SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Biclique cryptanalysis using balanced complete bipartite subgraphs

GONG Zheng[1,2*], LIU Shusheng[1], WEN Yamin[3], LUO Yiyuan[4] & QIU Weidong[5]

[1]*School of Computer Science, South China Normal University, Guangzhou* 510631*, China;*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[3]*School of Mathematics and Statistics, Guangdong University of Finance and Economics, Guangzhou* 510320*, China;*
[4]*School of Electronics and Information, Shanghai Dian Ji University, Shanghai* 200240*, China;*
[5]*School of Information Security Engineering, Shanghai Jiao Tong University, Shanghai* 200240*, China*

Appendix A Comparison of related cryptanalysis on mCrypton and Piccolo

Table A1 Summary of related cryptanalyses on mCrypton and Piccolo.

Cipher	Rounds	Time	Memory	Data	Type	Reference
mCrypton-64	full	$2^{62.67}$	2^4	$2^{33.9}$	Biclique	[1]
mCrypton-96	full	$2^{93.93}$	2^{20}	$2^{27.54}$	Biclique	[2]
mCrypton-96	full	$2^{93.75}$	2^4	$2^{63.4}$	Biclique	[2]
mCrypton-128	full	$2^{126.05}$	2^4	–	Biclique	[2]
mCrypton-64	full	$2^{62.9}$	2^4	2^{32}	Biclique	this paper
mCrypton-96	full	$2^{94.6}$	2^4	2^{64}	Biclique	this paper
mCrypton-128	full	$2^{126.57}$	2^4	2^{64}	Biclique	this paper
Piccolo-80 (without postwhitening keys)	full	$2^{78.95}$	2^8	2^{48}	Biclique	[3]
Piccolo-80	full	$2^{79.13}$	2^8	2^{48}	Biclique	[4]
	full	$2^{79.34}$	2^8	2^{48}	Biclique	[5]
	full	$2^{79.7}$	2^{80}	1	MITM	[6]
	full	$2^{79.07}$	2^8	2^4	Biclique	[7]
	full	$2^{79.2}$	2^8	2	Biclique	[7]
	full	$2^{78.9}$	2^8	2^{64}	Biclique	this paper

Appendix B Notions and notation

• Let $\mathcal{E} = f \circ g \circ h$ denote a block cipher as a composition of subciphers f, g and h. For a n-round block cipher $\mathcal{E}$, the r-th round is denoted by $Round_r$ where $r \in \{1, 2, \cdots, n\}$.

• For a binary string x, $|x|$ denotes its bitwise length. For a set V, $|V|$ denotes the number of elements in the set.

• Let $G = (V, U, E)$ denote a bipartite graph G where its vertices can be divided into two disjoint sets V and U with edges in E. $V^{'} \subseteq V$ if a set $V^{'}$ is a subset of a set V. $G^{'} \subseteq G$ if a graph $G^{'}$ is a subgraph of a graph G.

• Let K denote the master key of $\mathcal{E}$. sk_r ($r \in \{1, 2, \cdots, n+1\}$) represents the r-th round subkey of $\mathcal{E}$ such that sk_{n+1} is the output whitening key of $\mathcal{E}$.

• A key K can be represented by $K = k_{\ell-1}k_{\ell-2}\cdots k_1k_0$, where k_i is the i-th operating unit in the key schedule. The set $\mathcal{I} = \{i \mid i \in [0, \ell)\}$ consists of all the indices of K. For example, if $K = k_3k_2k_1k_0$, then $\mathcal{I} = \{0, 1, 2, 3\}$.

• ΔK_i (or ∇K_i) is a truncated related-key difference that is different at the i-th operating unit k_i only. For example, if $K = k_3k_2k_1k_0$, then $\Delta K_2 = 0x00$ (or $\nabla K_2 = 0x00$) indicates that k_2 is active.

* Corresponding author (email: cis.gong@gmail.com)

• ΔK_i (or ∇K_j)-differentials are a truncated differentials that map a zero (or non-zero) input difference to a non-zero (or zero) output difference under the key difference ΔK_i (or ∇K_j). Let $AN_{\Delta K_i}$ (or $AN_{\nabla K_j}$) denote the indices set of active units of $\mathcal{E}$ under Δ_i-differentials (or ∇_j-differentials).

• $\Delta K_I = \bigoplus_{i\in I} \Delta K_i$ are truncated differentials of two master keys that are different at k_i where $i \in I$ and $I \subseteq \mathcal{I}$. For example, if $K = k_3k_2k_1k_0$ and $I = \{1, 3\}$, then $\Delta K_I = x0x0$ (or $\nabla K_I = x0x0$).

• ΔK_I(or ∇K_J)-differentials are truncated differentials that map zero (or non-zero) input differences to non-zero (or zero) output differences under the key difference ΔK_I (or ∇K_J) where $I, J \subseteq \mathcal{I}$. Let $AN_{\Delta K_I}$ (or $AN_{\nabla K_J}$) denote the indices set of active units of $\mathcal{E}$ under ΔK_I-differentials (or ∇K_J-differentials).

Appendix C Biclique cryptanalysis

Biclique cryptanalysis is derived from the splice-and-cut framework [8] in the cryptanalysis of the hash function and then applied to block cipher cryptanalysis [9]. Following the work of [9], the concept of the biclique cryptanalysis is described in below.

Biclique. Let a block cipher $\mathcal{E} = f \circ g \circ h$. The subciphers f, g, h map a plaintext P to the ciphertext C where $f_K(P) = V, g_K(V) = S, h_K(S) = C$. For $I = \{i|\ 0 \leqslant i < 2^d\}$ and $J = \{j|\ 0 \leqslant j < 2^d\}$, let $\{S_j\}$ be two sets of 2^d internal states, $\{C_i\}$ be a set of 2^d ciphertexts and $\{K[i, j]\}$ be a set of 2^{2d} keys. If $C_i = h_{K[i,j]}(S_j)$ for all $i \in I, j \in J$, the tuple $[\{C_i\}, \{S_j\}, \{K[i, j]\}]$ is called a *d-dimensional biclique* for the block cipher $\mathcal{E}$.

Based on the definition of biclique, the biclique cryptanalysis compromises in the following steps.

1. **Preparation** The adversary finds two independent related-key differentials ΔK_I-differentials and ∇K_J-differentials. Because $I = \{i|\ 0 \leqslant i < 2^d\}$ and $J = \{j|\ 0 \leqslant j < 2^d\}$, ΔK_I-differentials and ∇K_J-differentials divide the key space into 2^{n-2d} groups of keys with cardinality 2^{2d}.
2. **Constructing the biclique** For 2^{n-2d} groups of keys, let the key $K[0, 0]$ map the intermediate state S_0 to ciphertext C_0 such that $S_0 \xrightarrow[h]{K[0,0]} C_0$. Selecting every ΔK_i in ΔK_I-differentials maps the input difference zero to the output difference Δ_i, and every ∇K_j in ∇K_J-differentials maps the non-zero input difference ∇_j to the output difference zero. The adversary can construct a d-dimensional biclique $[\{C_i\}, S\{j\}, \{K[i, j]\}]$ from $S_j \xrightarrow[h]{K[i,j]} C_i$ such that $i \in I, j \in J$ as follows.
$$\begin{aligned} S_j &= S_0 \oplus \nabla_j \\ C_i &= C_0 \oplus \Delta_i \\ K[i, j] &= K[0, 0] \oplus \Delta K_i \oplus \nabla K_j \end{aligned}$$
3. **Decryption step** The adversary asks the oracle to decrypt ciphertexts C_i with the master key K and obtains 2^d plaintexts P_i, $C_i \xrightarrow{decryption\ oracle} P_i$ where $i \in I$.
4. **Matching with precomputations** Matching with precomputations, which was proposed by Bogdanov et al. [9], can speed up biclique cryptanalysis for key recovery attacks. For each key candidate, the adversary checks whether the following equation holds:
$$\forall i \in I, \forall j \in J, P_i \xrightarrow[f]{K[i,0]} \overrightarrow{v} \stackrel{?}{=} \overleftarrow{v} \xleftarrow[g^{-1}]{K[0,j]} S_j$$
The adversary precomputes and stores in memory 2×2^d full computations of f and g up to the matching variables $\overrightarrow{v}$ and $\overleftarrow{v}$. To reduce the time complexity, the adversary only recomputes the part of intermediate values that are partially changed from the saved values for each (P_i, S_j).

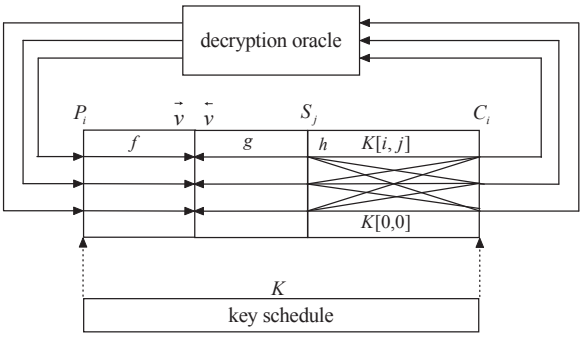

Figure C1 Illustration of biclique cryptanalysis

Appendix D Proof of Theorem 1

Theorem 1. Let a subcipher h of $\mathcal{E} = f \circ g \circ h$ satisfy the (**strong**) independent differential property, and the balanced bipartite graph $G = (V, U, E)$ is constructed from h. If (**and only if**) there exists a balanced complete bipartite subgraph $G^{'} = (V^{'}, U^{'}, E^{'}) \subseteq G$, such that $I = \{i \mid v_i \in V^{'}, 0 \leqslant i < |V^{'}|\}$, $J = \{j \mid u_j \in U^{'}, 0 \leqslant j < |U^{'}|\}$ and $E^{'} = \{(v_i, u_j) \mid v_i \in V^{'}, u_j \in U^{'}, i \in I, j \in J\}$, then ΔK_I-differential and ∇K_J-differential are independent.

Proof. According to the process of constructing G, a balanced complete bipartite subgraph $G^{'} \subseteq G$ implies that for any $i \in I$ and $j \in J$, (ΔK_i-differential,∇K_j-differential)$\in R$. If h satisfies the independent differential property, ΔK_I-differentials and ∇K_J-differentials are independent.

Moreover, if h has the strong independent property, $G^{'} = (V^{'}, U^{'}, E^{'})$ can be transformed into two independent ΔK_I-differential and ∇K_J-differential. Consequently, ΔK_I-differential and ∇K_J-differential can be transformed into the $(|V^{'}| \times |k_i|)$-dimensional biclique. Assume there exists a $((|V^{'}| + 1) \times |k_i|)$-dimensional biclique of h, then we determine find $\Delta K_{I^{''}}$-differential and $\nabla K_{J^{''}}$-differential that are independent, in which $\Delta K_{I^{''}}$-differential and $\nabla K_{J^{''}}$-differential can be transformed into the $((|V^{'}| + 1) \times |k_i|)$-dimensional biclique. The size of $I^{''}$ (and $J^{''}$) is $(|V^{'}| + 1)$. According to Theorem 1, there exists a balanced complete bipartite subgraph $G^{''} = (V^{''}, U^{''}, E^{''})$ such that $|V^{''}| = |V^{'}| + 1$. Thus $G^{'}$ is not a maximum balanced complete bipartite subgraph of G, which is conflicted with the precondition that $G^{'}$ is a maximum balanced complete bipartite subgraph of G. Therefore, one from this that the $(|V^{'}| \times |k_i|)$-dimensional biclique, which is transformed from $G^{'}$, is the maximum dimensional biclique of h.

Appendix E Algorithms for searching independent related-key differentials

The main steps of Algorithm E1 are explained as follows:

Step 1 $G = (V, U, E)$ is divided into several connected components G_i^c of the graph G.

Step 2 For each connected components G_i^c, we remove the vertices of G_i^c whose degree are less than D, and repeat this step until no vertex is removed. After this step, we obtain the revised connected component $G_i^{rc} = \{CV, CU, CE\}$ where the degree of vertices in the vertex sets CV and CU are no less than D.

Step 3 For each G_i^{rc}, we determine all $(2D)$-vertex balanced complete bipartite subgraph of G_i^{rc}.

Step 4 The set of $2D$-vertex balanced complete bipartite subgraphs of the G consists of $2D$-vertex balanced complete bipartite subgraphs of G_i^{rc}.

Complexity of Algorithm E1 and Algorithm E2. The complexity of Algorithm E2 is approximately equal to Algorithm E1. Thus, we only need to accurately evaluate the complexity of Algorithm E1. Let $|G_D|$ be the number of $2D$-vertex balanced complete bipartite subgraph. Checking whether a subgraph is a balanced complete bipartite subgraph as the basic operations of time complexity, where the time complexity $C_{time} = \sum\limits_{0 \leqslant i < D} |E| \times |G_i|$ and memory complexity $C_{memory} \leqslant 2 \times \max\limits_{0 \leqslant i \leqslant D} |G_i|$.

Appendix F Biclique cryptanalysis of full-round mCrypton

In this section, first we describe that mCrypton has the independent differential property. Next, we use Algorithm E2 to determine independent related-key differentials of 4-round mCrypton-96, which is transformed into 4-dimensional bicliques. Furthermore, we use independent biclique approaches to perform matching with precomputations. Finally, we present the complexities of our key recovery attack on full-round mCrypton-96. Note that similar results can also be provided for mCrypton-64/128.

Appendix F.1 Brief description of mCrypton

mCrypton is a lightweight block cipher with 64-bit block size and 64/96/128-bit key, which are named mCrypton-64/96/128, respectively [10]. The internal state and round key are represented by a 4×4 nibble matrix. The round function consists of four steps: nibble-wise substitution γ, column-wise bit permutation π, column-to-row transposition τ and the key addition σ. More specifically, the round function for encryption is defined by $\rho_{sk_i} = \sigma_{sk_i} \circ \tau \circ \pi \circ \gamma$, where σ_{sk_i} is key addition with round key sk_i. The encryption E_K of mCrypton under master key K consists of an initial key addition and 12 times repetitions of the same round function and then a finale output function ϕ.

Key schedule for mCrypton-96. Because we will present the biclique cryptanalysis on mCrypton-96 as the example, in this paper we recall the key schedule for mCrypton-96 for a better understanding of the attack. Let $K = (K[0], K[1], K[2], K[3], K[4], K[5])$ be the secret key for mCrypton-96, where $K[i]$ represents the i-th 16-bit word in K. Let $C[r]$ be the constant of round r. The key register U is first initialized with the secret key K and subkeys are computed as follows: For round $r = 0, 1, \cdots, 12$,

$$
\begin{gathered}
T \leftarrow S(U[0]) \oplus C[r] \\
T_i \leftarrow T \odot M_j, j = \{0, 1, 2, 3\}, M_0 = 0xf000, M_1 = 0x0f00, M_2 = 0x00f0, M_3 = 0x000f \\
sk_r \leftarrow (U[1] \oplus T_0, U[2] \oplus T_1, U[3] \oplus T_2, U[3] \oplus T_3) \\
U \leftarrow (U[5], U[0] \lll 3, U[1], U[2], U[3] \lll 8, U[4]).
\end{gathered}
$$

Algorithm E1 Searching $2D$-vertex balanced complete bipartite subgraphs of graph G.

Require: $G = (V, U, E)$: the balanced bipartite graph G is constructed from a subcipher h of the cipher $\mathcal{E} = f \circ g \circ h$.
D: an integer chosen for D-dimensional biclique.;
Ensure: $BCBS[D]$: the set of $2D$-vertex balanced complete bipartite subgraphs of the graph G.;
$G = (V, U, E)$ is divided into several connected components G_i^c of the graph G.
for each connected components G_i^c **do**
  Remove the vertices of G_i^c whose degree are less than D, and repeat this step until no vertex is removed. Obtain revised connected component $G_i^{rc} = \{CV, CU, CE\}$ where the degree of the vertices in the vertex sets CV and CU is no less than D;
end for
for each $G_i^{rc} = \{CV, CU, CE\}$ **do**
  empty TBCBS; $k \leftarrow 0$;
  for each edge (v_x, u_y) in CE **do**
    storage 2-vertex balanced complete bipartite subgraphs $G_1 = (V^{'} = \{v_x\}, U^{'} = \{u_y\}, E' = \{(v_x, u_y)\})$ into $TBCBS[1][k]$; $k \leftarrow k + 1$;
  end for
  for $d \leftarrow 1$; $d \leqslant D$; $d \leftarrow d + 1$ **do**
    $k \leftarrow 0$;
    for each $2d$-vertex balanced complete bipartite subgraph $G_d = (V^{'}, U^{'}, E^{'})$ in $TBCBS[d]$ **do**
      for each edge (v_x, u_y) in CE **do**
        $V^{''} = V^{'} \cup \{v_x\}$; $U^{''} = U^{'} \cup \{u_y\}$; $E^{''} = E^{'} \cup \{(v_i, u_y) \mid \forall v_i \in V^{'} and\ (v_i, u_y) \in CE\} \cup \{(v_x, u_j) \mid \forall u_j \in U^{'} and\ (v_x, u_j) \in CE\}$;
        if $G_{d+1} = (V^{''}, U^{''}, E^{''})$ is a $2(d+1)$-vertex balanced complete bipartite subgraph and G_{d+1} is not in $TBCBS[d+1]$ **then**
          storage G_{d+1} into $TBCBS[d+1][k]$; $k \leftarrow k + 1$;
        end if
      end for
    end for
  end for
  $BCBS[D] \leftarrow BCBS[D] \cup TBCBS[D]$;
end for

Algorithm E2 Constructing independent related-key truncated differentials from BCBS

Require: a subcipher h of a cipher $\mathcal{E} = f \circ g \circ h$.
Ensure: pairs of independent related-key truncated differentials of h .
By following the steps described in Appendix E, construct the balanced bipartite graph $G = (V, U, E)$ from h; Calculate the set of maximum balanced complete bipartite subgraphs $\{G^{'} = (V^{'}, U^{'}, E^{'})\} = Algorithm E1(G)$;
for each maximum balanced complete bipartite subgraph $G^{'} = (V^{'}, U^{'}, E^{'})$ **do**
  transform $G^{'}$ into two index sets I and J, which are calculated as follows.

$$
\begin{aligned}
I &= \{i \mid v_i \in V^{'}\ and\ 0 \leqslant i < |V^{'}|\} \\
J &= \{j \mid u_j \in U^{'}\ and\ 0 \leqslant j < |U^{'}|\} \\
E^{'} &= \{(v_i, u_j) \mid v_i \in V^{'}, u_j \in U^{'}, i \in I, j \in J\}
\end{aligned}
$$

  (ΔK_I-differentials, ∇K_J-differentials) is a pair of independent related-key truncated differentials.
end for

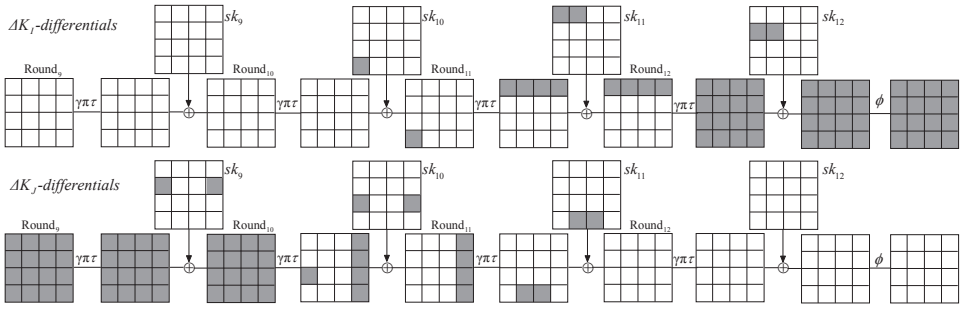

Figure F1 The independent ΔK_I-differential and ∇K_J-differential of 4-round mCrypton-96.

Appendix F.2 The independent differential property of mCrypton-96

Because the key schedule of mCrypton-96 is bit-oriented, the operating unit of key schedule is bit. Let the key register in the key schedule be denoted by $K = k_{95} \cdots k_1 k_0$. The set $\mathcal{I} = \{i | 0 \leqslant i < 96\}$. The round function of mCrypton-96 is nibble-oriented and the size of the nonlinear component is 4-bit, therefore the active nonlinear component set consists of active 4-bit s-boxes. We found that mCrypton-96 has the following properties.

- for any $I \subseteq \mathcal{I}$, the ΔK_I-differentials of mCrypton satisfies $AN_{\Delta K_I} \subseteq \bigcup_{i \in I} AN_{\Delta K_i}$
- for any $J \subseteq \mathcal{I}$, the ∇K_J-differentials of mCrypton satisfies $AN_{\nabla K_J} \subseteq \bigcup_{j \in J} AN_{\nabla K_j}$

Therefore, if $\forall\ i \in I$ and $j \in J$, $AN_{\Delta K_i} \cap AN_{\nabla K_j} = \emptyset$, then $AN_{\Delta K_I} \cap AN_{\nabla K_J} = \emptyset$. To summarize, if $\forall\ i \in I$ and $j \in J$, ΔK_i-differentials and ∇K_j-differentials are independent, then ΔK_I-differentials and ∇K_J-differentials are independent. Therefore mCrypton satisfies the independent differential property.

Appendix F.3 4-Round biclique of dimension 4 of mCrypton-96

Because mCrypton satisfies the independent differential property, we can use the Algorithm E2 to determine the independent related-key differentials (ΔK_I-differentials, ∇K_J-differentials) of subcipher h of mCrypton-96, where h is defined by $h = \phi \circ \rho_{sk_{12}} \circ \rho_{sk_{11}} \circ \rho_{sk_{10}} \circ \rho_{sk_9}$. For simplicity, we define the key groups with respect to the key register of sk_9. We obtain a 4-dimensional biclique from two related-key differentials (ΔK_I-differentials, ∇K_J-differentials), which are obtained by Algorithm E2 and depicted in Figure F1. The sets $I = \{64, 65, 66, 67\}$ and $J = \{16, 17, 30, 31\}$. ΔK_I-differentials and ∇K_J-differentials are based on the related-key differences ΔK_I and ∇K_J respectively.

Key partitioning. According to the Δ_I-differential and ∇_J-differential, the base keys $K[0,0]$ are all possible values (2^{88}) with the bits k_i ($i \in \{I \cup J\}$) fixed to 0 while the remaining 88 bits run over all values. The keys $\{K[i,j]\}$ in a group are enumerated by all possible bit differences k_i ($i \in \{I \cup J\}$) with respect to the base $K[0,0]$. This yields the partition of the key space into the 2^{88} groups of 2^8 keys each.

Appendix F.4 Matching over 8 rounds: independent biclique approach

For performing matching with precomputations, mCrypton is decomposed as $E : P \underset{f}{\rightarrow} V \underset{g}{\rightarrow} S \underset{h}{\rightarrow} C$. Now we check whether the master key K belongs to the key group $\{K[i,j]\}$.

Step 1 We make 2^{4+1} precomputation of v and store values, in addition to the intermediate states and subkeys in memory.

$$\text{For } 0 \leqslant i < 2^4\ P_i \xrightarrow[f]{K[i,0]} \overrightarrow{v} \text{ and for } 0 \leqslant j < 2^4\ \overleftarrow{v} \xleftarrow[g]{K[0,j]} S_j.$$

Step 2 We determine the right key by computing an intermediate variable v in both directions. We check equation F1 for every i, j by recomputing only those variables that differ from those stored in memory.

$$P_i \xrightarrow[f]{K[i,j]} \overrightarrow{v} = \overleftarrow{v} \xleftarrow[g]{K[i,j]} S_j \tag{F1}$$

Now we evaluate the quantity of recomputation in the backward and forward directions. Because s-boxes are the main part of the time complexity of mCrypton, we regard the number of s-boxes as the complexity of recomputation in the key recovery attack.

Backward recomputation. The difference between the computation $\overleftarrow{v} \xleftarrow[g]{K[i,j]} S_j$ and the stored version $\overleftarrow{v} \xleftarrow[g]{K[0,j]} S_j$ is shown in the backward recomputation of Figure F2. The black blocks in sk_i are differences of subkeys, and the black blocks in the round state are recomputed nibbles. The recomputation is determined by the influence of the difference between keys $K[i,j]$ and $K[0,j]$, which can be seen as the key difference ΔK_I. As the backward recomputation of Figure F2 shown, we choose the black nibble of $Round_5$ as the matching variable v. Thus, to recompute v under the key difference ΔK_I, we need to recompute the black blocks of $Round_i$ ($5 \leqslant i \leqslant 8$). Moreover, only two s-boxes in the key schedule are required for

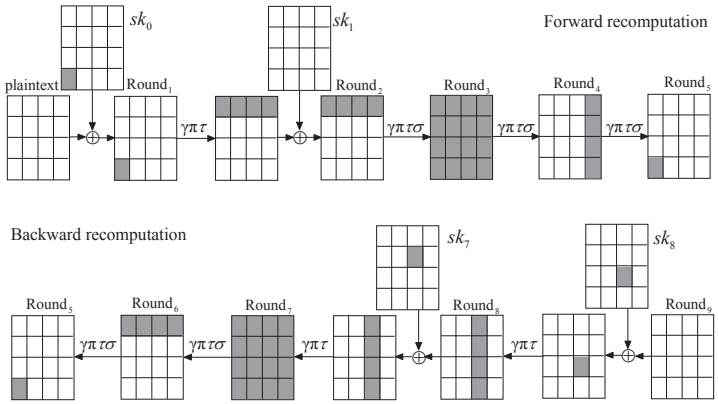

Figure F2 Recomputation in the backward and forward direction of mCrypton-96.

recomputation. Thus the backward recomputation includes 25+2 s-boxes.

Forward recomputation. The difference between the computation $P_i \xrightarrow[f]{K[i,j]} \overrightarrow{v}$ and the stored version $P_i \xrightarrow[f]{K[i,0]} \overrightarrow{v}$ is shown in the forward recomputation of Figure F2. Similarly, the recomputation is determined by the influence of the difference between keys $K[i,j]$ and $K[i,0]$, which can be seen as the difference ∇K_J. Only three s-boxes in the key schedule are required for recomputation. As the forward recomputation of Figure F2 shows, the forward recomputation includes 25+3 s-boxes.

Appendix F.5 Complexities

Because s-boxes are the main part to of the time complexity of biclique attacks, it is widely-accepted that we can count the number of recomputed s-boxes and compare the result to that in the full cipher to estimate the attack complexity. The full complexity of the key recovery attack can be evaluated as follows.

$$C_{full} = 2^{|K|-2d}[C_{biclique} + C_{precomp} + C_{recomp} + C_{falsepos}]$$

- Where $C_{biclique}$ is the complexity of constructing a single biclique. Constructing 4-dimensional biclique of 4-round mCrypton, we need to run f 2^{4+1} times. Thus, The $C_{biclique}$ is equivalent to $2^5 \times \frac{4}{12} = \frac{2^5}{3}$ times of the full mCrypton-96.
- Where $C_{precomp}$ is the complexity of the precomputation. Performing precomputation, we have to run 2^4 times of ε_1 and ε_2. Therefore, $C_{precomp}$ is equivalent to $2^4 \times \frac{8}{12} = \frac{2^5}{3}$ runs of the full mCrypton-96.
- Where C_{recomp} is the complexity of the recomputation of the interval variable v 2^{2d} times. As shown in Figure F2, we have to recompute 25 s-boxes in each forward or backward recomputation. Taking the key schedule into account, 3 and 2 s-boxes are recomputed in each forward and backward recomputation, respectively. As a result, C_{recomp} is equivalent to $2^8 \times (\frac{25+2+25+3}{12\times(16+4)}) = \frac{176}{3}$ runs of the full mCrypton-96.
- Where $C_{falsepos}$ is the complexity generated by false positives, which have to be matched on other variables. If we match on a single nibble, the number of false positives is approximately 2^{2d-4}. Each one requires only a few operations to recheck.

The full computational complexity is approximately

$$C_{full} = 2^{88}(\tfrac{2^5}{3} + \tfrac{2^5}{3} + \tfrac{176}{3} + 2^4) < 2^{94.6}.$$

Because we have to store 2^4 precomputations of ε_1 and ε_2, the memory requirement is upper-bounded by the storage of 2^4 full computations of mCrypton-96. Because the Δ_I-differentials affects 16 nibbles of the ciphertext, the data complexity does not exceed 2^{64}. Based on the aforementioned method, we also determine new biclique constructions for mCrypton-64/128, which are depicted in Figures F3 and F4. The attack complexities can be easily calculated by using the similar analysis of mCrypton-96 thus we omit the details here. Compare our biclique cryptanalyses of mCrypton to the related works [1, 2], although the time complexities of our attacks are slightly higher than the previous results, we note that the dimension of the constructed bicliques are the same. Moreover, the independent related-key differentials that are used to build the bicliques of mCrypton-64/96/128 are different from [1, 2], which means Algorithm E1 and Algorithm E2 are also practical for automatically searching independent related-key differentials, especially to construct high-dimensional bicliques.

Appendix G Biclique cryptanalysis of full-round Piccolo-80

In this section, first we describe Piccolo-80 satisfy the strong independent differential property (Definition 3 in Section 3.1). Then we apply Algorithm E2 for biclique cryptanalysis of Piccolo-80 and propose an attack on full round Piccolo-80, including all whitening keys, which takes advantage of 7-round bicliques of dimension 8.

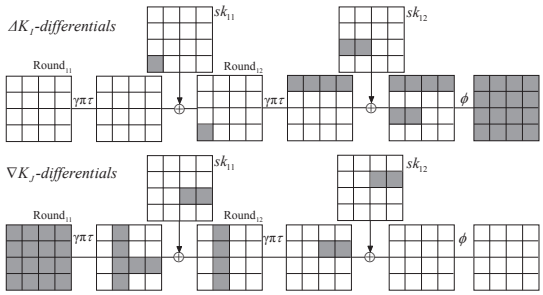

Figure F3 Recomputation in the backward and forward direction of mCrypton-64.

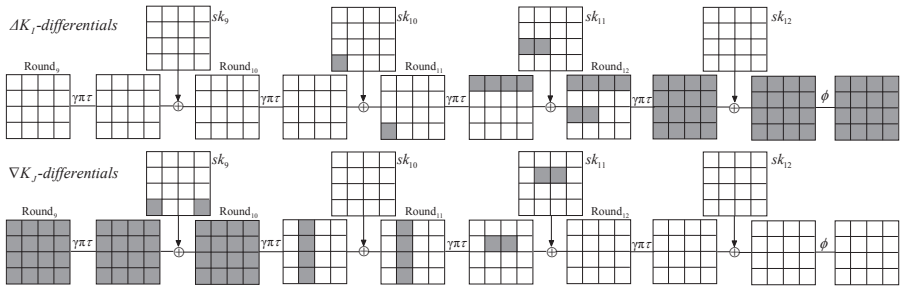

Figure F4 Recomputation in the backward and forward direction of mCrypton-128.

Appendix G.1 Brief description of Piccolo-80

Piccolo is a lightweight block cipher with 80/128-bit key length, which was proposed at CHES 2011 [11]. Piccolo-80 has a 64-bit block size and 80-bit key length. The number of iterative rounds is 25. The iterative structure and components of Piccolo-80 can be found in [11]. The round structure consists of two functions F and one round permutation RP. Furthermore, the function F consists of two s-box layers separated by a diffusion matrix M. The s-box layer consists of the four same 4-bit bijective s-boxes S. The round permutation RP divides a 64-bit input into 8 bytes, and then permutes the positions of the 8 bytes.

Key schedule for Piccolo-80. The key scheduling function divides an 80-bit key K into 5×16-bit k_i $(0 \leqslant i < 5)$: $K = k_4k_3k_2k_1k_0$. It provides 4×16-bit whitening keys wk_i $(0 \leqslant i < 4)$ and 50×16-bit round key sk_i $(0 \leqslant i < 50)$ as follows:

$wk_0 \leftarrow k_0^L \| k_1^R$, $wk_1 \leftarrow k_1^L \| k_0^R$, $wk_2 \leftarrow k_4^L \| k_3^R$, $wk_3 \leftarrow k_3^L \| k_4^R$
for $i \leftarrow 0$ to 24 do
  if $i \bmod 5 = 0$ or 2, then $(rk_{2i}, rk_{2i+1}) \leftarrow (con_{2i}^{80}, con_{2i+1}^{80}) \oplus (k_2, k_3)$
  if $i \bmod 5 = 1$ or 4, then $(rk_{2i}, rk_{2i+1}) \leftarrow (con_{2i}^{80}, con_{2i+1}^{80}) \oplus (k_0, k_1)$
  if $i \bmod 5 = 3$, then $(rk_{2i}, rk_{2i+1}) \leftarrow (con_{2i}^{80}, con_{2i+1}^{80}) \oplus (k_4, k_4)$

where k_i^L and k_i^L are left and right half 8 bits of k_i.

Appendix G.2 The strong independent differential property of Piccolo-80

The size of the nonlinear component of Piccolo-80 is 4-bit, and the round structure of Piccolo-80 is nibble-oriented and the key schedule of Piccolo-80 has no nonlinear component. Thus, the operating unit of the key schedule is 4-bit. Let the master key $K = k_{19} \cdots k_1k_0$, in which k_i is the i-th nibble of master key. The set $\mathcal{I}$ is $\{i \mid 0 \leqslant i < 20\}$. We find that Piccolo-80 has the following two properties.

- For any $I \subseteq \mathcal{I}$, the ΔK_I-differentials of Piccolo satisfies $AN_{\Delta K_I} = \bigcup_{i \in I} AN_{\Delta K_i}$
- For any $J \subseteq \mathcal{I}$, the ∇K_J-differentials of Piccolo satisfies $AN_{\nabla K_J} = \bigcup_{j \in J} AN_{\nabla K_j}$

Therefore, if $AN_{\Delta K_I} \cap AN_{\nabla K_J} = \emptyset$, then $\forall\, i \in I$ and $j \in J$, $AN_{\Delta K_i} \cap AN_{\nabla K_j} = \emptyset$, and vice versa. To summarize, $\forall\, i \in I$ and $j \in J$, ΔK_i-differentials and ∇K_j-differentials are independent, if and only if ΔK_I-differentials and ∇K_J-differentials are independent. Now we can conclude that Piccolo-80 satisfies the strong independent differential property.

Appendix G.3 7-Round biclique of dimension 8 of Piccolo-80

Because Piccolo-80 satisfies the strong independent differential property, we can use the Algorithm E2 to determine independent related-key differentials, which can be transformed into a maximum dimensional biclique.

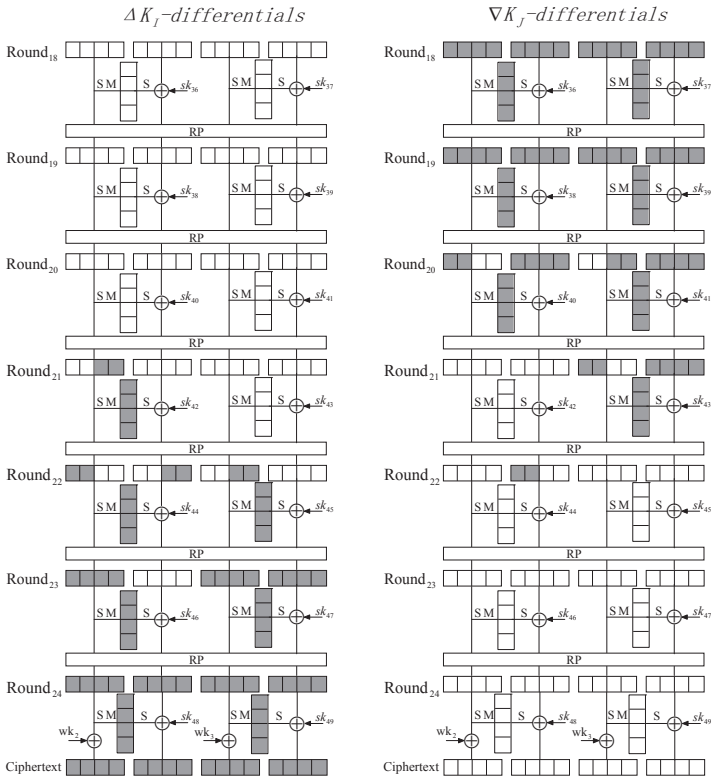

Figure G1 The independent ΔK_I-differential and ∇K_J-differential of last 7-round Piccolo-80.

Let f be the last 7 rounds of Piccolo-80. We obtain a 8-dimension biclique from two related-key differentials (ΔK_I-differentials, ∇K_J-differentials), which are obtained by Algorithm E2 and depicted in Figure G1. The set $I = \{12, 13\}$ and $J = \{10, 11\}$. ΔK_I-differentials and ∇_J-differentials are based on the related-key key differences ΔK_I and ∇K_J respectively.

Key partitioning. According to the ΔK_I-differential and ∇K_J-differential, the base keys $K[0, 0]$ are all possible 2^{64} values with the 4-nibble $\{k_{10}, k_{11}, k_{12}, k_{13}\}$ fixed to 0 whereas the remaining 64 bits run over all values. The keys $\{K[i, j]\}$ in a group are enumerated by all possible differences at $\{k_{10}, k_{11}, k_{12}, k_{13}\}$ with respect to the base $K[0, 0]$. This yields the partition of the K space into the 2^{64} groups of 2^{16} keys each.

Appendix G.4 Matching over 18 rounds: independent biclique approach

Performing matching with precomputations, Piccolo-80 is decomposed as $E : P \underset{f}{\rightarrow} V \underset{g}{\rightarrow} S \underset{h}{\rightarrow} C$. Now we evaluate the complexity of recomputation in backward and forward direction. Because s-boxes are the main part of the time complexity of Piccolo-80, we regard the number of s-boxes as the complexity of recomputation in the key recovery attack. Taking the computational complexity into account, we choose the 0-th and 1-th nibble of $Round_9$ as the matching variable v.

Backward recomputation. The difference between the computation $\overleftarrow{v} \xleftarrow[g]{K[i,j]} S_j$ and the stored version $\overleftarrow{v} \xleftarrow[g]{K[0,j]} S_j$ is shown in the backward recomputation of Figure G2. The black blocks in $Round_i$ are recomputed nibbles. The recomputation is determined by the influence of the difference between keys $K[i, j]$ and $K[0, j]$, which can be seen as the key difference ΔK_I. Thus, to recompute v under the key difference ΔK_I, we need to recompute the black blocks of $Round_i$ ($9 \leqslant i \leqslant 18$). The backward recomputation includes 83 s-boxes.

Forward recomputation. The difference between the computation $P_i \xrightarrow[f]{K[i,j]} \overrightarrow{v}$ and the stored version $P_i \xrightarrow[f]{K[i,0]} \overrightarrow{v}$ is shown in the forward recomputation of Figure G2. Similarly, the recomputation is determined by the influence of the difference between keys $K[i, j]$ and $K[i, 0]$, which can be seen as the difference ∇K_J. As shown in the forward recomputation of Figure G2 shows, the forward recomputation includes 99 s-boxes.

Appendix G.5 Complexities

As shown in Figure G2, we have to recompute $(83 + 99)$ s-boxes each time. As a result, C_{recomp} is equivalent to $2^{16} \times (\frac{83+99}{25\times16}) = 2^{14.864}$ times of the full-round Piccolo-80. Constructing 8-dimensional biclique of 7-round Piccolo-80, we need to run 2^{8+1} times f. Thus, The $C_{biclique}$ is equivalent to $2^9 \times \frac{7}{25} = 2^{7.164}$ runs of the full Piccolo-80. Performing

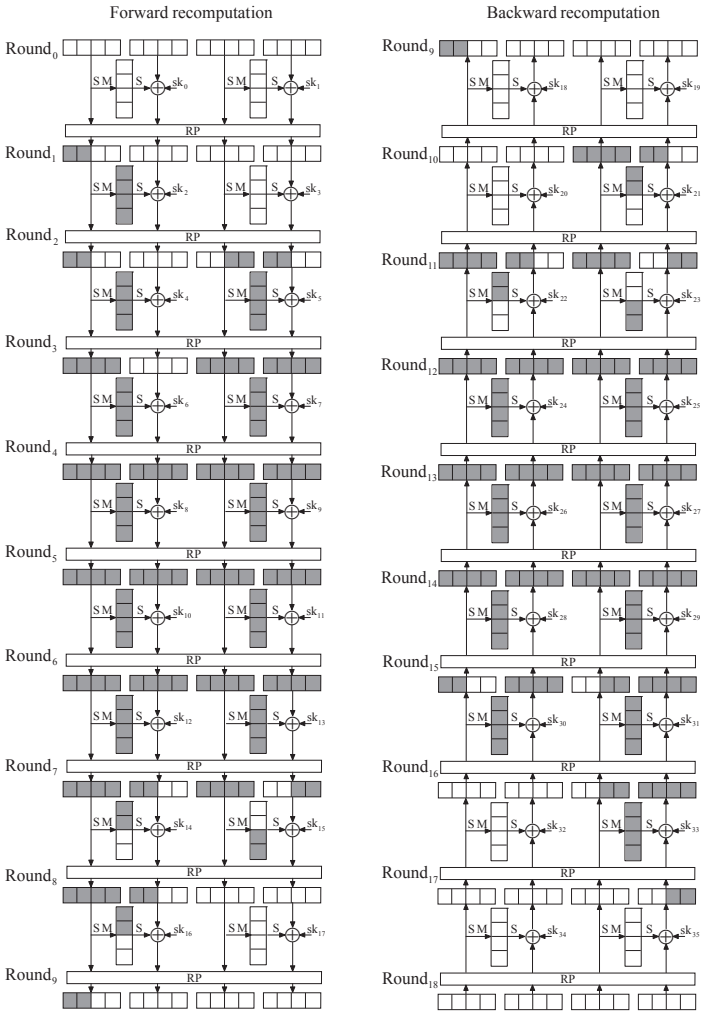

Figure G2 Recomputation in the backward and forward direction of Piccolo-80.

precomputation, we have to run 2^8 times of f and g. Therefore, The $C_{precomp}$ is equivalent to $2^8 \times \frac{18}{25} = 2^{7.256}$ runs of the full Piccolo-80. The full computational complexity is approximately

$$C_{full} = 2^{64}(2^{7.164} + 2^{14.864} + 2^{7.256} + 2^8) < 2^{78.9}.$$

Because we have to store 2^8 precomputations of the subciphers f and g, the memory requirement is upper-bounded by the storage of 2^8 full computations of Piccolo-80. Because the Δ_I-differentials affect 16 nibbles of the ciphertext, the data complexity does not exceed 2^{64}. Similar to our biclique cryptanalyses on mCrypton, our new biclique attack on Piccolo-80 uses a different independent related-key differential to construct the 8-dimensional biclique. The construction of the new biclique supports the proposal that our definition of the strong independent differential property and the searching algorithms are useful for determining maximal dimension bicliques (i.e., independent related-key differentials with maximal active units). We also attempt execute the similar analysis on Piccolo-128, but our experiment to determine the maximum BCBS is failed because Piccolo-128 has more rounds and computing units in its key schedule. Thus whether the proposed biclique cryptanalyses on Piccolo-128 [7] remains an open problem.

References

1 Shakiba M, Dakhilalian M, and Mala H. Cryptanalysis of mCrypton-64. *International Journal of Communication Systems* (2014), Volume 28 Issue 8, 1401–1418.

2 Shakiba, M., Dakhilalian, M., and Mala, H. Non-isomorphic biclique cryptanalysis and its application to full-round mCrypton. Cryptology ePrint Archive, Report 2013/141, 2013. `http://eprint.iacr.org/`.

3 Wang Y, Wu W, and Yu X. Biclique cryptanalysis of reduced-round Piccolo block cipher. In *ISPEC 2012*, Ryan M, Smyth B, and Wang G, Eds., LNCS 7232, Hangzhou, Berlin: Springer, 2012, pp. 337–352.

4 Jeong K, Kang H, Lee C, Sung J, and Hong S. Biclique cryptanalysis of lightweight block ciphers PRESENT, Piccolo and LED. Cryptology ePrint Archive, Report 2012/621, 2012. `http://eprint.iacr.org/`.

5 Song J, Lee K, and Lee H. Biclique cryptanalysis on lightweight block cipher: Hight and Piccolo. *International Journal of Computer Mathematics*, 2013, Volume 90 Issue 12, 2564–2580.

6 Huang J, and Lai X. What is the effective key length for a block cipher: an attack on every practical block cipher. *Science China Information Sciences*, 2014, 57:(7), 1–11.

7 Ahmadi S, Ahmadian Z, Mohajeri J, and Aref M. Low-data complexity biclique cryptanalysis of block ciphers with application to Piccolo and Hight. *IEEE Transaction on Information Forensics and Security*, 2014, 9:(10), 1641–1652.

8 Aoki K, and Sasaki Y. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *CRYPTO 2009*, Haveli S, Ed., LNCS 5677, Santa Barbara, USA, Berlin: Springer, 2009, pp. 70–89.

9 Bogdanov A, Khovratovich D, and Rechberger C. In *ASIACRYPT 2011*, Lee D and Wang X, Eds., LNCS 7073, Seoul, Berlin: Springer, 2011, pp. 344–371.

10 Lim C H, and Korkishko T. mCrypton - A lightweight block cipher for security of low-cost RFID tags and sensors. In *WISA 2005*, Song J, Kwon T, and Yung M, Eds., LNCS 3786, Jeju Island, Korea, Berlin: Springer, 2005, pp. 243–258.

11 Shibutani K, Isobe T, Hiwatari H, Mitsuda A, Akishita T, and Shirai T. Piccolo: An ultra-lightweight blockcipher. In *CHES 2011*, Preneel B and Takagi T, Eds., LNCS 6917, Nara, Japan, Berlin: Springer, 2011, pp. 342–357.