Low Data Complexity Biclique Cryptanalysis of Block Ciphers with Application to Piccolo and HIGHT

Siavash Ahmadi, Zahra Ahmadian, Javad Mohajeri, and Mohammad Reza Aref

Sharif University of Technology, Tehran, Iran.
{s_ahmadi,ahmadian}@ee.sharif.edu,{mohajer,aref}@sharif.edu

Abstract. In this paper, we present a framework for biclique cryptanalysis of block ciphers with an extremely low data complexity. To that end, we enjoy a new representation of biclique attack. Then an algorithm for choosing two differential characteristics is also presented to simultaneously minimize the data complexity and control the computational complexity.
Then we characterize those block ciphers that are vulnerable to this technique and among them, we apply this attack on lightweight block ciphers Piccolo-80, Piccolo-128 and HIGHT. The data complexities of these attacks are considerably less than the existing results. For full-round Piccolo-80 and 128, the data complexity of the attacks are only 16 plaintext-ciphertext pairs and for full-round HIGHT our attack requires 256 pairs. In all attacks the computational complexity remains the same as the previous ones or even it is slightly improved.

Keywords: Biclique Cryptanlysis, Attack Complexity, Lightweight Block Ciphers

1 Introduction

Amongst the current methods for security evaluation of block ciphers, biclique attack has attracted lots of attention since it could be applied on the full version of those block ciphers on which many other existing attacks did not work [1–10]. In spite of most of the statistical attacks whose computational complexities increase exponentially with the number of rounds, biclique attack is inherently an accelerated exhaustive search (that is why it can break the full-round cipher) in which some optimizations are made in two directions:

1. Use of a biclique structure that reduces the meet in the middle part and partitions the key space into some groups of some related keys.
2. Exploiting the similarity of the keys to be tested in each key group to avoid a limited number of recomputations.

So due to the exhaustive search essence of this attack, there is no surprise that computational complexity is very close to brute force attack, but in many cases,

beside the marginal computational complexity, the data complexity is not practical as well.

On the other hand, in order to adhere to "real-life" scenarios, it is encouraged that some restrictions on the resources available to the adversary is taken into account [11, 12]. A direction of research could be the situation where the computational power of the attacker is assumed to be bounded by the exhaustive search, but the data complexity is restricted to a few known or chosen plaintexts. The importance of this scenario is apparent when a lightweight block cipher is targeted by the attacker where due to the small size of the cipher key (typically 64 bits), an accelerated exhaustive search might not be too far from reality, under a practical assumption for the attacker's computational budget. But, the cryptographic protocol in which that primitive is used, might not allow the attacker to eavesdrop on an enormous amount of plaintext-ciphertext pairs.

Motivated by this approach, we propose a new variant of biclique attack, namely Low Data Complexity (LDC) biclique attack, that aims to confine the amount of data required for the attack to a practical value without any expenses in the computational complexity. Out results shows that compared to the conventional biclique attacks the computational complexity of this attack is not increased or even it is improved slightly. There are a few work on reducing the data complexity of biclique attack [13, 14] both on AES. The approach of [13] is reducing the biclique dimension to 4 while the attack in [14] uses the same dimension biclique, i.e. 8, but with a shorter biclique.

In a nutshell, block ciphers with round keys shorter than the master key (usually generalized feistel ciphers) which have a word-wise permutation-like key schedules are potential targets for this attack. In order to confine the data complexity to a practical value, we enjoy the asymmetric biclique in a special form, in which the number of vertexes in the plaintext (or ciphertext) side of the biclique is the square of that in the other side (i.e. 2^d and 2^{2d} respectively, for some d). The other trick that we used is the computations in the matching part that proceeds in 3 levels: computations that are performed once, those performed 2^d times, and those performed for 2^{2d} times.

Using this method, we propose new attacks on the full-round Piccolo-80, Piccolo-128 [15] and HIGHT [16] block ciphers. Our results show a significant reduction in data complexity: in case of Piccolo-80 and Piccolo-128 the amount of data required by the attack is just 16 pairs of plaintext-ciphertext and in case of HIGHT, it is 256 pairs. It is worth noting again that computational complexities of our attacks does not exceeds the previous ones, even there are slight improvements in computational complexity in the case of Piccolo-80 and Piccolo-128 achieved. A summary of the previous biclique attacks on full-round Piccolo and HIGHT as well as our results is reported in Table 1.

The outline of the paper is as follows: Section 2 presents the biclique attack puzzle. Section 3 presents the method used for low data complexity biclique cryptanalysis. The block ciphers vulnerable to this technique are also characterized in this section. Section 4 presents brief descriptions of Piccolo and HIGHT,

Table 1. Summary of cryptanalytic results of biclique attacks on Piccolo and Hight

Block cipher	Rounds	Data	Computations	Biclique length	Reference
Piccolo-80	25*	2^{48}	$2^{78.95}$	6	[3]
	Full (25)	2^{48}	$2^{79.13}$	6	[4]
	Full (25)	2^{48}	$2^{79.34}$	4	[5]
	Full (25)	$\mathbf{2^4}$	$\mathbf{2^{79.07}}$	**3**	**This paper**
Piccolo-128	28	2^{24}	$2^{126.79}$	6	[3]
	Full (31)	2^{24}	$2^{127.35}$	7	[4]
	Full (31)	2^{48}	$2^{127.36}$	4	[5]
	Full (31)	$\mathbf{2^4}$	$\mathbf{2^{127.12}}$	**5**	**This paper**
Hight	Full (32)	2^{48}	$2^{126.4}$	8	[6]
	Full (32)	2^{48}	2^{126**}	9	[5]
	Full (32)	$\mathbf{2^8}$	$\mathbf{2^{126.07}}$	**5**	**This paper**

*Without postwhitening key
**The corrected computations of the attack in [5].

respectively. We apply our attacks on Piccolo-80, Piccolo-128 and HIGHT block ciphers in Section 5. Finally we conclude our work in Section 6.

2 Biclique Attack Puzzle

In the biclique attack, the biclique can be constructed either in the plaintext or ciphertext side. In this section, we explain the attack steps in the case that the biclique is constructed in the plaintext side. We consider an asymmetric biclique in which the number of vertexes in the two sides are not equal [9]. More precisely, we enjoy an special case of this structure in which the number of vertexes in the internal state side of the biclique is 2^{2d} while it is 2^d in the plaintext side. The four main steps of this attack are as follows.

2.1 Key partitioning

Let us divide the master key K into four disjoint sets of bits, namely K^{f_0}, K^{f_1}, K^b and K^g, with d, d, d and $n-3d$ bit lengths, respectively. We can partition the key space into 2^{n-3d} groups of keys where the value of K^g is fixed in each group (and hence enumerates the groups), and K^{f_0}, K^{f_1} and K^b take all possible values. All the keys in a group are denoted by $K[i,j]$ where $(K^{f_1}||K^{f_0}) = j$, $j \in \{0,...,2^{2d}-1\}$ and $K^b = i$, $i \in \{0,...,2^d-1\}$. Then, the next three steps are carried out for each group. We also define the differentials $\nabla_i^K = K[0,0] \oplus K[i,0]$ and $\Delta_j^K = K[0,0] \oplus K[0,j]$.

2.2 Asymmetric biclique constructing

Definition 1. $(\mathbf{d}, \mathbf{2d})$***-dimensional asymmetric biclique.*** *The 3-tuple* $[\{P_i\},\{S_j\},\{K[i,j]\}]$ *is called a* $(d,2d)$*-dimensional asymmetric biclique with length* l*, if* $\forall i \in \{0,1,...,2^d-$

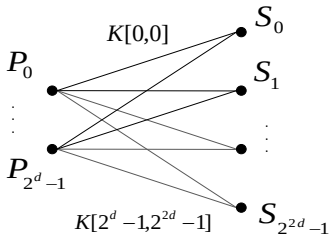

Fig. 1. $(d, 2d)$-Dimensional asymmetric biclique in plaintext side

$1\}$ *and* $\forall j \in \{0, 1, ..., 2^{2d-1}\}$*:*

$$P_i \xrightarrow[0,l-1]{K[0,j]} S_j \tag{1}$$

Where $\{P_i\}$ *is a set of* 2^d *plaintexts,* $\{S_j\}$ *is a set of* 2^{2d} *internal states and* $\xrightarrow[a,b]{K}$ *denotes the encryption with key* K *from round* a *to round* b ($\xleftarrow[a,b]{K}$ *stands for the corresponding decryption).*

The structure of the $(d, 2d)$-dimensional asymmetric biclique is shown in Fig. 1.

The two groups of vertexes in the graph stand for some plaintexts and some internal states of the cipher respectively, while the edges stand for the keys under which the encryption of the plaintext yields the corresponding internal state. All the 2^{3d} keys in a group defined in section 2.1 should fit into an asymmetric biclique. The most common method for constructing the biclique is independent biclique method [1]:

- Step 1. Choose a random plaintext P_0 and compute S_0 as $P_0 \xrightarrow[0,l-1]{K[0,0]} S_0$
- Step 2. Compute P_i as $P_i \xleftarrow[0,l-1]{K[i,0]} S_0$ for all $i \in \{1, ..., 2^d - 1\}$.
- Step 3. Compute S_j as $P_0 \xrightarrow[0,l-1]{K[0,j]} S_j$ for all $j \in \{1, ..., 2^{2d} - 1\}$.

It can be shown that (1) is satisfied for the above-generated 3-tuple $[\{P_i\}, \{S_j\}, \{K[i, j]\}]$, if the related key differential characteristic Δ_j^K in forward direction does not share any active nonlinear component with the related key differential characteristic ∇_i^K in backward direction within rounds 0 to $l - 1$.

The data complexity of the attack is upper bounded by all the possible values of active plaintext bits in the backward differential characteristic $P_i \xleftarrow[0,l-1]{K[i,0]} S_0$, provided that all the bicliques are constructed from the same P_0.

2.3 Partial matching with precomputation and recomputation

Partial matching with precomputation and recomputation is the procedure in which all the keys in a group are tested in an efficient way [1]. In the case that

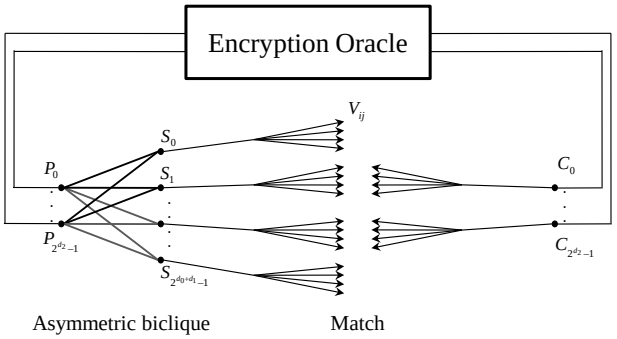

Fig. 2. Asymmetric biclique cryptanalysis

biclique is constructed at the plaintext side, partial matching is performed at the ciphertext side. The intermediate variable V, is selected in an appropriate position between round l and the last round. In forward direction, we partially encrypt S_j under key $K[0,j]$ to drive the matching variable in forward direction $\overrightarrow{V_{0,j}}$, and also save all the intermediate states associated to this computation. Similarly, in backward direction, we partially decrypt C_i under key $K[i,0]$ to drive the matching variable in backward direction $\overleftarrow{V_{i,0}}$, and save again all the intermediate states associated to this computation.

Now suppose that we want to check $K[i,j]$. In forward direction, for finding $\overrightarrow{V_{i,j}}$ by encrypting S_j under key $K[i,j]$, $i \neq 0$, we only need to recompute those bytes between S and V that are influenced by K^b when i changes while the other bytes are not recomputed (i.e. the active bytes in ∇_i differential characteristic in the forward direction in the matching part are computed 2^d times and the other ones are computed only once). In backward direction, for finding $\overleftarrow{V_{i,j}}$ by decrypting C_i under key $K[i,j]$, $j \neq 0$, we only need to recompute those bytes between C and V that are influenced by $K^{f_0}||K^{f_1}$ when j changes (i.e. the active bytes in Δ_j differential characteristic in the backward direction in the matching part). But, there are some bytes which are influenced either by K^{f_0} only or by K^{f_1} only. Such bytes can be recomputed only 2^d times and just the bytes influenced by both of K^{f_0} and K^{f_1} should be recomputed 2^{2d} times. This method has shown some improvement in the computational complexity of the biclique attack on some algorithms. Finally, If $\overrightarrow{V_{i,j}} = \overleftarrow{V_{i,j}}$, key $K[i,j]$ will be saved as a right key candidate.

The memory required for the attack is bounded by the amount of memory required for saving 2^{2d} intermediate states for an encryption algorithm.

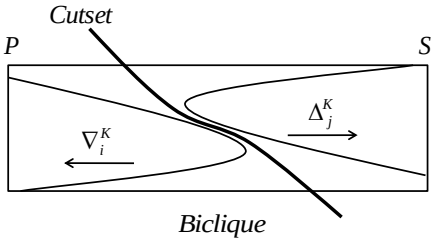

Fig. 3. A cutset in the biclique

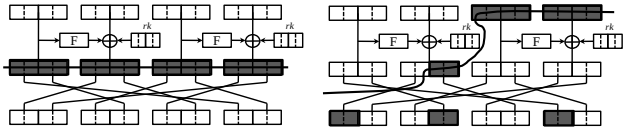

Fig. 4. Example of two cutsets

2.4 Rechecking the candidate keys

Finally, we test the candidate keys by a valid (P, C) pair to filter out all the wrong keys and find the correct key.

The total scheme of the asymmetric biclique cryptanalysis of a block cipher is shown in Fig. 2. For the biclique in the plaintext side, this attack is a chosen plaintext attack in the single key model and needs only the encryption oracle (For the biclique in the ciphertext side, it is a chosen ciphertext attack and needs the decryption oracle).

3 Low Data Complexity Biclique Cryptanalysis

3.1 A New Representation for Biclique Attack

In this section, we propose a new representation of biclique that is very useful in describing the algorithm of LDC biclique attack. First, some preliminaries and definitions are stated.

Definition 2. ***Cutset.*** *Let b be the block size. A b-bit part of the intermediate states of the cipher from which the plaintext and ciphertext can be calculated completely, provided that the cipher key is known, is called a cut-set.*

Plaintext and ciphertext are two trivial examples of cutset. Two other instances are shown in Fig. 4

Definition 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two cutsets in a block cipher and $g(\cdot)$ be the partial encryption/decryption function from one cutset to another. $L(g)$ is defined as the amount of computations involved in g.*

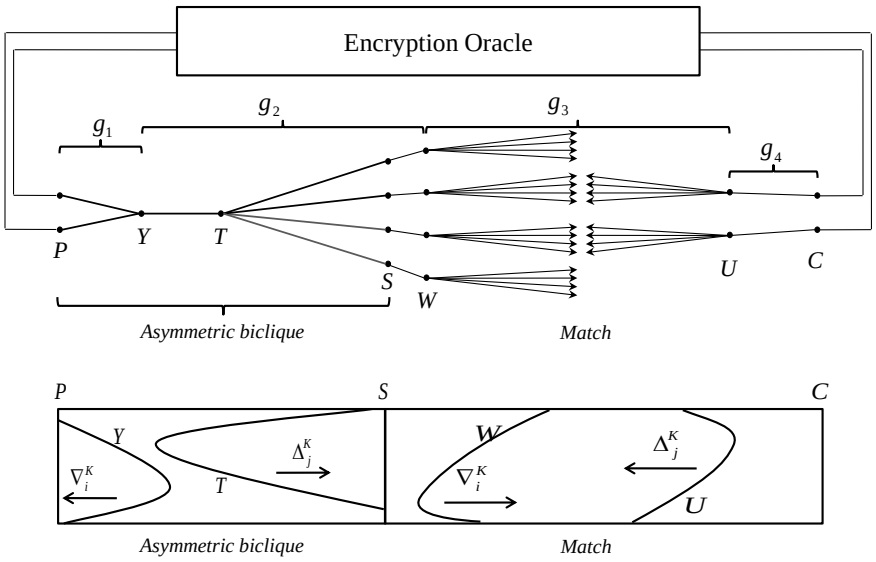

Fig. 5. Asymmetric biclique cryptanalysis model

It is conventional in biclique attack to estimate this value by the number of computationally dominate component of the algorithm.

Definition 4. *The **distance** between two cutsets $\mathcal{A}$ and $\mathcal{B}$ is defined based on the value of the associated $L(g)$.*

It can be shown that for an independent biclique, there exists at least a cutset among the biclique rounds, in which all bits are unaffected by ∇_i^K differentials in backward direction and Δ_j^K differentials in forward direction (Fig. 3). This is exactly equivalent to the independence of the two characteristics. We call such cutsets the *root cutset* of the biclique, since they are common in all paths $P_i \xrightarrow[0,l-1]{K[i,j]} S_j$. In the other words, all the P_i and S_j can be generated from that cutset along with the associated related key differential ∇_i^K and Δ_j^K, respectively. Thus, the biclique topology reduces to that shown in Fig. 5. In this figure, all the points between $\mathcal{Y}$ and $\mathcal{T}$ are actually a root cutset.

In some cases these cutsets can be seen clearly, while in some other cases they may not be seen explicitly. The root cutset for the AES-128 independent biclique in [1] is shown in Fig. 6 with light gray boxes.

This representation of biclique attack is very useful in describing the algorithm of LDC biclique attack presented in the following.

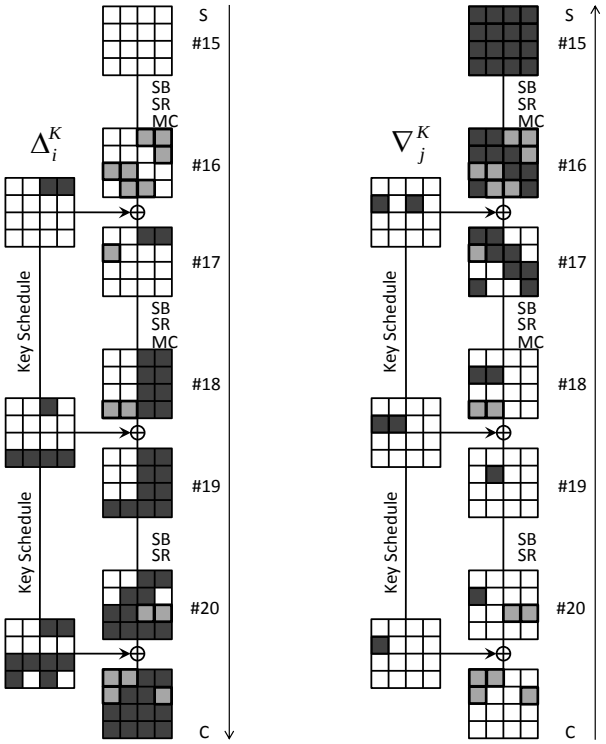

Fig. 6. Independent biclique of AES-128. The differntial characteristics are shown in dark gray. The set of light gray boxes makes the root cutset

3.2 Algorithm of Choosing Two Differntials

Suppose that S is the internal state of the round on which the ∇_i^K differentials in forward direction in matching part does not affect any. Let $\mathcal{Y}$ be the closest cutset to the plaintext that is not affected by the ∇_i^K differential in backward direction in the biclique, $\mathcal{T}$ be the closest cutset to S that is not affected by Δ_j^K differential in forward direction in the biclique, $\mathcal{W}$ be the farthest cutset from S that is not affected by ∇_i^K differentials in forward direction in the partial matching, and $\mathcal{U}$ be the farthest cutest from C that is not affected by the Δ_j^K differentials in backward direction in the partial matching. All the above defined are shown in Fig. 5.

Let g_1 be equal to partial encryption from P to $\mathcal{Y}$, g_2 be equal to partial encryption from $\mathcal{Y}$ to $\mathcal{W}$, g_3 be equal to partial encryption from $\mathcal{W}$ to $\mathcal{U}$, and g_4 be equal to partial encryption from $\mathcal{U}$ to C (see Fig. 5). We define g_i^{-1} as the associated decryption algorithm of g_i.

if ∇_i^K differential is selected in such way that the common bits between $\mathcal{Y}$ and plaintext cutset P are as many as possible, the data complexity of the attack

will reduce as much as possible, as well. This is the key idea behind the low data complexity biclique attack. In fact, the diffusion of ∇_i^K backward differential in biclique in the plaintext directly determines the data complexity of the attack, and it exclusively depends on g_1^{-1}diffusion. So, $L(g_1^{-1})$ should be minimized. On the other hand, the longer biclique, the less computational complexity; So, $L(g_2)$ should be maximized, as well. Note that choosing ∇_i^K differential solely determines the exact value of $L(g_1)$ and $L(g_2)$ or S. So, Here we are actually faced with the following multi objective optimization problem:

$$\begin{aligned} &\min\ L(g_1)\ ,\ \max\ L(g_2) \\ &\text{s.t. all possible } \nabla_i^K \end{aligned} \tag{2}$$

The optimum point for (2), ∇_i^{K*}, guarantees the lowest possible data complexity and the longest possible biclique length at the same time. Now there is still room for reducing the computational complexity by an appropriate choose of Δ_j^K.

In order to reduce the computational complexity, we should find Δ_j^K differential which has maximum $L(g_4^{-1})$ and, at the same time, does not share any active nonlinear component in the biclique part with ∇_i^{K*} differential (it means that the cutset $\mathcal{T}$ can be driven only by *encrypting* the cutset $\mathcal{Y}$). Since $L(g_1)$ has been minimized, there is not so limitation in choosing Δ_j^K differentials and there is enough room for choosing an appropriate one that most reduces the computational complexity by maximizing $L(g_4^{-1})$. So, after solving (2) the following optimization problem should be solved:

$$\begin{aligned} &\max\ L(g_4^{-1}) \\ &\text{s.t. } \Delta_j^K \perp\!\!\!\perp \nabla_i^{K*} \end{aligned} \tag{3}$$

where $\perp\!\!\!\perp$ is the symbol for the independence of two characteristics.

The presented algorithm actually breaks the problem of simultaneous search of the two characteristics Δ_j^K and ∇_i^K in the conventional biclique attack into the two separate searches of first ∇_i^K, then Δ_j^K. It is clear that it is impossible to search all the possible values of Δ_j^K and ∇_i^K differentials, but this algorithm gives the cryptanalyst a deeper perspective for finding appropriate differential for the biclique attack.

3.3 The Potential Vulnerable Algorithms to This Method

For an efficient attack, it is required that some consecutive rounds of the algorithm are independent of some parts of the master key. In more details, we require that g_2 is independent of K^b according to the definition of $\mathcal{W}$ and $\mathcal{Y}$ and g_4 is independent of K^f, according to the definition of $\mathcal{U}$. The farther $\mathcal{W}$ and $\mathcal{Y}$, and also the farther $\mathcal{U}$ and C result in a more efficient attack.

One can immediately concludes that this requirement implies that the round key of the target block cipher must be shorter than the master key, that is a

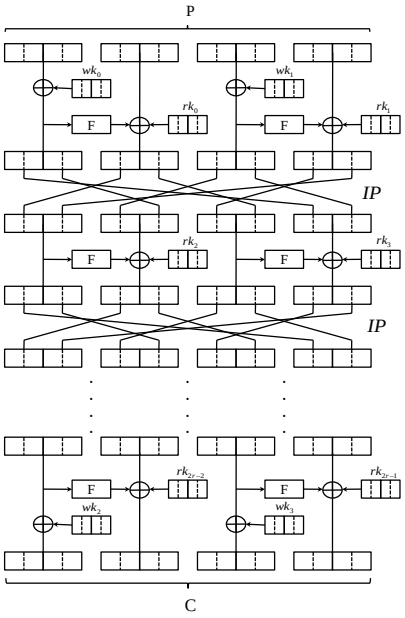

Fig. 7. Piccolo block cipher

necessary condition for not depending the round keys on all bits of the master keys. Generalized feistel structures typically have such a property.

The other condition is on the key schedule. To fulfil the independence of g_4 from K^f, the diffusion of the key schedule should not be too high to mix all bits of the master key in the last rounds. So, the key schedules whose operations are confined to word-wise permutations, Sbox operations, and eventually addition to constants are very vulnerable to this attack.

We found Piccolo family of block ciphers and HIGHT block cipher proper targets for this attack. In the following sections we briefly describe these algorithms and then, examine the LDC biclique attack method on them.

4 Brief Description of Piccolo and Hight

In this section we briefly describe Piccolo and HIGHT block ciphers. For more details about these two algorithms refer to [15, 16].

4.1 Piccolo-80 and 128

Notations. Let k_i be the i^{th} 16-bit part of the master key K counting from left. Then, we call the left and right halves of k_i, k_i^L and k_i^R, respectively. Also, we call the left and right nibbles of k_i^R, $k_i^{R_0}$ and $k_i^{R_1}$, respectively. The same notation is used for k_i^L. Round i of the algorithm uses two 16-bit subkeys namely

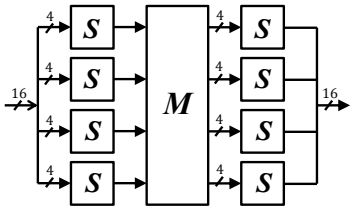

Fig. 8. Structure of Piccolo F-function [15]

(rk_{2i}, rk_{2i+1}). Two 16-bit prewhitening keys (wk_0, wk_1), and postwhitening keys (wk_2, wk_3) are also used in the first and last rounds of the algorithm, respectively.

Specifications. Piccolo citepiccolo is a lightweight block cipher with a generalized Fiestel structure. It has two versions Piccolo-80 and Piccolo-128 with 80 and 128-bit key sizes, and 25 and 31 rounds, respectively. Both versions have 64-bit block size. Each round i consists of two nonlinear F-functions ($F : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$), an internal permutation ($IP : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$), and a addition to subkeys (rk_{2i}, rk_{2i+1}). Fig.7 shows Piccolo block cipher in detail.

Piccolo F-function. F-function of Piccolo is a SDS network consisting of an Sbox layer with four parallel 4-bit Sboxes, followed by a linear matrix M, and finally another Sbox layer (Fig. 8). Since the computational complexity for complete calculation of a single F-function has the dominate complexity compared to the other operations (XOR and IP) in the algorithm, all the computations of the attack are estimated by the number of F-functions to be calculated. In the matching step, we will also require to compute only half of the F-function's output bits that involves computing the four input Sboxes and two output Sboxes. Hence, its complexity is equal to 3/4 F-function. Moreover, if half (or one fourth) of the input bits are active, the complexity will be equal to 3/4 F-function (5/8 F-function), too.

Key schedule. Piccolo-80 and 128 have simple linear key schedules that are summarized in Tables 2 and 3 of appendix A. Ignoring the constants additions, the key schedule of Piccolo can be regarded as a word-wise permutation-based key schedule.

4.2 HIGHT

Notations. Let k'_i be the i^{th} bytes of the master key K counting from right.
Round i of the algorithm uses four 8-bit subkeys namely $sk_{4i+3}, sk_{4i+2}, sk_{4i+1}, sk_4$.
Four 8-bit prewhitening keys (wk_0, wk_1, wk_2, wk_3), and postwhitening keys (wk_4, wk_5, wk_6, wk_7), are also used in the first and last rounds of the algorithm, respectively.

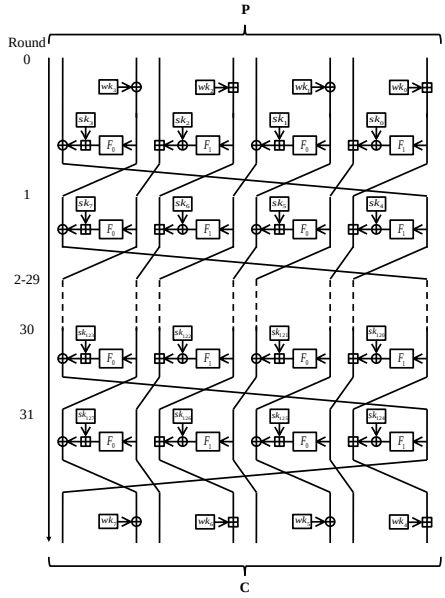

Fig. 9. HIGHT block cipher

Specifications. HIGHT [16] is a lightweight block cipher with generalized Fiestel structure. It has 64-bit block size, 128-bit key length, and 32 rounds encryption. Each round i consists of two linear functions ($F_0 : \{0,1\}^9 \rightarrow \{0,1\}^8, F_1 : \{0,1\}^9 \rightarrow \{0,1\}^8$), an internal permutation ($IP : \{0,1\}^{64} \rightarrow \{0,1\}^{64}$) and, four XOR and modular addition ($mod\ 2^8$). Fig. 9 shows HIGHT block cipher in detail. Both F_0 and F_1 are referred to as F-function.

F-functions. The F_0 and F_1 functions have linear structure as follows:

$$\begin{aligned} F_0(x) &= x \lll 1 \oplus x \lll 2 \oplus x \lll 7 \\ F_1(x) &= x \lll 3 \oplus x \lll 4 \oplus x \lll 6 \end{aligned} \tag{4}$$

Since any F-function is followed by a XOR and a modular addition operation, counting the number of F-function calculations in the attack procedure is a good criteria for the attack computations.

Key schedule. HIGHT has a simple key schedule that is summarized in Tables 4 of appendix A. Regardless some modular additions to round constants, HIGHT key schedule can be regarded as a byte-wise permutation-based key schedule, too.

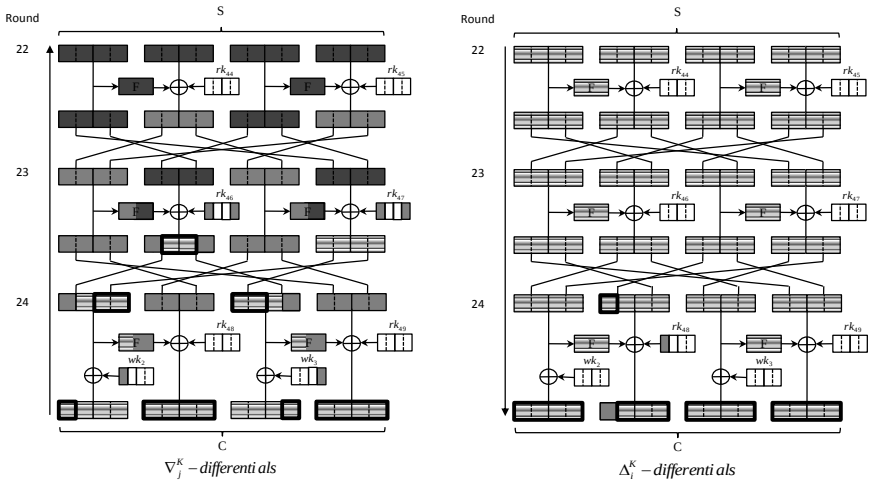

Fig. 10. 3-Round (4,8)-dimensional asymmetric biclique constructing for Piccolo-80

5 Low Data Complexity Biclique cryptanalysis of Piccolo-80, Piccolo-128 and Hight

5.1 Attack on Piccolo-80

With use of LDC biclique attack algorithm described in section 3.2 in a scenario that biclique is constructed in the ciphertext side, we obtained $K^f = k_0^{L_0}, K^{b_0} = k_4^{L_0}$, and $K^{b_1} = k_4^{R_1}$ as a good key partitioning. A 3-round (4,8)-dimensional asymmetric biclique at the ciphertext side is constructed on rounds 22 to 24. Thus, the intermediate state S refers to the input state of round 22. As it can be seen in Fig. 10, the ∇_j^K and Δ_i^K differentials are independent within the last three rounds of the algorithm. The cutsets $\mathcal{Y}$ and $\mathcal{T}$ are those boxes with thickened borders in right and left part of Fig 10. Furthermore, note that there are no active F-functions in the Δ_i^k differential characteristic in biclique part. This provided us with maximum degree of freedom for selecting K^{b_0} and K^{b_1} that is chosen in such way that maximizes $L(g_4)$ in the optimization problem (3) (see Fig. 11). The cutsets $\mathcal{U}$ and $\mathcal{W}$ are the boxes with thickened borders in left and right part of Fig. 11.

Data Complexity As we can see in Fig.9, $C_i = C_0 \oplus \Delta_i^K$. Therefore without any computations, C_i are known all. So, the data complexity is exactly 2^4 plaintext-ciphertext pairs.

Computational Complexity Computational Complexity of the attack is three-fold: biclique, partial matching and candidate key testing complexities:

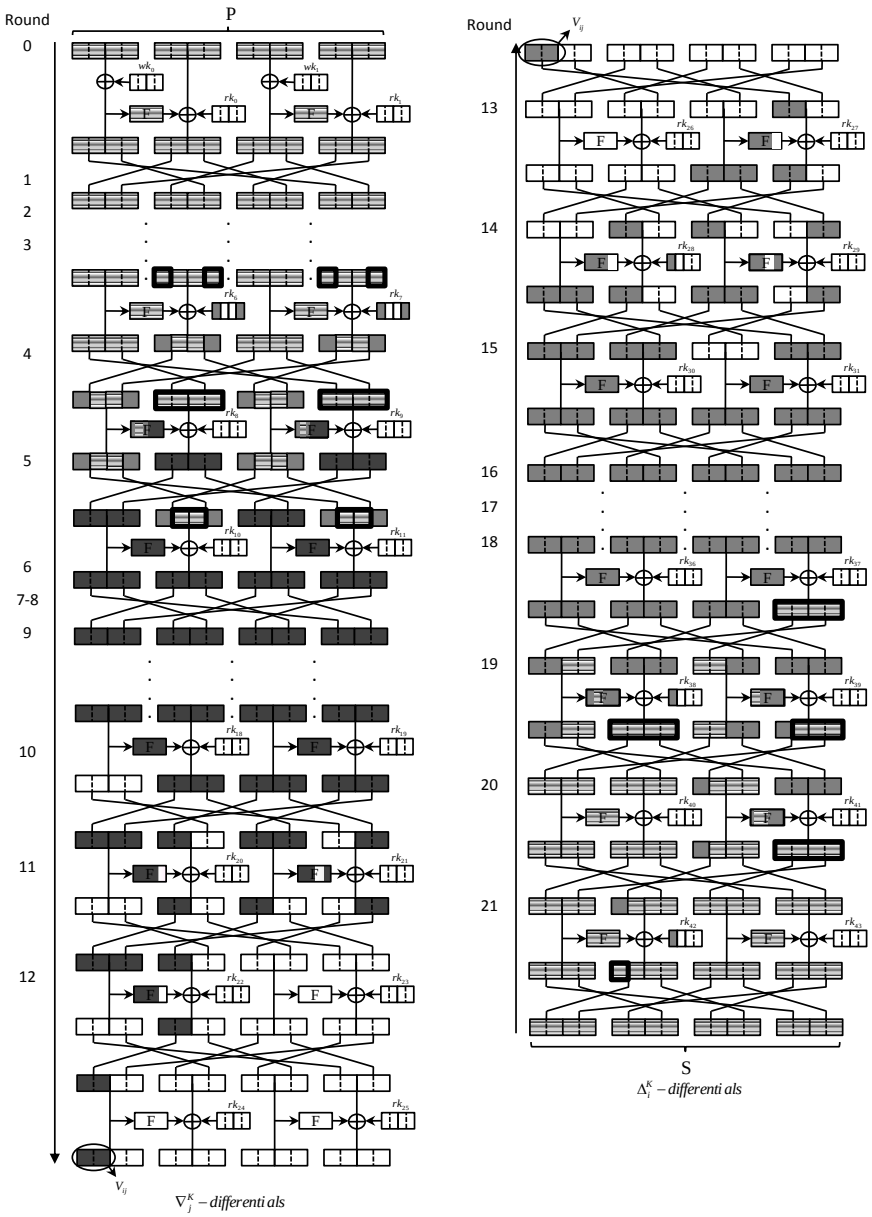

Fig. 11. Partial Matching for Piccolo-80

Biclique complexity. In order to compute S_j in a biclique, (Fig. 10, left), 0.75 F-function should be calculated once (the gridded bytes), 2.25 F-function should be calculated 2^4 times (the light gray bytes), and 3 F-functions should be cal-

culated 2^8 times (the dark gray bytes). So, the normalized computational complexity of this step is (a 25-round encryption of Piccolo-80 is taken as the unit of computation which is equivalent to 50 F-function computations):

$$C_{biclique} = \frac{0.75 + 2.25 \times 2^4 + 3 \times 2^8}{50} = 2^{4.01} \tag{5}$$

Matching complexity. The left most byte of the input of IP layer in round 12 is chosen as the matching variable V. In backward direction of partial matching (rounds 21 to 13) for each S_j, 3.875 F-functions should be calculated once and 12.375 F-functions should be calculated 2^4 times (see Fig. 11, right; The bytes in white is not required to be computed). Also, in forward direction of partial matching (rounds 0 to 12) for each P_i, 8.5 F-functions should be calculated once, 0.5 F-function should be calculated 2^4 times, and 13.25 F-functions should be calculated 2^8 times (see Fig. 11, left). Hence, the computational complexity for checking all the keys in a group normalized to a full-round encryption of Piccolo-80 is:

$$\begin{aligned} C_{backward} &= \frac{2^8(3.875 + 12.375 \times 2^4)}{50} = 2^{10.01} \\ C_{forward} &= \frac{2^4(8.5 + 0.5 \times 2^4 + 13.25 \times 2^8)}{50} = 2^{10.09} \end{aligned} \tag{6}$$

Hence,

$$C_{match} = C_{backward} + C_{forward} = 2^{11.05} \tag{7}$$

Candidate keys testing. By using of the 8-bit matching variable, the probability of accepting a wrong key is 2^{-8}. Also, we check 2^{12} keys in each group. So, the computational complexity of rechecking the false keys is:

$$C_{recheck} = 2^{12} \times 2^{-8} = 2^4 \tag{8}$$

Since all steps are executed for each group, the total computational complexity of the attack is:

$$\begin{aligned} C_{total} &= 2^{68} \times (C_{biclique} + C_{match} + C_{recheck}) \\ &= 2^{68} \times (2^{4.01} + 2^{11.05} + 2^4) = 2^{79.07} \end{aligned} \tag{9}$$

5.2 Attack on Piccolo-128

Here, we obtained $K^{f_0} = k_1^{R_0}, K^{f_1} = k_1^{R_1}$, and $K^b = k_0^{L_0}$ for a 5-round (4,8)-dimensional asymmetric biclique at the plaintext side for rounds 0 to 4. Thus, the intermediate state S refers to the output state of round 4. Fig. 12 shows the biclique part of the attack and Fig. 13 shows the matching part. The associated cutsets $\mathcal{Y}, \mathcal{T}, \mathcal{W}, \mathcal{U}$ are shown as the same manner with Piccolo-80, in Figurs 12 and 13.

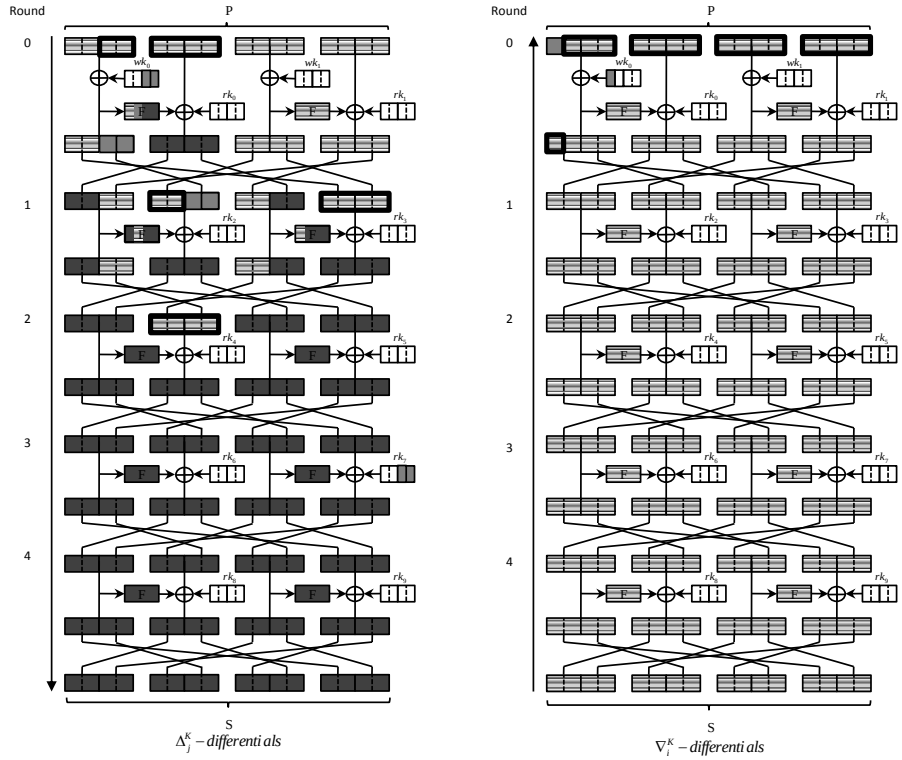

Fig. 12. 5-Round (4,8)-dimensional asymmetric biclique constructing for Piccolo-128

Data complexity As we can see in Fig. 12, $P_i = P_0 \oplus \nabla_i^K$. Thus, P_i are known all, without any computations. So, the data complexity is exactly 2^4 plaintext-ciphertext pairs.

Computational complexity

Biclique complexity In order to compute S_j in a biclique, 1.75 F-function should be calculated once, 0.25 F-function should be calculated 2^4, and 8 F-functions should be calculated 2^8 times (see Fig. 12, left). So, the normalized computational complexity of asymmetric biclique constructing is:

$$C_{biclique} = (1.75 + 0.25 \times 2^4 + 8 \times 2^8)/62 = 2^{5.05} \tag{10}$$

Matching complexity. The left most byte of the input of IP layer in round 16 is chosen as the matching variable V. In backward direction of partial matching (rounds 30 to 17) for each C_i, 9.75 F-functions should be calculated once, 0.25

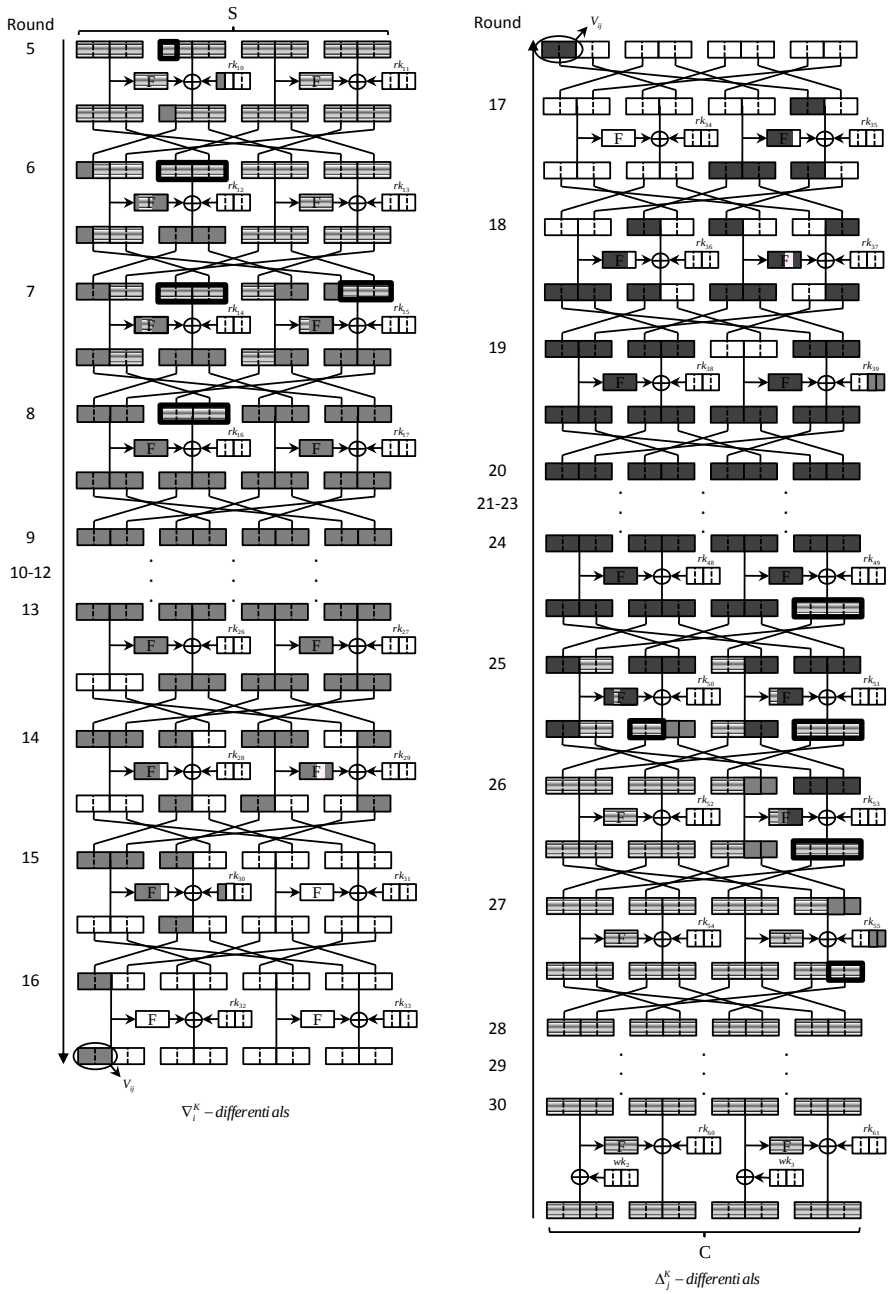

Fig. 13. Partial Matching for Piccolo-128

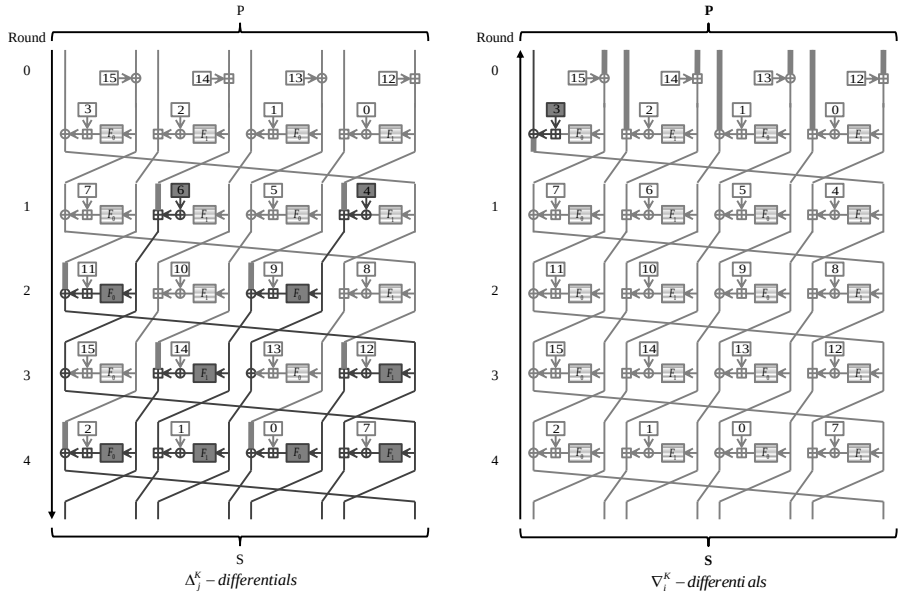

Fig. 14. 5-Round (8,16)-dimensional asymmetric biclique constructing for HIGHT

F-function should be calculated 2^4 times, and 16.25 F-functions should be calculated 2^8 times (see Fig. 13, right). Also, in forward direction of partial matching (rounds 5 to 16) for each S_j, 3.875 F-functions should be calculated once and 16.375 F-functions should be calculated 2^4 times (see Fig. 13, left). Hence, the computational complexity for checking all the keys in a group normalized to a full-round encryption of Piccolo-128 is:

$$\begin{aligned} C_{forward} &= \frac{2^8(3.875 + 16.375 \times 2^4)}{62} = 2^{10.10} \\ C_{backward} &= \frac{2^4(9.75 + 0.25 \times 2^4 + 16.25 \times 2^8)}{62} = 2^{10.07} \end{aligned} \tag{11}$$

Therefore,

$$C_{match} = C_{forward} + C_{backward} = 2^{11.09} \tag{12}$$

Candidate keys testing. The complexity of this procedure is totally similar to that in Piccolo-80. Since all steps are executed for each key group, the total computational complexity of the attack is:

$$\begin{aligned} C_{total} &= 2^{116} \times (C_{biclique} + C_{match} + C_{recheck}) \\ &= 2^{116} \times (2^{5.05} + 2^{11.09} + 2^4) = 2^{127.12} \end{aligned} \tag{13}$$

5.3 Attack on HIGHT

Here, we obtained $K^{f_0} = k'_4, K^{f_1} = k'_6$, and $K^b = k'_3$ for a 5-round (8,16)-dimensional asymmetric biclique at the plaintext side for rounds 0 to 4. Thus, the intermediate state S refers to the output state of round 4. See Fig. 14 and 15 for biclique and matching part, repectively. In these figures, the associated cutsets are shown with thickened data paths.

Data Copmlexity As we can see in Fig. 14, $P_i = P_0 \oplus \nabla_i^K$. Therefore without any computations, P_i are known all and the data complexity is exactly 2^8 plaintext-ciphertext pairs.

Computational Complexity

Biclique complexity To compute S_j in a asymmetric biclique, 12 F-functions should be calculated once and 8 F-functions should be calculated 2^8 times (see Fig. 14, left). So, the normalized computational complexity of biclique constructing is (a 32-round encryption of HIGHT is taken as the unit of computation which is equivalent to 128 F-function computations):

$$C_{biclique} = \frac{12 + 8 \times 2^8}{128} = 2^{4.01} \tag{14}$$

Matching complexity The right most byte of the input of round 15 is chosen as the matching variable V. In backward direction of partial matching (rounds 31 to 15) for each C_i, 18 F-functions should be calculated once, 16 F-function should be calculated 2^8 times and 18 F-functions should be calculated 2^{16} times (see Fig. 15, right). Also, in forward direction of partial matching (rounds 5 to 14) for each S_j, 13 F-functions should be calculated once and 15 F-functions should be calculated 2^8 times (see Fig. 15, left). Hence, the computational complexity for checking all the keys in a group normalized to a full-round encryption of HIGHT is:

$$\begin{aligned} C_{backward} &= \frac{2^8(18 + 16 \times 2^8 + 18 \times 2^16)}{128} = 2^{21.18} \\ C_{forward} &= \frac{2^{16}(13 + 15 \times 2^8)}{128} = 2^{20.91} \end{aligned} \tag{15}$$

Hence,

$$C_{match} = C_{backward} + C_{forward} = 2^{22.05}. \tag{16}$$

Candidate keys testing. By using of the 8-bit matching variable, the probability of accepting a wrong key is 2^{-8}. Also, we check 2^24 keys in each group. So, the computational complexity of rechecking false keys is:

$$C_{recheck} = 2^{24} \times 2^{-8} = 2^{16} \tag{17}$$

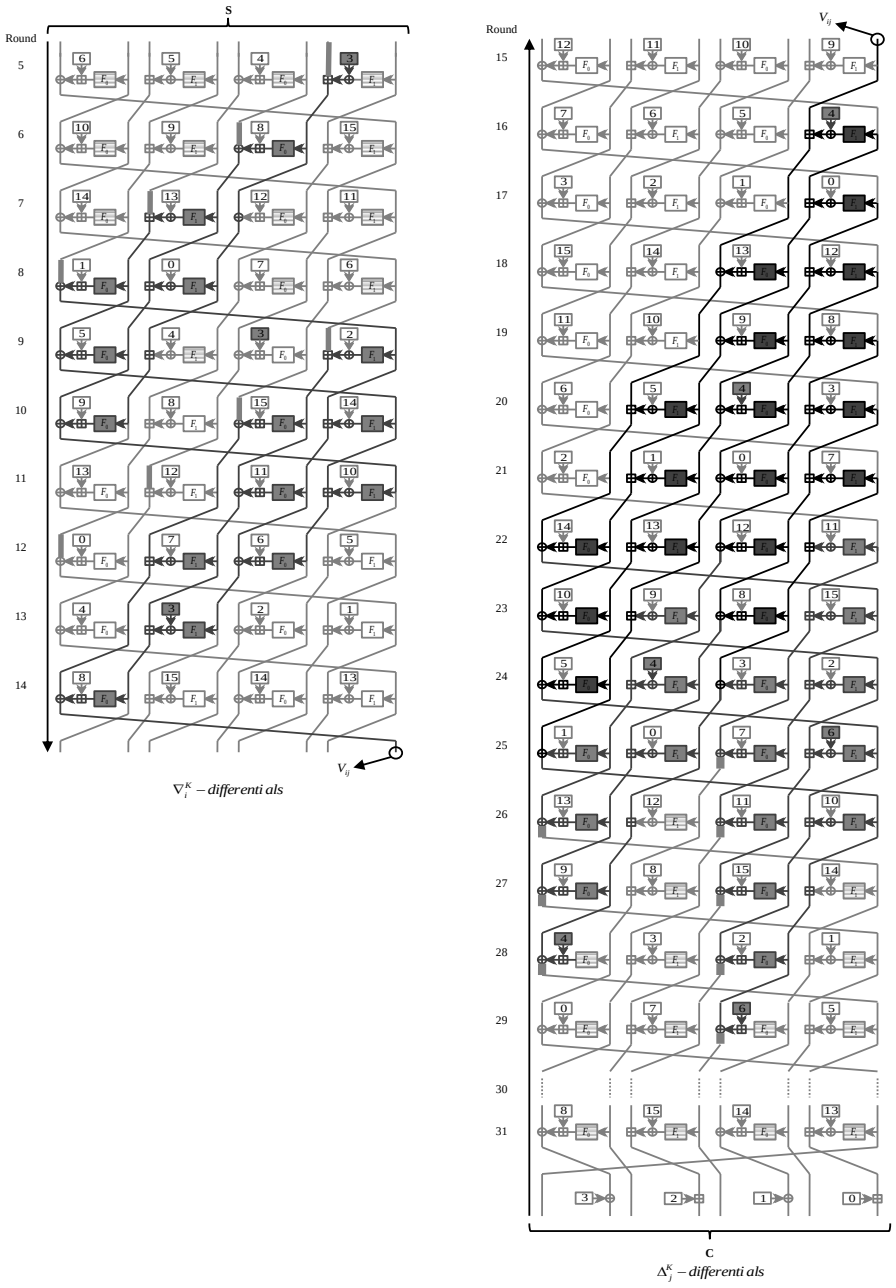

Fig. 15. Partial Matching for HIGHT

Finally, since all steps are executed for each group, the total computational complexity of the attack is:

$$\begin{aligned} C_{total} &= 2^{104} \times (C_{biclique} + C_{match} + C_{recheck}) \\ &= 2^{104}(2^{4.01} + 2^{22.05} + 2^{16}) = 2^{126.07}. \end{aligned} \tag{18}$$

This algorithm is also analyzed in [5] by a biclique attack, in which the computational complexity is computed erroneously where the internal permutation between rounds 11 and 12 was not considered in matching part. We took this into account and calculated the correct computations for this attack as $2^{126.0}$.

6 Conclusions

We presented a variant of biclique attack adapted to cryptanalysis of generalized feistel ciphers with word-wise permutation-like key schedules. What distinguishes our attack is the dramatically low amount of data that it requires. Our attack model fits some realistic scenarios where the data available to the adversary is limited, while its computational budget is not considered to be limited.

We applied this attack on piccolo-80, Piccolo-128 and HIGHT block ciphers. In the two former cases the attack requires only 16 plaintext-ciphertext pairs and in the latter case it requires 256 pairs. The presented method is flexible enough to control the computational complexity as well, and hence, the computational complexity of the presented attacks are not sacrificed for reducing data complexity and even some slight improvements are achieved.

It is worth noticing that these low data complexities are achieved by making use of an asymmetric biclique shorter than that in the most efficient attacks on Piccolo-80, Piccolo-128 and HIGHT. Our results also challenge the convention that the longer biclique necessarily results in a more efficient attack.

References

1. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES, ASIACRYPT 2011, LNCS, vol. 7073, pp. 344-371. Springer, Heidelberg (2011)
2. Khovratovich, D., Leurent, G., Rechberger, C.: Narrow-Bicliques: Cryptanalysis of Full IDEA. In EUROCRYPT 2012, LNCS, pp. 392-410. Springer, Heidelberg (2012).
3. Wang, Y., Wu, W., and Yu, X.: Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher, ISPEC 2012, LNCS 7232, pp. 337-352, Springer, Heidelberg (2012)
4. Jeong, K., Kang, H. C., Lee, C., Sung, J., Hong, S.: Biclique Cryptanalysis of Lightweight Block Ciphers PRESENT, Piccolo and LED. Cryptology ePrint Archive, Report 2012/621, (2012)
5. Song, J., Lee, K., and Lee, H.: Biclique cryptanalysis on lightweight block cipher: HIGHT and Piccolo. International Journal of Computer Mathematics, (2013)
6. Hong, D., Koo, B., and Kwon, D.: Biclique attack on the full HIGHT. Information Security and Cryptology-ICISC2011, LNCS 7259, pp. 365-374, Springer, Berlin, (2012)

7. Wang, Y., Wu, W., Yu, X., Zhang, L.: Security on LBlock against Biclique Cryptanalysis, WISA 2012, LNCS 7690, pp 1-14, Springer, Heidelberg, (2012)
8. Karakoc, F., Demirci, H., Harmanci, A.E.: Biclique cryptanalysis of LBlock and TWINE, Information Processing Letters, Volume 113, Issue 12, pp. 423429, (2013)
9. Ahmadian, Z., Salmasizadeh, M., Aref, M.R.: Biclique Cryptanalysis of the Full-round KLEIN Block Cipher, Cryptology ePrint Archive, Report 2013/097 (2013)
10. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J., A Framework for Automated Biclique Cryptanalysis of Block Ciphers, FSE 2013.
11. Canteaut A., Naya-Plasencia M., Vayssiere B., Sieve-in-the-Middle: Improved MITM Attacks, CRYPTO'13, (2013)
12. Bouillaguet, C., Derbez, P. , Dunkelman, O. , Fouque, P. , Keller, N. , Rijmen, V., Low-Data Complexity Attacks on AES, IEEE Transactions on Information Theory, Volume 58, Issue 11, pp. 7002-7017, (2012)
13. Chang, D., Ghosh, M., Sanadhya, S. K., Biclique cryptanalysis of full round AES with reduced data complexity, IIITD-TR-2013-001 Report, 2013.
14. Bogdanov, A., Kavun, E. B., Paar, C., Rechberger, C., and Yalcin, T., Better than brute-force-optimized hardware architecture for efficient biclique attacks on AES-128, SHARCS 2012.
15. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita T., and Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher, CHES 2011, LNCS 6917, pp. 342-357, Springer, Heidelberg, (2011)
16. Hong, D., Sung, J., Hong, S., Lim , J., Lee , S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., and Chee, S., HIGHT: A new block cipher suitable for low-resource device, Cryptographic Hardware and Embedded Systems-CHES2006, LNCS 4249, pp. 4659, Springer, Berlin, (2006)

Appendix A

Tables 2 and 3 show the key schadule of Piccolo block cipher. (rk_{2i}, rk_{2i+1}) are 16-bit round keys and $(con^{80}_{2i}, con^{80}_{2i+1})$ and $(con^{128}_{2i}, con^{128}_{2i+1})$ are 16-bit round constants for Piccolo-80 and Piccolo-128, respectively.

Table 2. Key schedule of Piccolo-80

Piccolo-80					
Whitening Keys	$wk_0 = k_0^L \|\| k_1^R,\ wk_1 = k_1^l \|\| k_0^R$ $wk_2 = k_4^L \|\| k_3^R,\ wk_3 = k_3^L \|\| k_4^R$				
round i	$rk_{2i} \oplus con_{2i}^{80}$	$rk_{2i+1} \oplus con_{2i+1}^{80}$	round i	$rk_{2i} \oplus con_{2i}^{80}$	$rk_{2i+1} \oplus con_{2i+1}^{80}$
0	k_2	k_3	13	k_4	k_4
1	k_0	k_1	14	k_0	k_1
2	k_2	k_3	15	k_2	k_3
3	k_4	k_4	16	k_0	k_1
4	k_0	k_1	17	k_2	k_3
5	k_2	k_3	18	k_4	k_4
6	k_0	k_1	19	k_0	k_1
7	k_2	k_3	20	k_2	k_3
8	k_4	k_4	21	k_0	k_1
9	k_0	k_1	22	k_2	k_3
10	k_2	k_3	23	k_4	k_4
11	k_0	k_1	24	k_0	k_1
12	k_2	k_3			

Table 3. Key schedule of Piccolo-128

Piccolo-128					
Whitening Keys	$wk_0 = k_0^L \|\| k_1^R,\ wk_1 = k_1^l \|\| k_0^R$ $wk_2 = k_4^L \|\| k_7^R,\ wk_3 = k_7^L \|\| k_4^R$				
round i	$rk_{2i} \oplus con_{2i}^{128}$	$rk_{2i+1} \oplus con_{2i+1}^{128}$	round i	$rk_{2i} \oplus con_{2i}^{128}$	$rk_{2i+1} \oplus con_{2i+1}^{128}$
0	k_2	k_3	16	k_2	k_7
1	k_4	k_5	17	k_4	k_3
2	k_6	k_7	18	k_6	k_5
3	k_2	k_1	19	k_2	k_1
4	k_6	k_7	20	k_6	k_5
5	k_0	k_3	21	k_0	k_7
6	k_4	k_5	22	k_4	k_3
7	k_6	k_1	23	k_6	k_1
8	k_4	k_5	24	k_4	k_3
9	k_2	k_7	25	k_2	k_5
10	k_0	k_3	26	k_0	k_7
11	k_4	k_1	27	k_4	k_1
12	k_0	k_3	28	k_0	k_7
13	k_6	k_5	29	k_6	k_3
14	k_2	k_7	30	k_2	k_5
15	k_0	k_1			

Table 4. Key schedule of HIGHT (the i^{th} byte of the master key K counting from right, is shown with i itself.)

HIGHT			
Whitening Keys	$wk_0 = 12,\ wk_1 = 13,\ wk_2 = 14,\ wk_3 = 15$ $wk_4 = 0,\ wk_5 = 1,\ wk_6 = 2,\ wk_7 = 3$		
Round i	$sk_{4i+j} \boxplus \delta_{4i+j}, j = 0, ..., 3$	Round i	$sk_{4i+j} \boxplus \delta_{4i+j}, j = 0, ..., 3$
0	3,2,1,0	16	7,6,5,4
1	7,6,5,4	17	3,2,1,0
2	11,10,9,8	18	15,14,13,12
3	15,14,13,12	19	11,10,9,8
4	2,1,0,7	20	6,5,4,3
5	6,5,4,3	21	2,1,0,7
6	10,9,8,15	22	14,13,12,11
7	14,13,12,11	23	10,9,8,15
8	1,0,7,6	24	5,4,3,2
9	5,4,3,2	25	1,0,7,6
10	9,8,15,14	26	13,12,11,10
11	13,12,11,10	27	9,8,15,14
12	0,7,6,5	28	4,3,2,1
13	4,3,2,1	29	0,7,6,5
14	8,15,14,13	30	12,11,10,9
15	12,11,10,9	31	8,15,14,13