bicliques are dependent to the prcomputed sets $\{A\}_k^D$s (or $\{B\}_k^D$s), $0 \le k < |G^D|$. In the rest of this section, we introduce how to determine the topology of bicliques in a biclique group. As it will be seen, according to the base keys and the bicliques topology in the biclique groups, the whole 96-bit key space is partitioned, definitely.

Each *biclique group* must partition 2^{20} values of user key while according to the 4-round biclique introduced in Section 4.1, each biclique contains at most $61 \times 16 = 976$ distinct keys. Hence, the whole key space of a *biclique group* cannot be covered by a single biclique.

In other hand, D has 976 elements each of 20 bits length. An important issue is that we must check D to ensure that all of the elements are distinct. If there are

some repeated elements they must be eliminated from *D*. Since the two sets Δ and ∇ do not have any overlapped nibbles, all of the 976 elements are distinct. Now, we call the algorithm in Appendix A, for *m = 20* and the *D* to obtain minimal space generator elements $G_k^D, 0 \le k < |G^D|$ and their corresponding $\{B\}_k^D$ s. Then for each G_k^D, we make a biclique of *biclique group*. For constructing the biclique corresponds to G_k^D, we need to define two sets of key differences are denoted by $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ both of them are obtained from $\{B\}_k^D$. Actually, $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ are subsets of Δ_i^K and ∇_j^K, respectively. For clarifying, $\{B\}_k^D$ is a subset of *D*, hence according to (1), it is possible to be some elements Δ_i^K or ∇_j^K which has no effect on $\{B\}_k^D$ and can be ignored. For example if there is no ∇_j^K, $0 \le j \le 15$, which $D_{47\times16+j} = \Delta^K_{47,[32,..,47,82,..,85]} \oplus \nabla^K_{j,[32,..,47,82,..,85]} \in \{B\}_k^D$ then Δ_{47}^K can be ignored. In a same way if there is no Δ_i^K, $0 \le i \le 60$, which $D_{i\times16+9} = \Delta^K_{i,[32,..,47,82,..,85]} \oplus \nabla^K_{9,[32,..,47,82,..,85]} \in \{B\}_k^D$ then ∇_9^K can be ignored. Hence the elements of $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ are those of Δ_i^K $, 0 \le i \le 60$, and ∇_j^K, $0 \le j \le 15$, which are not ignored.

At first, $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ are initialized as null sets. If for an *i*, $0 \le i \le 60$, there be at least one *j*, $0 \le j \le 15$ which $D_{i\times16+j} \in \{B\}_k^D$ then we add $\Delta_{i,[32,..,47,82,..,85]}$ to $\{B^\Delta\}_k^D$. In the same way, we examine all $0 \le j \le 15$ and if for an *j* there be at least one *i*, $0 \le i \le 60$, which $D_{i\times16+j} \in \{B\}_k^D$ then we add $\nabla^K_{j,[32,..,47,82,..,85]}$ to $\{B^\nabla\}_k^D$. Considering this fact, we will make the biclique corresponds to G_k^D by $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ instead all Δ_i^K s, $0 \le i \le 60$ and ∇_j^K s, $0 \le j \le 15$.

As it was mentioned, each G_k^D is the *base key* of *k*-th biclique (in *biclique group*). and $\{A\}_k^D$s are a partitioning of the space $\boldsymbol{S}$ with *m* = 20. Hence by replacing $\{A\}_{k,z}^D$s, $0 \le z \le \left|\{A\}_k^D\right|$, in bits $[32,\ldots,47,82,\ldots,85]$ of K^b we obtain the corresponding user keys and by examining all base keys G_k^D s we obtain all 2^{20} user keys in *biclique group*. In the other hand, as it is expected, for each $\{B\}_{k,z}^D \in \{B\}_k^D$ there is only one pair $\{B^\Delta\}_{k,i_1}^D$ and $\{B^\nabla\}_{k,i_2}^D$ which $\{B\}_{k,z}^D = \{B^\Delta\}_{k,i_1}^D \oplus \{B^\nabla\}_{k,i_2}^D$. We store those pairs (i_1, i_2) in a key index set $\{\Lambda\}_k$ as follow:

$$\underset{0\le i_1\le\left|\{B^\Delta\}_k^D\right|;0\le i_2\le\left|\{B^\nabla\}_k^D\right|}{\forall i_1, i_2} \quad if \;\; \{B^\Delta\}_{k,i_1}^D \oplus \{B^\nabla\}_{k,i_2}^D \in \{B\}_k^D \Rightarrow Add \;\; pair \;\; (i_1, i_2) \;\; to \;\; \{\Lambda\}_k$$

We have $|\{B\}_k^D| = |\{\Lambda\}_k|$, $0 \le k \le |G^D| - 1$, because for each $\{B\}_{k,z}^D \in \{B\}_k^D$ there is a unique pair $(\{B^\Delta\}_{k,i_1}^D, \{B^\nabla\}_{k,i_2}^D)$ which $\{B^\Delta\}_{k,i_1}^D \oplus \{B^\nabla\}_{k,i_2}^D = \{B\}_{k,z}^D$. So, $\{\Lambda\}_k$ s (same as $\{B\}_k^D$ s) are corresponds to a partitioning of 2^{20} user key values in each *biclique group*; hence $\sum_{k=0}^{|G^D|-1} |\{\Lambda\}_k| = 2^{20}$. Note that, there may be some $\{B^\Delta\}_{k,i_1}^D$ and

$\{B^{\nabla}\}^{D}_{k,i_2}$ which $\{B^{\Delta}\}^{D}_{k,i_1} \oplus \{B^{\nabla}\}^{D}_{k,i_2} \notin \{B\}^{D}_{k}$ so $|\{B^{\nabla}\}^{D}_{k}| \times |\{B^{\Delta}\}^{D}_{k}| \geq |\{B\}^{D}_{k}|$, $0 \leq k < |G^{D}|$.

In the case of mCrypton-96 there is a property for the key differences sets Δ^K and ∇^K leads to a special case of asymmetric key partitioning which is explained in the next section.

4.2.3 Asymmetric key partitioning for non-overlapping Δ^K and ∇^K sets

As it can be seen, we have no shared active bits between two key differences sets Δ^K_i and ∇^K_j defined in Section 4.1, and 4 active bits in ∇^K_j s change independently of 16 active bits in Δ^K_i s. Thus, according to the definition of D in (1), $D_{[16,17,18,19]}$ and $D_{[0,1,\ldots,15]}$ only depend on ∇^K and Δ^K, respectively. In such a case, a space generator can be constructed by combining two minimal space generators for Δ^K and ∇^K which are obtained independently. For clarifying, assume that by recalling the algorithm in Appendix A, we obtain a minimal space generator G^{D_Δ} and its corresponding partitioning set $\{B\}^{D_\Delta}$ for $D_\Delta = \Delta^K_{i,[32,\ldots,47]}$, $0 \leq i \leq 60$ and $m = 16$. Also we obtain a minimal space generator G^{D_∇} and its corresponding $\{B\}^{D_\nabla}$ for $D_\nabla = \nabla^K_{j,[82,\ldots,85]}$ and $m = 4$. Now, there is a minimal space generator G^D for defined D and $m = 20$ which has $|G^{D_\Delta}| \times |G^{D_\nabla}|$ elements and is constructed by combining G^{D_Δ} and G^{D_∇} as follows:

$$\underset{\substack{0 \leq i \leq |G^{D_\Delta}|-1,\\ 0 \leq j \leq |G^{D_\nabla}|-1}}{\forall i,j} \quad k = i \times |G^{D_\nabla}| + j, \begin{cases} G^{D}_{k,[0,1,\ldots,15]} = G^{D_\Delta}_{i} \\ G^{D}_{k,[16,17,18,19]} = G^{D_\nabla}_{j} \end{cases}$$

This way, for each $\{B^{\Delta}\}^{D}_{k}$ and $\{B^{\nabla}\}^{D}_{k}$, $0 \leq k < |G^{D_\Delta}| \times |G^{D_\nabla}|$, we have:

$$\underset{\substack{0 \leq i \leq |G^{D_\Delta}|-1,\\ 0 \leq j \leq |G^{D_\nabla}|-1}}{\forall i,j} \quad k = i \times |G^{D_\nabla}| + j, \begin{cases} \{B^{\Delta}\}^{D}_{k,[0,1,\ldots,15]} = \{B\}^{D_\Delta}_{i} \ , \ \{B^{\Delta}\}^{D}_{k,[16,17,18,19]} = 0 \\ \{B^{\nabla}\}^{D}_{k,[0,1,\ldots,15]} = 0 \ , \ \{B^{\nabla}\}^{D}_{k,[16,17,18,19]} = \{B\}^{D_\nabla}_{j} \end{cases}$$

Hence, each $\{\Lambda\}_k$, $0 \leq k < |G^{D_\Delta}| \times |G^{D_\nabla}|$, contains all possible pairs (i_1, i_2), $0 \leq i_1 \leq \left|\{B^{\Delta}\}^{D}_{k}\right|$, $0 \leq i_2 \leq \left|\{B^{\nabla}\}^{D}_{k}\right|$, and so there is no need to compute separately.

According to $D_\nabla = \nabla^K_{j,[82,\ldots,85]}$, it is easy to see that the minimal space generator G^{D_∇} consists of only one 4-bit zero vector; so, its corresponding $\{B\}^{D_\nabla}_0$ takes all 16 possible values. However, to find a minimal G^{D_Δ} we call the algorithm in Appendix A for $D^{\Delta} = \Delta^K_{i,[32,\ldots,47]}$, $0 \leq i \leq 60$, and $m = 16$. The obtained G^{D_Δ} contains $2688 \approx 2^{11.4}$ elements of 16-bit length which is less than $2^{18}/61 \approx 2^{12}$. According to the algorithm, it is expected that the cardinality of $\{B\}^{D_\Delta}_k$s , $0 \leq k \leq 2687$, be in a descending order. Table. 2 shows the changes of

$|\{B\}_k^{D_\Delta}| = |\{A\}_k^{D_\Delta}|$ according to k. Also we have $\sum_{k=0}^{2687} |\{B\}_k^{D_\Delta}| = 2^{16}$ because $\{B\}_k^{D_\Delta}$s corresponds to a partitioning of a space $\boldsymbol{S}$ with m=16.

Table. 2 Cardinality of obtained $\{B\}_k^{D_\Delta}$s

k	0-255	256-511	512-767	768-895	896-1279	1280-1791	1792-1983	1984-2431	2432-2687
$\lvert\{B\}_k^{D_\Delta}\rvert$	61	49	41	37	33	13	5	3	2

Now, we construct a G^D by combining G^{D_Δ} and G^{D_∇}. G^{D_Δ} consists of 2688 elements while G^{D_∇} has only one element; so, the number of elements of G^D is also 2688. For each G_k^D, $0 \le k \le 2687$, it is enough to set $G^D_{k,[16,17,18,19]}$ to $G_0^{D_\nabla}$ which is always zero and set $G^D_{k,[0,1,\ldots,15]}$ to $G_k^{D_\Delta}$. Also, for each $0 \le k \le 2687$, $\{B^\Delta\}^D_{k,[0,1,\ldots,15]}$ is equal to $\{B\}_k^{D_\Delta}$, and $\{B^\Delta\}^D_{k,[16,17,18,19]}$ is always zero. In a same way, for each $0 \le k \le 2687$, $\{B^\nabla\}^D_{k,[16,17,18,19]}$ always contains all of the 16 possible values (according to $\{B\}_0^{D_\nabla}$) and $\{B^\nabla\}^D_{k,[0,1,\ldots,15]}$ is always zero.

As it can be seen, in each *biclique group* which is constructed based on the proposed partitioning we have 2688 bicliques are classified into the 9 non-isomorph bicliques. Actually, we have 256 bicliques with dimension (61,16), 256 bicliques with dimension (49,16), …, 448 bicliques with dimension (3,16) and 256 bicliques with dimension (2,16). As it is expected, $\sum_{k=0}^{2687} |\{B\}_k^{D_\nabla}| \times |\{B\}_k^{D_\Delta}| = 2^{20}$ and these bicliques will cover all 2^{20} distinct keys in each biclique group.

4.3. Non-isomorphic biclique attack on mCrypton-96

Using the proposed key partitioning and the method for constructing 4-round bicliques, now we propose a biclique attack on full-round mCrypton-96 as follows. In this attack the matching positions are two nibbles 0 and 12 of x_8^σ.

Step 1: Two key differences sets Δ_i^K and ∇_j^K, has no shared active bits so pre-compute space generator G^D and its corresponding sets $\{B^\Delta\}^D$ and $\{B^\nabla\}^D$ (as it was described in section 4.2.3). For the sake of simplicity, we expand obtained G_k^Ds, $0 \le k \le 2687$, and their corresponding sets $\{B^\Delta\}_k^D$ and $\{B^\nabla\}_k^D$ to a 96-bit length version (equal to the length of user key) through zero-padding. So, hereinafter, each element of them has its original value in 20 bits $[32,\ldots,47,82,\ldots,85]$ and is zero in the other 76 bits.

Step 2: Make a new *biclique group*. For this purpose, at first, take a new value for 96 bit user key K^b (initial value of register *U*) in 76 bits which are always zero in $\Delta_i^K \oplus \nabla_j^K$ s. The other 20 bits $[32,\ldots,47,82,\ldots,85]$ of K^b are fixed to zero.

Step 3: For each $G_k^D, 0 \le k \le 2687$ compute base key $K[0,0] = K^b \oplus G_k^D$ and do the following Steps 4 to 7 for $K[0,0]$.

Step 4: Make a biclique according to the obtained base key $K[0,0]$. For this purpose, do the following sub-steps:

Step 4.1: (*base computation*): Encrypt the plaintext $P_0 = 0$ by $K[0,0]$ to compute the intermediate value x_4^τ and assume it S_0.

Step 4.2: For each $1 \le i < |\{B^\Delta\}_k^D|$ compute the plaintext value by decrypting S_0 under key $K[i,0] = K[0,0] \oplus \{B^\Delta\}_{k,i}^D$ and assume it P_i.

Step 4.3: For each $1 \le j < |\{B^\nabla\}_k^D|$ compute the intermediate value x_4^τ by encrypting $P_0 = 0$ under key $K[0,j] = K[0,0] \oplus \{B^\nabla\}_{k,j}^D$ and assume it S_j (clearly, for $j = 0$ we get S_0)

Step 5: (pre-computation) For each P_i, $0 \le i < |\{B^\Delta\}_k^D|$, ask the oracle to encrypt P_i and obtains its corresponding ciphertext C_i. Then, according to $K[i,0]$ compute four round keys K_9 to K_{12}. Also, obtain the values of 16+12=28 intermediate nibbles (indicated in Fig. 6.a) with decryption C_i under key $K[i,0]$. Note that pre-computation is only done for the nibbles which are not needed to re-compute in Step 7. As it is shown in Fig. 6.a, totally we need to compute and store the values of 28+4=32 S-boxes and 12 π operations for each P_i.

Step 6: (pre-computation) For each S_j, $0 \le j < |\{B^\nabla\}_k^D|$, according to $K[0,j]$ compute five round keys K_4 to K_8 and obtain the values of 12 intermediate nibbles with encryption S_j under key $K[0,j]$. As it is shown in Fig. 6.b, we need to compute and store the values of 12 S-boxes for each S_j.

Step 7: (re-computation) For each $0 \le i < |\{B^\Delta\}_k^D|$, $0 \le j < |\{B^\nabla\}_k^D|$ re-compute the value of two matching nibbles 0 and 12 of x_8^σ by decryption C_i under key $K[0,j]$ and re-compute the value of the same nibbles by encryption S_j under key $K[i,0]$. Then check if these two values are equal or not. If they are not equal then $K[i,j] = K[0,0] \oplus \{B^\Delta\}_{k,i}^D \oplus \{B^\nabla\}_{k,j}^D$ is certainly incorrect. Nonetheless, if they are equal, it is possible for $K[i,j]$ to be the correct user key. So, we must check the correctness of $K[i,j]$ with a trial full-round encryption. If $K[i,j]$ is the correct user key, terminate the procedure and return value of the found user key. Otherwise, if you did not have yet checked all base keys $K[0,0]$, then go to Step 3 to check another base key and if all of the base keys has been checked (i.e. k = 2687) then go to Step 2 for a new *biclique group*.

The most important issue in the re-computation step, is that actually we do not need to re-compute full rounds to obtain the values of the two mentioned matching nibbles. As it is shown in Fig. 7.a and Fig. 7.b, for each *i* and *j*, it is enough to re-compute only the values of 44 S-boxes and 14 π operation in the data encryption direction, 10 S-boxes and 6 π operations in the data decryption direction and 4 S-boxes in the key scheduling process. So, for each *i* and *j* we need to re-compute the values of 58 S-boxes and 20 π operations, totally.

A note on the general form of non-isomorphic biclique attack: As it was described in Section 4.2.3, for the cases with non-overlapping sets Δ^K and ∇^K, such as the case of mCrypton-96, each $\{\Lambda\}_k$ includes all of the possible ordered pairs of indexes. However, in the general form of *non-isomorphic biclique attack*, we must also compute $\{\Lambda\}_k$s in Step 1 and revise Step 7 such that the re-computation is performed for each $(i, j) \in \{\Lambda\}_k$.

Improvements for the attack procedure: There are some optimizations for Steps 4 and 7 which reduce the complexity of these steps. In Step 4, for each biclique construction, all of P_is are obtained from a single intermediate state S_0, and all intermediate states S_js are obtained from a single plaintext P_0. Hence, the biclique construction can be performed using a pre-computation and re-computations. To do this, first we perform a base pre-computation $P_0 \overset{K[0,0]}{\rightarrow} S_0$ and store all intermediate states values. Then, equivalent to the Step 4.2, the values of P_is are determined by re-computation through $P_i \overset{K[i,0]}{\leftarrow} S_0$. One can easily observe in Fig. 2 that the re-computation of each P_i needs only computation of 6 S-boxes and 3 π operations. In a same way, equivalent to the Step 4.3, all S_js are determined by re-computation through $P_0 \overset{K[0,j]}{\rightarrow} S_j$. As it can be seen in Fig. 2, re-computation of each S_j needs only computation of 28 S-boxes (26 S-boxes in data processing part and 2 S-boxes in the round keys) and 10 π operations.

Another optimization is in Step 7 where both of matching nibbles 0 and 12 of x_8^σ are checked simultaneously. We can reduce the complexity of this step using an *early abort* technique. For this purpose, at first we perform the matching for nibble 0 of x_8^σ and if the matching holds for this nibble with probability 2^{-4}, then we perform the matching for nibble 12 of x_8^σ. Thus, instead of 58 S-boxes and 20 π operations for each re-computation, we only re-compute 53 S-boxes and 19 π operations to verify nibble 0 matching and if this matching holds with probability 2^{-4} then the matching of nibble 12 is verified by re-computation of remained 5 S-boxes and one π operation.

4.4. Complexity of the attack

Data complexity: According to Fig. 3 the number of possible plaintext differences is about $61 \times (3 \times 2^{4\times5} + 2^{4\times4}) \approx 2^{27.54}$. In the other hand, we always assume starting plaintext $P_0 = 0$ for all bicliques. Hence, we need at most $2^{27.54}$ plaintexts and their corresponding ciphertexts.

Time Complexity: The most significant part of time complexity in the attack procedure, is the re-computation complexity in Step 7. In a re-computation process it is needed to compute only a partial encryption or decryption. Thus, we need a criterion to determine its complexity. As it was mentioned in [3], S-boxes are the major contributor to the practical complexity of AES encryption/decryption both in hardware and software implementation. The same situation is acceptable for the most of block ciphers especially for the AES-like ciphers. So, it seems reasonable to determine the complexity of re-computation with respect to the number of S-boxes which must be re-computed. However, in this paper we get a stricter criterion. Thus the complexity of each S-box computation and also each binary bit permutation π is considered as a single memory access. According to this criterion, the complexity of one full-round encryption/decryption of mCrypton-96 is about 296 memory accesses including $(12\times16)+(13\times4)=244$ memory accesses for S-boxes (in both of data processing part and key scheduling) and $13\times4=52$ memory accesses for π operations. Note that, the complexity of key-addition and other linear operations, compared with S-boxes and π operations is absolutely negligible. Thus, the complexity of re-computation is determined as the relative number of needed S-boxes and π operations to all 296 memory accesses.
As it can be seen in the attack procedure, all of the *biclique groups* have the same process. So, for overall time complexity computation, it is enough to consider the complexity for a single *biclique group*. For this purpose, the complexity of steps 3 to 7 is determined as follows:

Step 3: consists of 2688 XOR operation which is negligible.

Step 4: for *k*-th element of minimal space generator G^D, this step is performed through one 4-round encryption (base computation) , 28+10=38 memory accesses to compute each S_j, $1\le j<|\{B^{\nabla}\}_k^D|$, and 6+3=9 memory accesses to compute each P_i, $1\le i<|\{B^{\Delta}\}_k^D|$. So, the complexity of this step for a *biclique group* is totally equivalent to $\frac{4}{12}\times2688+\left(\frac{38}{296}\times\sum_{k=0}^{2687}(|\{B^{\nabla}\}_k^D|-1)\right)+\left(\frac{9}{296}\times\sum_{k=0}^{2687}(|\{B^{\Delta}\}_k^D|-1)\right)$ full-round encryptions. $|\{B^{\nabla}\}_k^D|$ is equal to 16 for each *k* and $\sum_{k=0}^{2687}|\{B^{\Delta}\}_k^D|$ is equal to 2^{16}. Hence, the time complexity of this step is equivalent to $\frac{4}{12}\times2688+\frac{38}{296}\times(16-1)\times2688+\frac{9}{296}\times(2^{16}-2688)=2^{12.96}$ full-round encryptions.

Step 5: for *k*-th element of G^D, this step is performed through $|\{B^{\Delta}\}_k^D|$ times 32+12=44 memory accesses. So, the overall complexity of this step for a *biclique group* is equivalent to $\frac{44}{296}\times\sum_{k=0}^{2687}|\{B^{\Delta}\}_k^D|=\frac{44}{296}\times2^{16}=2^{13.24}$ full-round encryptions.

Step 6: for *k*-th element G^D, this step is performed through $|\{B^{\nabla}\}_k^D| = 16$ times of 12 memory accesses. So, the overall complexity of this step for a *biclique group* is equivalent to $\frac{12}{296} \times \sum_{k=0}^{2687} |\{B^{\nabla}\}_k^D| = 2^{10.76}$ full-round encryptions.

Step 7: Time complexity of this step is consists of two parts. First part is the complexity of re-computations for determining the values of two matching nibbles and the other one is the complexity to filter out the false positives. For each re-computation we need to re-compute the values of 58 S-boxes and 20 π operations, totally. According to the early abort technique, 53 S-boxes and 19 π operations must be re-computed for the first matching nibble and another 5 S-boxes and one π operation is re-computed with probability 2^{-4}. Hence, for *k*-th element of G^D, the time complexity is about $|\{B^{\Delta}\}_k^D| \times |\{B^{\nabla}\}_k^D| \times \left((53+19) + 2^{-4} \times (5+1)\right) = |\{B^{\Delta}\}_k^D| \times |\{B^{\nabla}\}_k^D| \times 2^{6.18}$ memory accesses, on average. So, the overall time complexity of this step is equivalent to $\frac{1}{296} \times 2^{6.18} \times \sum_{k=0}^{2687} |\{B^{\Delta}\}_k^D| \times |\{B^{\nabla}\}_k^D| = 2^{-2.03} \times 2^{20} = 2^{17.97}$ full-round encryptions for a *biclique group*.

Note that, in the general form of *non-isomorphic biclique attack*, re-computation must be done for each $(i, j) \in \{\Lambda\}_k$. So, this complexity is equal to $\frac{1}{296} \times 2^{6.18} \times \sum_{k=0}^{2687} |\{\Lambda\}_k|$. As it was mentioned in Section 4.2.2, $\{\Lambda\}_k$s are corresponds to a partitioning of whole space $\boldsymbol{S}$ with *m*=20. Hence $\sum_{k=0}^{2687} |\{\Lambda\}_k| = 2^{20}$ and the type of differences sets (to be overlapped or non-overlapped) has no any effect on the complexity of this step.

For the second part, the chance of getting a false positive is about 2^{-8}, so for each *biclique group* we obtain about $2^{20} \times 2^{-8} = 2^{12}$ user key values must be checked by a trial encryption. Thus, the time complexity of this part is about 2^{12} full-round encryptions for a *biclique group*.

Totally, the time complexity for each *biclique groups* is about $2^{12.96} + 2^{13.24} + 2^{10.76} + 2^{17.97} + 2^{12} = 2^{18.09}$ full-round encryption. Hence, the overall time complexity is about $2^{76} \times 2^{18.09} = 2^{94.09}$ full-round encryptions. Note that, if the complexity is determined based on the number of active S-boxes (typical criterion), then complexity of this attack is estimated about $2^{93.93}$ full-round encryptions.

Memory Complexity: We need less than 2^{20} bytes of memory to store the pre-computed tables in Step 1. The online stage of the attack (Steps 2 to 7) is also needs negligible bytes of memory to keep intermediate states values.

4.5. Another attack on mCrypton-96 based on a 5-round biclique

The proposed attack in the previous sections can be improved by expanding the biclique of mCrypton-96 to a five round version. For this purpose, we define two sets of key differences Δ_i^K and ∇_j^K on the user key (*U* register) as follows:

$$\underset{0\le i\le 30}{\Delta_i^K}:\begin{cases}\Delta(U[k])=0, & k=0,2,3,4,5\\ \Delta(U[1])=a_i, & \pi_3(a_i<<8)=[0,0,0,0]or[0,0,0,t]or[0,0,t,0],\ t\neq 0\end{cases}$$

$$\underset{0\le j\le 15}{\nabla_j^K}:\begin{cases}\Delta U[k]=0, & k=0,1,2,3,4\\ \Delta U[5]=a_j, & a_{j,[2,3,4,5]}=dec2bin(j), a_{j,[0,1,6,7,8,9,10,11,12,13,14,15]}=0\end{cases}$$

Table 5.b in Appendix B shows the round sub-key differences ΔK_r, $0\le r\le 12$, according to each of these two key differences sets. Also, Fig. 4 shows that the corresponding 5-round $\Delta-differential$ and $\nabla-differential$ share no active components. Thus, a 5-round biclique of at most $31\times 16=496$ keys can be constructed by $1+15+30=46$ 5-round encryptions.

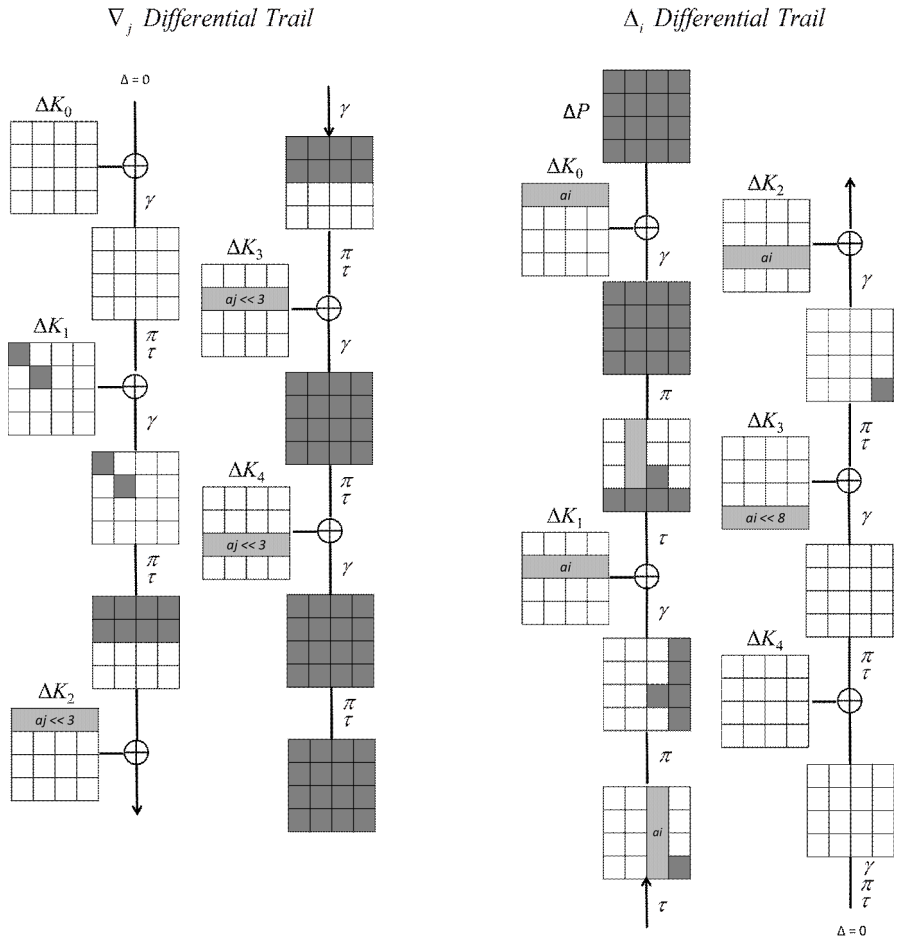

Fig. 4. Differential trails for a five round biclique of mCrypton-96 according to key differences ∇_j^K (left) and Δ_i^K (right).

Depends to the place of active nibble after $\pi_3(a_i<<8)$, there is two possible $\Delta-differential$ s where only one of them is shown in Fig. 4. In another one, each possible differences Δx_2^{γ} is non-zero in 3-th column and zero in the others. It seems that the number of possible differences for the input plaintexts is almost equal to the whole codebook. However, we must consider this fact that for each non-zero difference in the input of mCrypton S-boxes there is only 7 possible differences in its output. Also, there is only 30 possible non-zero differences in

the first column and the same number of non-zero differences in the last column of Δx_1^{π} (15 non-zero difference for each $\Delta-$*differential*). Hence, we expect that about $2^{16}\times\left(1-\frac{7^4}{2^{16}}\right)^{30}\approx 2^{16}\times 2^{-1.62}=2^{14.38}$ differences don't occur in the first and also in the last columns of Δx_1^{I}. An exact evaluation reveals that the number of such differences in each of these columns is $29033=2^{14.83}$. Finally, according to the 31 possible differences of ΔK_0, an exact evaluation reveals the number of impossible plaintext differences in the first and last columns are $10694=2^{13.38}$ and $10735=2^{13.39}$ differences, respectively. Hence, the exact number of possible plaintext differences is about $2^{32}\times(2^{32}-2^{13.39})^2=2^{63.4}$. Thus, data complexity is slightly less than the whole codebook.

Same as the previous attack on mCrypton-96, Δ^K and ∇^K sets has no overlap. Hence, it is enough to construct two space generators G^{D_∇} for $D_\nabla=\nabla^K_{j,[82,..,85]}$, m=4 and G^{D_Δ} for $D^\Delta=\Delta^K_{i,[16,..,31]}$, m=16, separately. As the previous attack G^{D_∇} is only a 4-bit zero vector; so, its corresponding $\{B\}_0^{D_\nabla}$ always takes all of 16 possible values. Also, to find a minimal G^{D_Δ} we recall the algorithm in Appendix A for $D^\Delta=\Delta^K_{i,[16,..,31]}, 0\le i\le 31$ and m = 16. The obtained G^{D_Δ} contains exactly $4096=2^{12}$ elements of 16-bit length. Table 3 shows the cardinality of obtained $\{B\}_k^{D_\Delta}$ for each $0\le k\le 4095$. Thus, as it was discussed in Section 4.2.3, $\{B^\Delta\}_k^D$ s and $\{B^\nabla\}_k^D$ s are constructed same as the previous attack.

Table. 3 Cardinality of obtained $\{B\}_k^{D_\Delta}$ s

K	0-255	256-511	512-767	768-1023	1024-1279	1280-1535	1536-1791	1792-2047
$\lvert\{B\}_k^{D_\Delta}\rvert$	31	29	27	25	23	21	19	17
K	2048-2303	2304-2559	2560-2815	2816-3071	3072-3327	3328-3583	3584-3839	3840-4095
$\lvert\{B\}_k^{D_\Delta}\rvert$	15	13	11	9	7	5	3	1

The attack procedure is very similar to the attack procedure described in Section 4.3 with the same matching nibbles. The most important differences are listed below:

- The pre-computed space generator G^D and its corresponding sets $\{B^\Delta\}^D$ and $\{B^\nabla\}^D$, have 4096 elements instead of 2688. Thus, for each *biclique group*, we construct 4096 bicliques by 4096 base keys.
- In Step 4 after the five round base computation, as it can be seen in Fig. 4, we need 42+14+2=58 memory accesses to re-compute each S_j and 22+7=29 memory accesses to re-compute each P_i. So the complexity of this step is equivalent to $\frac{5}{12}\times 4096+\frac{58}{296}\times(16-1)\times 4096+\frac{29}{296}\times(2^{16}-4096)=2^{14.27}$ full-round encryptions for each *biclique group*.

- According to Fig. 8.a and Fig. 8.b Complexity of steps 5 and 6 are equivalent to $\frac{32+11}{296}\times 2^{16}=2^{13.25}$ and $\frac{12}{296}\times 16\times 4096=2^{11.37}$, respectively.
- In Step 7 for re-computing the value of matching nibbles, as it can be seen in Fig. 9.a and Fig. 9.b, we need (24+9+4)+(9+6) = 52 memory accesses to re-compute nibble 0 of x_8^{σ} and (4+1)+1=6 additional memory access to re-compute another nibble 12. So this step's complexity for each *biclique group* is equivalent to $\frac{1}{296}\times\left(52+2^{-4}\times 6\right)\times 2^{20}=2^{-2.5}\times 2^{20}=2^{17.5}$ full-round encryptions.

As it can be seen, the total time complexity of this attack is about $2^{76}\times\left(2^{14.27}+2^{13.25}+2^{11.37}+2^{17.5}+2^{12}\right)=2^{76}\times 2^{17.75}=2^{93.75}$ full-round encryptions. Considering the typical criterion (number of active S-boxes) the complexity estimation is reduced to about $2^{93.55}$ full-round encryptions.

5. Non-isomorphic Biclique Cryptanalysis of mCrypton-128

Non-isomorphic biclique attacks can also perform on the other versions of mCrypton. In this section we study such a attack for mCrypton-128. Table 4 shows how the internal register *U* changes through the rounds.

Table 4. Updated register ***U*** through the rounds of mCrypton-128

r = 0	U[0]	U[1]	U[2]	U[3]	U[4]	U[5]	U[6]	U[7]
r = 1	U[5]	U[6]	U[7]	U[0] << 3	U[1]	U[2]	U[3]	U[4] << 8
r = 2	U[2]	U[3]	U[4] << 8	U[5] << 3	U[6]	U[7]	U[0] << 3	U[1] << 8
r = 3	U[7]	U[0] << 3	U[1] << 8	U[2] << 3	U[3]	U[4] << 8	U[5] << 3	U[6] << 8
r = 4	U[4] << 8	U[5] << 3	U[6] << 8	U[7] << 3	U[0] << 3	U[1] << 8	U[2] << 3	U[3] << 8
r = 5	U[1] << 8	U[2] << 3	U[3] << 8	U[4] << 11	U[5] << 3	U[6] << 8	U[7] << 3	U[0] << 11
r = 6	U[6] << 8	U[7] << 3	U[0] << 11	U[1] << 11	U[2] << 3	U[3] << 8	U[4] << 11	U[5] << 11
r = 7	U[3] << 8	U[4] << 11	U[5] << 11	U[6] << 11	U[7] << 3	U[0] << 11	U[1] << 11	U[2] << 11
r = 8	U[0] << 11	U[1] << 11	U[2] << 11	U[3] << 11	U[4] << 11	U[5] << 11	U[6] << 11	U[7] << 11
r = 9	U[5] << 11	U[6] << 11	U[7] << 11	U[0] << 14	U[1] << 11	U[2] << 11	U[3] << 11	U[4] << 3
r = 10	U[2] << 11	U[3] << 11	U[4] << 3	U[5] << 14	U[6] << 11	U[7] << 11	U[0] << 14	U[1] << 3
r = 11	U[7] << 11	U[0] << 14	U[1] << 3	U[2] << 14	U[3] << 11	U[4] << 3	U[5] << 14	U[6] << 3
r = 12	U[4] << 3	U[5] << 14	U[6] << 3	U[7] << 14	U[0] << 14	U[1] << 3	U[2] << 14	U[3] << 3

Considering the key-scheduling of mCrypton-128 we construct a 4-round independent biclique for the intial rounds. For this purpose, we define two sets of key differences Δ_i^K and ∇_j^K as follow:

$$\underset{0\le i\le 30}{\Delta_i^K}:\begin{cases}\Delta(U[k])=0, k=0,1,2,3,4,5,7\\ \Delta(U[6])=a_i,\ \pi_3(a_i)=[0,0,0,0]or[t,0,0,0]or[0,t,0,0], t\neq 0\end{cases}$$

$$\underset{0\le j\le 15}{\nabla_j^K}:\begin{cases}\Delta(U[k])=0, k=0,1,2,4,5,6,7\\ \Delta(U[3])=a_j,\ a_{j,[11,12,13,14]}=dec2bin(j), a_{j,[0,1,2,3,4,5,6,7,8,9,10,15]}=0\end{cases}$$

Table 6 in Appendix B shows the round sub-key differences ΔK_r, $0\le r\le 12$, according to each of these key differences sets. Fig. 5 indicates that according to

these two key differential sets there is a 4-round independent biclique in the initial rounds of mCrypton-128. According to the property 1 we have $\Delta x^{\gamma}_{1,[0,4,8,12]} = \Delta x^{\gamma}_{2,[3,7,11,15]} << 1$ for Δ-differential trail (Parameters: f = 1, s = 0). Hence, dependent to which nibble is active in the output of $\pi(a_i)$, there are two possible Δ-differentials, one of them is indicated in Fig. 5. For another one, possible plaintexts differences are non-zero in the first column and zero in the others. Hence, there are totally about $2^{20} + 2^{16} \approx 2^{20.1}$ possible plaintext differences. Thus, according to this fact that P_0 is a fixed value ($P_0 = 0$) for all bicliques, the data complexity of this attack is also equal to about $2^{20.1}$ known plaintext/ciphertexts.

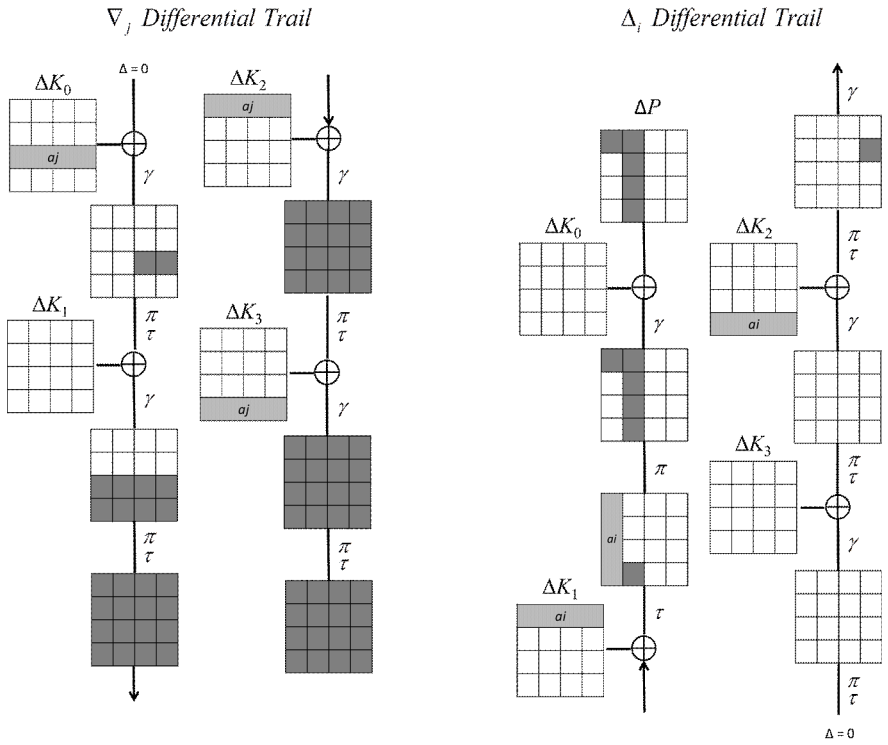

Fig. 5. Differential trails for a four round biclique of mCrypton-128 according to key differences ∇_j^K (left) and Δ_i^K (right).

Same as the previous attacks, Δ^K and ∇^K has no overlap. Also, G^{D_Δ} same as the attack described in Section 4.5 has exactly $4096 = 2^{12}$ elements. Similarly, cardinality of obtained $\{B\}_k^{D_\Delta}$ for each $0 \le k \le 4095$ are the same as Table 3. G^{D_∇} is also consists of only one 4-bit zero vector. The attack procedure is similar to the previous attacks except that the matching nibbles are nibbles 7 and 11 of x_6^{σ}. Also we have $2^{128} / 2^{20} = 2^{108}$ *biclique groups* in this attack.

As it can be seen in Fig. 5, after a 4-round base computation, we need 42+14=56 memory accesses to compute each S_j and at most 6+3=9 memory accesses to compute each P_i. So the complexity of step 4 for each *biclique group* is equivalent to $\frac{4}{12} \times 4096 + \frac{56}{296} \times (16-1) \times 4096 + \frac{9}{296} \times (2^{16} - 4096) = 2^{13.85}$ full-round encryptions. For pre-computation steps 5 and 6, as it can be seen in Fig. 10.a and

Fig. 10.b, complexities are equivalent to $\frac{28+11}{296}\times 2^{16}=2^{13.07}$ and $\frac{12}{296}\times 16\times 4096=2^{11.37}$ for each *biclique group*, respectively. For re-computing the value of matching nibbles in step 7, as it can be seen in Fig. 11.a and Fig. 11.b, we need (8+5)+(41+14) = 68 memory accesses to re-compute nibble 7 of x_6^{σ} and (4+1)+1=6 additional memory access to re-compute another nibble 12. So this step's complexity is equivalent to $\frac{1}{296}\times\left(68+2^{-4}\times 6\right)\times 2^{20}=2^{17.89}$ full-round encryptions for each *biclique group*. Hence, the total time complexity of this attack is about $2^{108}\times\left(2^{13.85}+2^{13.07}+2^{11.37}+2^{17.89}+2^{12}\right)=2^{108}\times 2^{18.05}=2^{126.05}$ full-round encryptions. It can be easily checked that if the time complexity is computed according to the typical criterion then the estimation of complexity is reduced to $2^{125.84}$ full-round encryptions.

6. Conclusion

In this paper, we introduced a modified version of biclique attack, called *non-isomorphic biclique* attack. In the proposed attack, at first, we defined two key differences sets Δ^K and ∇^K which their corresponding related key differential trails satisfy the necessary condition for an independent biclique. Then according to these key differences sets, we performed an asymmetric key partitioning as it was described in Section 4.2. Then this non-uniform partitioning is mapped to the biclique structure. Due to the asymmetry of key partitions, we could have bicliques of different dimensions and so non-isomorph. In the other hand we can choose key difference sets in a way which leads to the longer independent bicliques (as well as data complexity reduction). Through this method we showed that there are independent bicliques up to 5-rounds for the mCrypton cipher. Moreover, two additional minor techniques, one for increasing the speed of biclique construction and the other, early abort technique, for reducing the complexity of matching check have been exploited to reduce the overall time complexity of the attack procedure. Also, we have presented a property of the diffusion layer of mCrypton which can lead to a more reduction of data complexity.

In this paper, we have introduced the general form of asymmetric key partitioning. Moreover, we customized it form for the cases, in which the indicated key differences sets has no shared (overlapped) active bits. Attacks proposed in this paper have used this customization.

As it was mentioned in the introduction of the paper, the proposed method is also applicable to the Crypton cipher. Property 1 described in Section 2 is also compatible with the diffusion layer of Crypton. Hence, the non-isomorphic biclique cryptanalysis of Crypton will be considered as a future work.

References

1. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Biclique Cryptanalysis of the PRESENT and LED Lightweight Ciphers, Cryptol. ePrint Arch. Report 2012/591. http://eprint.iacr.org/2012/591 (2012).

2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology. 3, 3-72 (1991).

3. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: ASIACRYPT 2011, Lecture Notes in Computer Science, vol. 7073, pp. 344–371 (2011).

4. Chen, S., Xu, T.: Biclique Attack of the Full ARIA-256. Cryptol. ePrint Arch. Report 2012/11. http://eprint.iacr.org/2012/11 (2012).

5. Coban, M., Karakoc, F., Boztas, O.: Biclique Cryptanalysis of TWINE. Cryptol. In: CANS 2012. Lecture Notes in Computer Science, vol. 7712, pp. 43-55 (2012).

6. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer. 10(6), 74–84 (1977).

7. Hong, D., Koo, B., Kwon, D.: Biclique Attack on the Full HIGHT. In: ICISC 2011. Lecture Notes in Computer Science, vol. 7259, pp. 365-374 (2011).

8. Hyuk, J.: Security analysis of mCrypton proper to low-cost ubiquitous computing devices and applications. International Journal of Communication Systems. 22(8), 959-969 (2009).

9. Khovratovich, D., Leurent, G. Rechberger, C.: Narrow-Bicliques: Cryptanalysis of Full IDEA. In: EUROCRYPT 2012, Lecture Notes in Computer Science, vol. 7237, pp. 392-410 (2012).

10. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: FSE 2012, Lecture Notes in Computer Science, vol. 7549, pp. 244-263 (2012).

11. Lim, C.H.: Crypton: A New 128-bit Block Cipher. The First Advanced Encryption Standard Candidate Conference, NIST (1998).

12. Lim, C.H., Korkishko, T.: mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: WISA 2005, Lecture Notes in Computer Science, vol. 3786, pp. 243–258 (2006).

13. Lin, S., Castello, D.J.: Error control coding. Prentice-Hall (2004).

14. Mala, H.: Biclique Cryptanalysis of the Block Cipher SQUARE. Cryptol. ePrint Arch. Report 2011/500. http://eprint.iacr.org/2011/500 (2011).

15. Mala, H., Dakhilalian, M., Shakiba, M.: Cryptanalysis of mCrypton—A lightweight block cipher for security of RFID tags and sensors. International Journal of Communication Systems. 25, 415-426 (2011).

16. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: EUROCRYPT 1993, Lecture Notes in Computer Science, vol. 765, pp. 386-397 (1994).

17. Wang, Y., Wu, W., Yu, X.: Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher. In: ISPEC 2012, Lecture Notes in Computer Science, vol. 7232, pp. 337-352 (2012).

Appendix

A. An algorithm for finding a minimal space generator for a known space dimension *m* and a given difference set *D*.

Inputs: space dimension *m*, difference set ***D***

Outputs: minimal space generator G^D, Partitioning $\{A\}^D$ and its corresponding $\{B\}^D$

---

provide binary vector ***R*** of 2^m bits and **initialize** them with zero. %% zero bits of vector *R* indicate those elements of space *S* which have not yet been generated by G^D. On the other hand we set a bit of *R* to one if its corresponding element is generated by G^D %%

set $Nk\_max \leftarrow |D|;\ k \leftarrow 0$.

while there is at least one zero element in ***R,***

  provide vector ***E*** of 2^m elements and **initialize** them with zero.

  set $ind \leftarrow 0$.

  for $i = 0, 1, \ldots, 2^m-1$

    if *i* is not currently an element of G^D **then** ,

      provide vector ***T*** of $|D|$ elements.

      for each $0 \le j \le |D|-1$ **set** $T[j] \leftarrow i \oplus D_j$. %% *T* indicates those elements of space *S* which are generated by *i* based on set *D*%%

      set $E[i] \leftarrow |D| - \sum_{k=0}^{|D|-1} R[T[k]]$. %% *E*[*i*] keeps the number of elements of space *S* which are generated by *i* and their corresponding indexes are zero in vector *R* (and so they cannot generate with the current elements of G^D)%%

      if $E[i]$ is greater than or equal to Nk_max **then**

        set $Z \leftarrow i$.

        set $ind \leftarrow 1$, **break** and **goto** line 20. %% we have found an *i* which its *E*[*i*] is not smaller than current *Nk_max* so we choose it as a new element of G^D %%

      end if

    end if

  end for

  if *ind* is zero **then**

    assume that *q* is the first element of ***E*** which ***E***[*q*] is maximum, **set** $Z \leftarrow q$. %% we have examined all *i* values and there weren't *i* which its *E*[*i*] is equal or greater than current *Nk_max* so we choose *i* with the greatest *E*[*i*] as a new element of G^D %%

  end if

  for $0 \le j \le |D|-1$, if $R[Z \oplus D_j]$ is zero then add $Z \oplus D_j$ to $\{A\}_k^D$ and add D_j to $\{B\}_k^D$.

  return the value of *Z* as a new element of G^D indexed by k, G_k^D ; **for** each $0 \le j \le |D|-1$ **set** $R[Z \oplus D_j] \leftarrow 1$. %% elements of space *S* which can be generated by $G_k^D = Z$ are set to one in vector *R* %%

  set $Nk\_max \leftarrow E[Z];\ k \leftarrow k+1$. %% *Z* is the new element of G^D and we set its corresponding *E*[*Z*] as a new value for *Nk_max* %%

end while

B. Tables of round key differences for the attacks

Key differences for each of proposed attacks are provided in the three following tables. For example $(0,0,a_i,0)$ is corresponds to a 64-bit key difference which its value is a 16-bit value a_i in the 3th row and is zero in the other three rows. For more clarity, the difference of some rows is indicated for each 4-bit nibble. In such a case "s" is an unknown value specifies the output of S-box.

Table 5.a. (left), 5.b. (right) difference in round keys according to key differences Δ_i^K and ∇_j^K for the attacks on mCrypton-96 in Sections 4.3 and 4.5, respectively.

	Difference according to key difference Δ_i^K	Difference according to key difference ∇_j^K
ΔK_0	$(0,a_i,0,0)$	$(0,0,0,0)$
ΔK_1	$(0,0,a_i,0)$	$(s000,0s00,0,0)$
ΔK_2	$(0,0,0,a_i << 8)$	$(a_j << 3,0,0,0)$
ΔK_3	$(0,0,0,0)$	$(0,a_j << 3,0,0)$
ΔK_4	$(s000,0s00,00s0,000s)$	$(0,0,a_j << 3,0)$
ΔK_5	$(a_i << 11,0,0,0)$	$(0,0,0,a_j << 11)$
ΔK_6	$(0,a_i << 11,0,0)$	$(0,0,0,0)$
ΔK_7	$(0,0,a_i << 11,0)$	$(0,0s00,00s0,0)$
ΔK_8	$(0,0,0,a_i << 3)$	$(a_j << 14,0,0,0)$
ΔK_9	$(0,0,0,0)$	$(0,a_j << 14,0,0)$
ΔK_{10}	$(s000,0s00,00s0,000s)$	$(0,0,a_j << 14,0)$
ΔK_{11}	$(a_i << 6,0,0,0)$	$(0,0,0,a_j << 6)$
ΔK_{12}	$(0,a_i << 6,0,0)$	$(0,0,0,0)$

	Difference according to key difference Δ_i^K	Difference according to key difference ∇_j^K
ΔK_0	$(a_i,0,0,0)$	$(0,0,0,0)$
ΔK_1	$(0,a_i,0,0)$	$(s000,0s00,0,0)$
ΔK_2	$(0,0,a_i,0)$	$(a_j << 3,0,0,0)$
ΔK_3	$(0,0,0,a_i << 8)$	$(0,a_j << 3,0,0)$
ΔK_4	$(0,0,0,0)$	$(0,0,a_j << 3,0)$
ΔK_5	$(s000,0s00,00s0,000s)$	$(0,0,0,a_j << 11)$
ΔK_6	$(a_i << 11,0,0,0)$	$(0,0,0,0)$
ΔK_7	$(0,a_i << 11,0,0)$	$(0,0s00,00s0,0)$
ΔK_8	$(0,0,a_i << 11,0)$	$(a_j << 14,0,0,0)$
ΔK_9	$(0,0,0,a_i << 3)$	$(0,a_j << 14,0,0)$
ΔK_{10}	$(0,0,0,0)$	$(0,0,a_j << 14,0)$
ΔK_{11}	$(s000,0s00,00s0,000s)$	$(0,0,0,a_j << 6)$
ΔK_{12}	$(0,a_i << 6,0,0)$	$(0,0,0,0)$

Table 6. Difference in round keys according to key differences Δ_i^K and ∇_j^K for the attack on mCrypton-128 in Section 5

	Difference according to key difference Δ_i^K	Difference according to key difference ∇_j^K
ΔK_0	$(0,0,0,0)$	$(0,0,a_j,0)$
ΔK_1	$(0,a_i,0,0)$	$(0,0,0,0)$
ΔK_2	$(0,0,0,a_i)$	$(a_j,0,0,0)$
ΔK_3	$(0,0,0,0)$	$(0,0,0,a_j)$
ΔK_4	$(0,a_i << 8,0,0)$	$(0,0,0,0)$
ΔK_5	$(0,0,0,0)$	$(0,a_j << 8,0,0)$
ΔK_6	$(s000,0s00,00s0,000s)$	$(0,0,0,0)$

	Difference according to key difference Δ_i^K	Difference according to key difference ∇_j^K
ΔK_7	$(0,0,a_i << 11,0)$	$(s000,0s00,0,0)$
ΔK_8	$(0,0,0,0)$	$(0,0,a_j << 11,0)$
ΔK_9	$(a_i << 11,0,0,0)$	$(0,0,0,0)$
ΔK_{10}	$(0,0,0,a_i << 11)$	$(a_j << 11,0,0,0)$
ΔK_{11}	$(0,0,0,0)$	$(0,0,0,a_j << 11)$
ΔK_{12}	$(0,a_i << 3,0,0)$	$(0,0,0,0)$

C. Computation and Re-computation stages in the added rounds of proposed attacks

The following figures show the nibbles which their values is changed in the computation/re-computation process of the proposed attacks. Solid boxes (■) corresponds to the nibbles which are affected by S-box operation. Hashed boxes (▨) corresponds to the nibbles which are affected by a linear or shift operation. White boxes with a circle within (⊡) are the nibbles which their values are changed but are not interested. White boxes are the nibbles we don't care about their values.

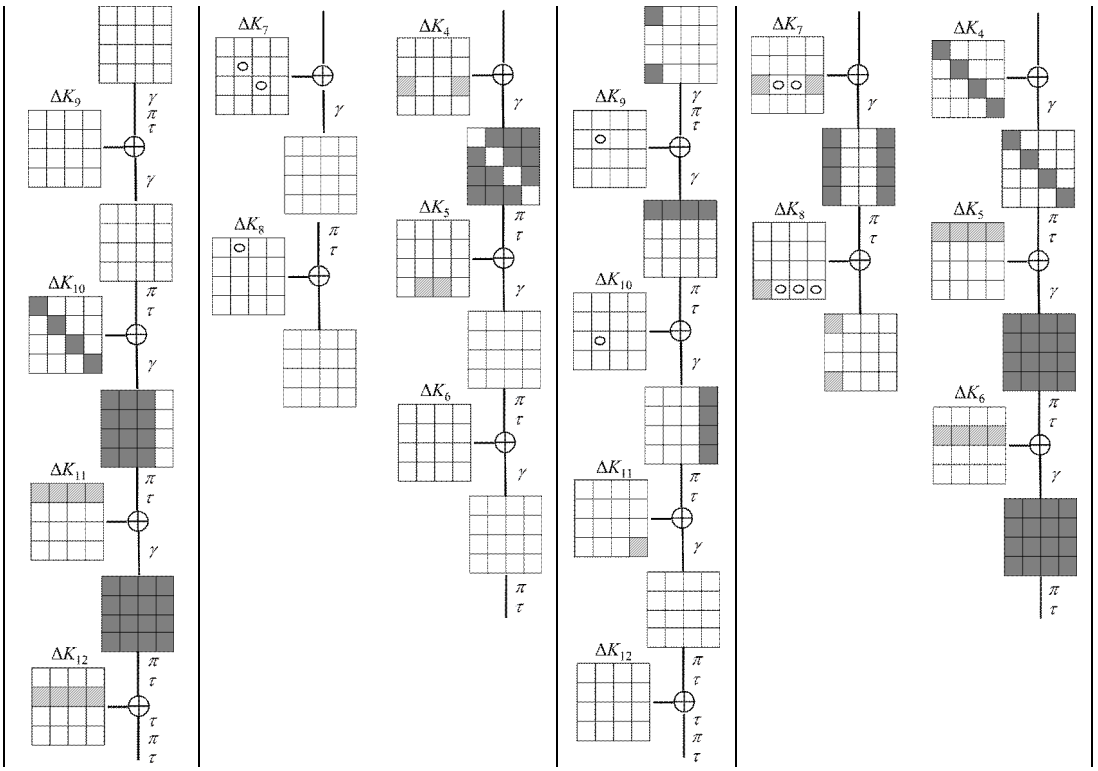

Fig. 6.a. (left), 6.b. (right) pre-computation stage for the attack in Section 4.3 according to key differences Δ_i^K and ∇_j^K, respectively

Fig. 7.a. (left), 7.b. (right) re-computation stage for the attack in Section 4.3 according to key differences ∇_j^K and Δ_i^K, respectively

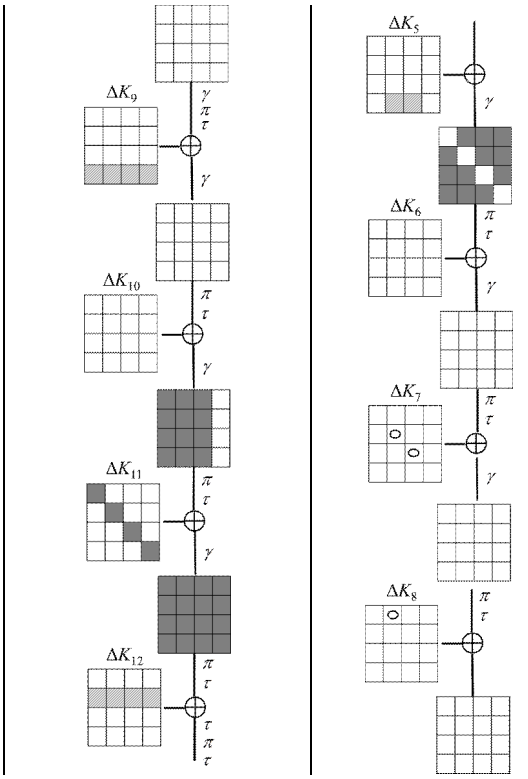

Fig. 8.a. (left), 8.b. (right) Computation stage for the attack in Section 4.5 according to key differences Δ_i^K and ∇_j^K, respectively

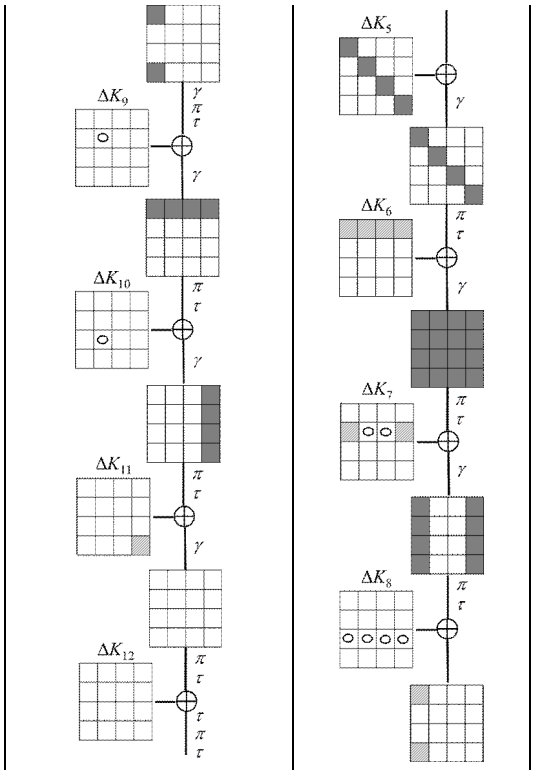

Fig. 9.a. (left), 9.b. (right) Re-computation stage for the attack in Section 4.5 according to key differences ∇_j^K and Δ_i^K, respectively

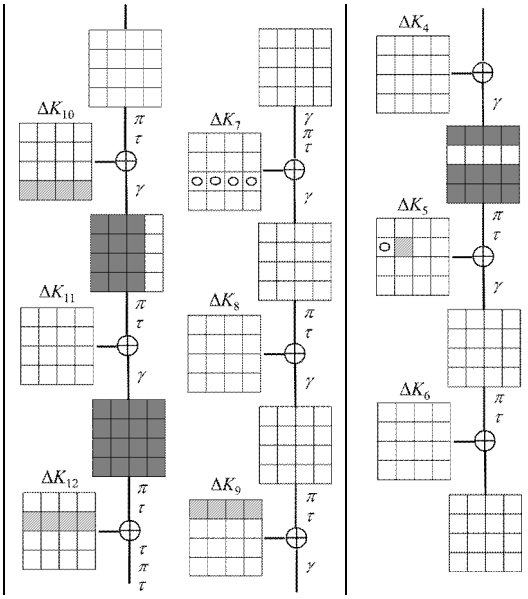

Fig. 10.a. (left), 10.b (right) Computation stage for the attack in Section 5 according to key differences Δ_i^K and ∇_j^K, respectively

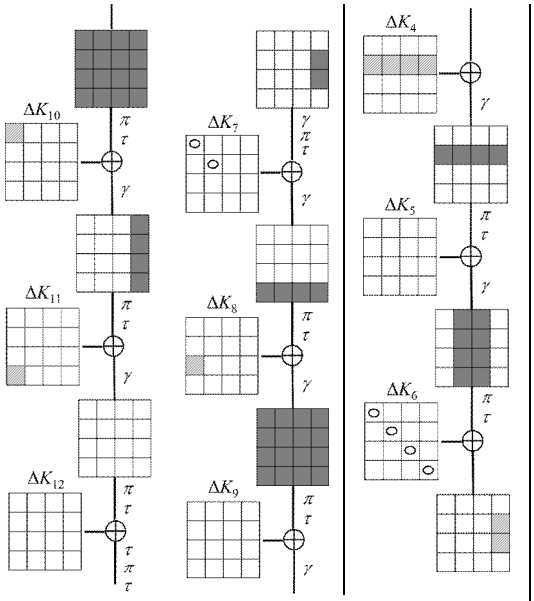

Fig. 11.a. (left), 11.b. (right) Re-computation stage for the attack in Section 5 according to key differences ∇_j^K and Δ_i^K, respectively