Biclique Cryptanalysis Of PRESENT, LED, And KLEIN
Revision 2013-05-20

Farzaneh Abed, Christian Forler, Eik List, Stefan Lucks, Jakob Wenzel

Bauhaus-Universität Weimar, Germany
`{firstname.lastname}@uni-weimar.de`

Abstract. In this paper, we analyze the resistance of the lightweight ciphers PRESENT, LED, and KLEIN to biclique attacks. Primarily, we describe attacks on the full-round versions PRESENT-80, PRESENT-128, LED-64, LED-128, KLEIN-80, and KLEIN-96. Our attacks have time complexities of $2^{79.49}$, $2^{127.32}$, $2^{63.58}$, $2^{127.42}$, $2^{79.00}$, and $2^{95.18}$ encryptions, respectively. In addition, we consider attacks on round-reduced versions of PRESENT and LED, to show the security margin for which an adversary can obtain an advantage of at least a factor of two compared to exhaustive search.

Keywords: PRESENT, LED, KLEIN, lightweight block cipher, independent biclique, matching-with-precomputations

1 Introduction

Biclique cryptanalysis. *Biclique cryptanalysis* is a rather young generic technique that was introduced by Khovratovich et al., first in 2011 [18], and later presented at the FSE 2012 [19]. Hereby, a biclique is a complete bipartite graph, where every element in a set of starting states is connected with every element in a set of ending states. In the context of cryptanalysis, every path in such a graph represents the encryption under a unique key over some steps of a primitive. If the paths do not share active non-linear components, then a biclique allows an adversary to test a set of key candidates very efficiently, which can be used to reduce the effort or to extend the number of steps in a meet-in-the-middle (MitM) or similar attack.
While their first applications targeted hash functions [18,19], bicliques have proven to be well-suited also for key-recovery attacks on block ciphers. In June 2011, Bogdanov et al. adapted the approach for their analysis of the AES. Their work received a high level of attention, since it showed the first attacks on full versions of the cipher in the single-key model [17,4]. Since then, biclique-based key-recovery attacks have been successfully applied to a variety of ciphers, including SQUARE, HIGHT, Piccolo, ARIA-256, L-Block, TWINE, IDEA, KLEIN-64, and mCrypton [1,9,7,13,16,20,24,26,27], all of these works being the first attacks on full-round versions of these ciphers.

Bicliques for bruteforce-like cryptanalysis. When there exists a MitM attack over the rounds not covered by a biclique, then an adversary can gain almost the same advantage as from the MitM attack. However, if there is no MitM attack available, one can still combine a biclique with an optimized bruteforce-like cryptanalysis, i.e., testing all keys over the remaining rounds; a technique which was introduced by Bogdanov et al. in [17,4] and which was essential to cover the full number of rounds in their attacks on the AES. Following this approach, the adversary precomputes and stores a number of computations and later re-uses the stored values to lower the total effort. As a consequence, the advantage for the adversary is rather low. The results mentioned above have gained a factor between 0.5 and 8. Moreover, since this approach is generic, it can be applied to any primitive and any number of rounds. Thus, a successfully mounted attack does not allow a cryptanalyst to identify cipher-specific weaknesses (cf. Wei et al. [28]). Thus, there is an ongoing debate on how to rate the importance of bruteforce-like cryptanalysis. Wei et al. stress that even small advantages can still be helpful for practical attacks, which applies to ciphers with a key length of 64-80 bits. For larger key lengths, such minor improvements were only of academic interest. We share this point of view; however, we generalize that biclique-based bruteforce-like cryptanalysis is an effective method to establish a new lower security margin for ciphers.

PRESENT. PRESENT is a 64-bit ultra-lightweight cipher that was proposed by Bogdanov et al. in 2007 [5] and is now standardized in the ISO/IEC Standard 29192-2. In their proposal, the designers described two versions of PRESENT; a recommended variant, which employs an 80-bit key (PRESENT-80), and a less important version with a 128-bit key (PRESENT-128). In 2008, Wang demonstrated a first differential analysis of a 16-round version of PRESENT, which had a time complexity equivalent to 2^{65} encryptions [25]. Later that year, Albrecht and Cid presented a combined differential-algebraic attack on 19 rounds with a time complexity equivalent to 2^{113} encryptions [2]. To the best of our knowledge, the previously most powerful attack on PRESENT without exhaustive components is the work of Cho [8], which allows to distinguish 25 rounds of PRESENT-80 from a random permutation. In the same work, the author proposed an additional attack on 26 rounds, which, in our opinion, is not a stronger analysis, since it requires the entire codebook. A few weeks after our initial publiciation of this work, Jeong et al. [15] published biclique-based bruteforce-like attacks on LED, PRESENT and Piccolo, which are similar to ours. Considering PRESENT, their attacks provide an extremely low advantage for the adversary, of 0.24 bits for PRESENT-80 and 0.19 bits for PRESENT-128.

LED. LED is a family of AES-like lightweight ciphers that was designed by Jian et al. in 2011 [12]. It supports arbitrary key lengths between 64 and 128 bits, where the two most-relevant versions are LED-64 and LED-128. In their security analysis, the designers of LED claimed that the best probabilistic differential attacks on LED could cover only 15 out of 32 rounds for the 64-bit and 27 out of 48 rounds for the 128-bit version. In [14], Isobe and Shibutani proposed a splice-and-cut attack on eight rounds of LED-64 with a computational complexity equivalent to 2^{56} encryptions. In the same work, the authors applied another splice-and-cut analysis on 16 rounds of LED-128 which required 2^{112} encryptions. Mendel et al. analyzed differential properties of LED and published related-key attacks on up to 16 rounds of LED-64 and up to 24 rounds of LED-128 [21]. At the FSE 2013, Nikolic et al. presented key-recovery attacks on up to eight rounds of LED-64 and 24 rounds of LED-128 (note, that we consider only the single-key model here), as well as chosen-key distinguishers on up to 16 rounds of LED-64 and up to 40 rounds of LED-128. Like it is the case for PRESENT, the best – when considering the number of covered rounds – previous attacks on LED in the single-key model are due to Jeong et al. [15], who describe an analysis of 29 rounds of LED-64 and 45 rounds of LED-128. Both results provide an advantage of about 0.5 bits. While the authors also consider an attack on full LED-128, we do not consider this as an attack, since it requires the full codebook.

KLEIN. Like LED, KLEIN is a family of AES-like lightweight block ciphers which was proposed by Gong, Nikova and Law at the RFIDSec 2011 [11]. The cipher has a 64-bit state and supports key lengths of 64, 80, or 96 bits. The best attacks on KLEIN before 2013 were published by Yu et al. [29], who proposed integral attacks on up to eight out of 12 rounds of KLEIN-64, and by Aumasson et al., who presented a practical distinguishing attack and several key-recovery attacks on up to eight rounds of this version. The best recent attack is due to Ahmadian et al., who published two biclique-based attacks on full KLEIN-64 with workloads of $2^{62.84}$ and $2^{62.81}$ encryptions and data complexities of 2^{39} and 2^{43} chosen plaintexts, respectively. So, since the security of KLEIN-64 has already been analyzed, we only considered the versions KLEIN-80 and KLEIN-96 in this work.

Contribution. In this paper, we describe our cryptanalytic results of biclique attacks on full PRESENT-80, PRESENT-128, LED-64, LED-128, KLEIN-80, and KLEIN-96. In addition, we give short descriptions of attacks on reduced versions of PRESENT, LED, and KLEIN-96 in order to illustrate the number of rounds required to obtain at least an advantage factor of two compared to a trivial exhaustive search. Our results are the best attacks on these ciphers in the single-key model regarding the number of rounds covered and

Primitive	Attack type	Rounds	Time	Data	Memory	Reference
PRESENT-80/-128	Differential	16	2^{65}	2^{64}	$6 \cdot 2^{29}$	[25]
	Diff. + Algebraic	19	2^{113}	-	-	[2]
	Saturation	24	2^{57}	2^{57}	2^{18}	[10]
	Linear	24	-	$2^{63.5}$	-	[23]
	Linear	25	2^{65}	$2^{62.4}$	2^{34}	[8]
	Linear	26	2^{72}	2^{64}	2^{34}	[8]
PRESENT-80	Biclique	17	$2^{78.76}$	2^{25}	$2^{34.6}$	Sec. 4.5
	Biclique	31 (full)	$2^{79.76}$	2^{23}	2^{7}	[15]
	Biclique	31 (full)	$2^{79.49}$	2^{25}	$2^{34.6}$	Sec. 4
PRESENT-128	Biclique	19	$2^{126.86}$	2^{23}	$2^{34.6}$	Sec. 5.5
	Biclique	31 (full)	$2^{127.81}$	2^{19}	2^{6}	[15]
	Biclique	31 (full)	$2^{127.32}$	2^{23}	$2^{34.6}$	Sec. 5
LED-64	Splice-and-cut	8	2^{56}	2^{8}	2^{11}	[14]
	Differential	8	2^{56}	2^{11}	2^{11}	[22]
	Biclique	16	$2^{62.95}$	2^{8}	2^{11}	Sec. 7.5
	Biclique	29	$2^{63.58}$	2^{40}	2^{11}	[15]
	Biclique	32 (full)	$2^{63.58}$	2^{8}	2^{11}	Sec. 7
LED-128	Splice-and-cut	16	2^{112}	2^{16}	2^{11}	[14]
	Differential	16	2^{96}	2^{32}	2^{35}	[22]
	Differential	24	$2^{124.4}$	2^{59}	2^{62}	[22]
	Biclique	32	$2^{126.99}$	2^{8}	2^{11}	Sec. 8.5
	Biclique	45	$2^{127.45}$	2^{32}	2^{11}	[15]
	Biclique	48 (full)	$2^{127.37}$	2^{64}	2^{11}	[15]
	Biclique	48 (full)	$2^{127.42}$	2^{8}	2^{11}	Sec. 8
KLEIN-64	Integral	7	-	-	-	[29]
	Differential	8	$2^{46.80}$	2^{32}	-	[29]
	Differential	8	2^{35}	2^{35}	-	[3]
	Biclique	12 (full)	$2^{62.84}$	2^{39}	$2^{7.5}$	[1]
	Biclique	12 (full)	$2^{62.81}$	2^{43}	$2^{7.5}$	[1]
KLEIN-80	Integral	8	-	-	-	[29]
	Biclique	16 (full)	$2^{79.00}$	2^{48}	2^{60}	Sec. 10
KLEIN-96	Biclique	16	$2^{95.00}$	2^{32}	2^{60}	Sec. 11
	Biclique	20 (full)	$2^{95.18}$	2^{32}	2^{60}	Sec. 11

Table 1. Best previously published attacks on PRESENT, LED, and KLEIN in the single-key model. Memory complexity is given in bytes, -: not given.

the computational complexity at the time of writing this paper. A comparison of previous works and our results is given in Table 1.

Outline. First, in Section 2, we recap the general details of independent-biclique cryptanalysis. Section 3 then provides the details of PRESENT, before we give a description of our attacks on PRESENT-80 and -128 in the Sections 4 and 5. Similarly, in Section 6, we first review the necessary details of LED, before Sections 7 and 8 explain our attacks on LED-64 and -128. Next, we have a short look on KLEIN in Section 9, and show attacks on full KLEIN-80 and KLEIN-96 in Sections 10 and 11. We conclude our paper in Section 12.

Revised aspects. In this version, we take care of the helpful comments we were given by the reviewers of the initial version of this paper. In our previous attacks, we had constructed bicliques over as many rounds as possible, which resulted in attacks with relatively high number of 2^{56} and 2^{64} required chosen plaintexts for LED-64 and LED-128, as well as 2^{60} and 2^{44} chosen plaintexts for PRESENT-80 and PRESENT-128, respectively. We re-constructed our bicliques in order to reduce the number of plaintexts which have to be collected by the adversary. In our attacks on LED, we could reduce the numbers to 2^8; for PRESENT, we could achieve 2^{25} and 2^{23} for PRESENT-80 and PRESENT-128.
Furthermore, concerning LED, we had mounted attacks on reduced versions without the wrapping key additions. Since the key injection is located after every fourth round, our reviewers pointed out the necessity of considering only attacks on full steps, i.e., 4-round intervals which include the wrapping key additions to be comparable to previous works. We revised our attacks on LED accordingly.

2 Biclique Cryptanalysis

In this section, we give a brief overview on biclique cryptanalysis based on the descriptions by Bogdanov et al. [4].

2.1 Definition

A biclique is a complete bipartite graph which connects every element in a set of starting states $\mathcal{S}$ with every element in a set of ending states $\mathcal{C}$. We represent the elements in $\mathcal{S}$ by S_j, and those in $\mathcal{C}$ by C_i. A path from S_j to C_i represents the encryption under some key $K[i,j]$ over some sub-cipher $\mathcal{B}$. The 3-tuple of sets $[\{S_j\},\{C_i\},\{K[i,j]\}]$ is called a *d-dimensional biclique*, if

$$\forall i,j \in \{0,\ldots,2^d-1\}: \quad S_j \xleftrightarrow[\mathcal{B}]{K[i,j]} C_i.$$

As a generalization, note that the sets $\mathcal{S}$ and $\mathcal{C}$ do not need to have identical numbers of elements. Then, we call the 3-tuple of sets $[\{S_j\},\{C_i\},\{K[i,j]\}]$ a (d_1,d_2)-dimensional (*asymmetric*) biclique, if

$$\forall i \in \{0,\ldots,2^{d_1}-1\}, \forall j \in \{0,\ldots,2^{d_2}-1\}: \quad S_j \xleftrightarrow[\mathcal{B}]{K[i,j]} C_i.$$

In the following, we regard the simple case of a d-dimensional biclique. Assume, an adversary is given a cipher E, on which she wants to mount a biclique-based attack. First, she divides the secret-key space into 2^{k-2d} subspaces of 2^{2d} keys each, where k denotes the key length and d the dimension of the used bicliques. Further, she defines a splitting $E = \mathcal{B} \circ E_2 \circ E_1$, where E_1 is the subcipher that maps a plaintext P to an internal state v, E_2 maps v to another internal state S and $\mathcal{B}$ maps the state at S to the ciphertext C:

$$P \xrightarrow{E_1} v \xrightarrow{E_2} S \xrightarrow{\mathcal{B}} C.$$

The adversary can construct a biclique over an arbitrary part of the cipher and can use a meet-in-the-middle or a bruteforce-like procedure to compute the remaining parts. Note, that she needs to have access to only either an encryption or decryption oracle to obtain plaintext-ciphertext pairs. This setting is illustrated in Figure 1 for a biclique over the last part of the cipher.
Bogdanov et al. introduced two different paradigms of bicliques for cryptanalysis: *independent bicliques*, which can be constructed with low effort, but cover only a small number of rounds, and *long bicliques*, which can potentially cover more rounds, but are harder to construct. In the following, we focus on the former approach.

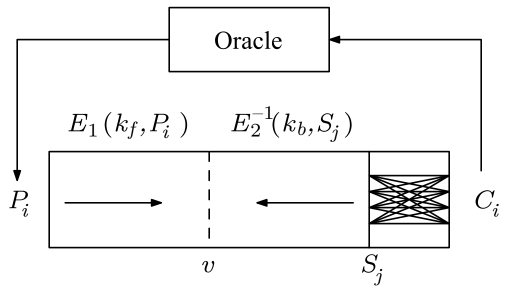


Fig. 1. MitM attack with a biclique at the end of the cipher.

2.2 Biclique Construction

Independent bicliques allow the construction of bicliques over some subcipher $\mathcal{B}$ from two differentials. The adversary chooses a so-called *base computation*, i.e., a 3-tupel $\{S_0, C_0, K[0,0]\}$, where the key $K[0,0]$ maps the internal state S_0 to the ciphertext C_0:

$$S_0 \xrightarrow[\mathcal{B}]{K[0,0]} C_0.$$

Then, she searches for 2^d forward differentials Δ_i, which connect the state S_0 with the ciphertexts C_i,

$$S_0 \xrightarrow[\mathcal{B}]{K[0,0]\oplus\Delta_i^K} C_0 \oplus \Delta_i = C_i,$$

and similarly, for 2^d backward differentials ∇_j, which connect the ciphertext C_0 with the states S_j:

$$S_j = S_0 \oplus \nabla_j \xleftarrow[\mathcal{B}^{-1}]{K[0,0]\oplus\nabla_j^K} C_0.$$

If the trails of all Δ_i-differentials *do not share active non-linear operations* with any of the ∇_j-differentials, then, there exists an encryption path connecting any of the 2^d input differences ∇_j with any of the 2^d output differences Δ_i. Thus, we obtain a set of 2^{2d} *independent* (Δ_i,∇_j)*-differential trails*:

$$S_0 \oplus \nabla_j \xleftrightarrow[\mathcal{B}]{K[0,0]\oplus\Delta_i^K\oplus\nabla_j^K} C_0 \oplus \Delta_i \qquad \forall i, j \in \{0, \ldots, 2^d - 1\}.$$

2.3 Matching-with-Precomputations

Khovratovich et al. introduced *matching-with-precomputations* as an effective technique to perform a matching on the parts not covered by the biclique. For this approach, an adversary first chooses an internal state v which splits the remaining parts into the sub-ciphers E_1 and E_2. Then, she precomputes and stores 2^d values $\overrightarrow{v_{i,0}}$ in forward direction from the plaintext P to v, and 2^d values $\overleftarrow{v_{0,j}}$ in backward direction from each of the starting states S_j:

$$P_i \xrightarrow[E_1]{K[i,0]} \overrightarrow{v_{i,0}}, \quad \text{and} \quad \overleftarrow{v_{0,j}} \xleftarrow[E_2^{-1}]{K[0,j]} S_j.$$

For all $2^{2d} - 2^d$ further computations

$$P_i \xrightarrow[E_1]{K[i,j]} \overrightarrow{v_{i,j}}, \quad \text{and} \quad \overleftarrow{v_{i,j}} \xleftarrow[E_2^{-1}]{K[i,j]} S_j,$$

the adversary has to recompute only those parts of the key schedule and the round transformation that differ from the stored values. By using this method, the computational effort for matching can be reduced significantly compared to an exhaustive search. A further reduction is possible by matching only in a part of the state at v (*partial matching*).

2.4 Complexity Calculations

For every biclique, the adversary tests 2^{2d} keys. Hence, it needs to construct 2^{k-2d} bicliques to cover the full key space. Concerning the time complexity, [4] proposed the equation

$$C_{full} = 2^{k-2d} \left(C_{biclique} + C_{decrypt} + C_{precomp} + C_{recomp} + C_{falsepos} \right), \tag{1}$$

where

- $C_{biclique}$ denotes the costs for computing $2 \cdot 2^d$ trails over $\mathcal{B}$,
- $C_{decrypt}$ is the complexity of the oracle to decrypt 2^d ciphertexts,
- $C_{precomp}$ represents the effort for 2^d computations of $E_2 \circ E_1$ to determine $\overleftarrow{v_{0,j}}$ and $\overrightarrow{v_{i,0}}$.
- C_{recomp} describes the costs of recomputing 2^{2d} values $\overleftarrow{v_{i,j}}$ and $\overrightarrow{v_{i,j}}$, and
- $C_{falsepos}$ is the complexity to eliminate false positives.

The full computational effort of the attack is dominated by the recomputations. The memory requirements are upper bounded by storing 2^d intermediate states $v_{i,j}$. Note, that in attacks with a low data complexity, it can be appropriate to ask and store all required plaintext-ciphertext pairs in advance, so that $C_{decrypt}$ becomes a negligible term in the full time complexity.

2.5 Sieve-in-the-Middle

The number of rounds covered by a MitM or a biclique-based attack can be further extended by precomputing a number of steps near the matching state, as announced by Canteaut et al. [6]. Prior to an attack, the adversary creates a precomputation table for the possible transitions through some sub-cipher S in the middle of a given cipher. In the MitM attack, instead of matching by computing the same bits of a matching state from both directions, the adversary can already stop at the input and output steps of S, and use the precomputed table to look, if inputs and outputs produce a valid transition to sieve out false keys. Thus, this procedure transforms the search for collisions at a certain matching point into the search for valid transitions. Assume, we are given an $m \times m$-S-box, i.e., inputs and outputs have a state length of m bit. We say that a transition through the S-box exists with a probability p. Further, we denote by n_{in} the number of fixed known input bits, by n_{out} the number of known output bits, then, there exist at most $2^{m-n_{in}}$ values for the output bits. The probability p for a valid transition is then given by

$$p \leq \frac{2^{m-n_{in}}}{n_{out}}.$$

To have an effective sieve, the adversary requires to have an a-priori probability of $p < 1$. Thus, we require $n_{in} + n_{out} > m$.

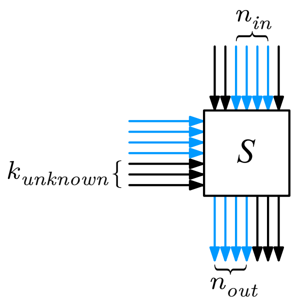


Fig. 2. Sieve in the middle. Colored trails indicate known, black trails unknown bits.

If the sieve considers key bits, which we denote by k_s, the number of unknown key bits has to be taken into account as a further parameter. If one denotes by $k_{unknown}$ the number of unknown key bits in a sieve, one obtains $2^{n_{in}+n_{out}-m-k_{unknown}}$ bits for sieving.

3 Brief Description Of PRESENT

3.1 Round Transformation

PRESENT is a 64-bit lightweight cipher which transforms the state in 31 rounds of a substitution-permutation network. After the final round, the state is XORed with a post-whitening round key to generate the ciphertext. Every round consists of three operations: a key addition with a round key (AK), a non-linear substitution layer (SL) and a permutation layer (PL), as shown in Figure 3 (cf. [5]).

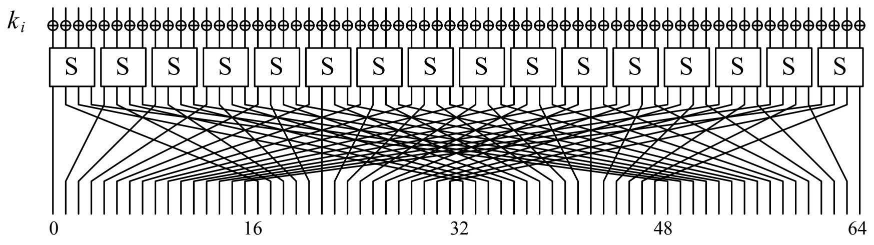


Fig. 3. Round structure in PRESENT. Each trail represents a bit.

3.2 Key Schedule

The key schedule of PRESENT expands the secret key to 32 round keys. At the beginning, the secret key is stored in a register. After extracting the most-significant 64 bits as the initial round key RK^1, the register is updated with a rotation by 61 positions to the left, an S-box call, and an XOR operation with a round counter r. This procedure is repeated 31 times until all round keys are generated.
For the 80-bit version, we denote the state of the register by $(k_{79}, k_{78}, \ldots, k_1, k_0)$, where k_i represents the i-th, and k_{79} the most-significant bit of the key. A formal description of the update function for the key register, which creates the round key RK^{r+1}, can be written as follows:

– $(k_{79}, k_{78}, \ldots, k_1, k_0) = (k_{18}, k_{17}, \ldots, k_1, k_0, k_{79}, k_{78}, \ldots, k_{20}, k_{19})$
– $(k_{79}, k_{78}, k_{77}, k_{76}) = Sbox(k_{79}, k_{78}, k_{77}, k_{76})$
– $(k_{19}, k_{18}, k_{17}, k_{16}, k_{15}) = (k_{19}, k_{18}, k_{17}, k_{16}, k_{15}) \oplus r$.

Similarly, the key schedule for the 128-bit version can be described by:

– $(k_{127}, k_{126}, \ldots, k_1, k_0) = (k_{66}, k_{65}, \ldots, k_1, k_0, k_{127}, k_{126}, \ldots k_{68}, k_{67})$
– $(k_{127}, k_{126}, k_{125}, k_{124}) = Sbox(k_{127}, k_{126}, k_{125}, k_{124})$
– $(k_{123}, k_{122}, k_{121}, k_{120}) = Sbox(k_{123}, k_{122}, k_{121}, k_{120})$
– $(k_{66}, k_{65}, k_{64}, k_{63}, k_{62}) = (k_{66}, k_{65}, k_{64}, k_{63}, k_{62}) \oplus r$.

Table 2 in Appendix A summarizes the secret-key bits on which the round keys RK^i depend.

3.3 Sieve-in-the-Middle

We can apply the sieve-in-the-middle approach to reduce the recomputational effort in our attacks. Hereby, we can exploit the limited single-round diffusion of PRESENT. More precisely, over the operation sequence $E_s = \text{SL} \circ \text{PL} \circ \text{AK} \circ \text{SL}$, we can identify four distinct groups of 16 bits of the output, which depend on only 16 bits of the input, and only 16 bits of the key. Thus, we can precompute a table $T_{\overrightarrow{v_{i,j}}, \overleftarrow{v_{i,j}}}$ which stores the transitions $\overrightarrow{v_{i,j}} \leftrightarrow \overleftarrow{v_{i,j}}$ for 2^{16} possible inputs $\overrightarrow{v_{i,j}}$ and 2^{16} possible key bits in RK^i over E_s with practical effort.

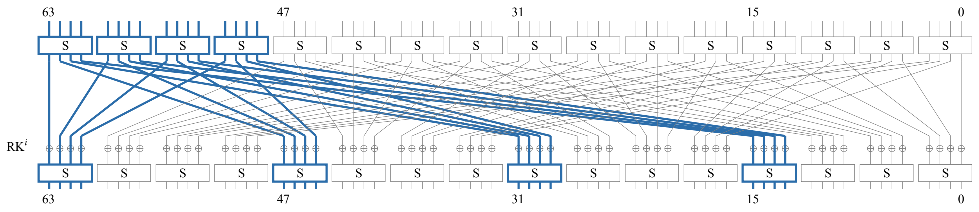


Fig. 4. Sieve over the steps SL ∘ PL ∘ AK ∘ SL for PRESENT.

In our attacks, we consider one of the four groups, which contains the most-significant 16 bits for the inputs, and the bits at the indices (63, ..., 60, 47, ..., 44, 31, ..., 28, 15, ..., 12) for the key and outputs. The trails of these bits through E_s are shown in Figure 4.
Our resulting table stores 2^{32} entries under the indices $(\overrightarrow{v_{i,j}} \| \overleftarrow{v_{i,j}} \| k_s)$, i.e., we require $2^{32} \cdot \frac{16+16+16}{8} = 3 \cdot 2^{33}$ bytes. Note, that the effort for constructing the table is a negligible summand in our attacks on PRESENT.

4 Independent-Biclique Attack On Full PRESENT-80

In this section, we describe an independent-biclique attack on PRESENT-80. The attack consists of three steps: partitioning the key space, constructing a biclique, and performing a matching over the remaining rounds. At the end of this section, we explain the resulting complexities of the attack in detail.

4.1 Key Space Partitioning

We divide the 80-bit key space into 2^{66} sets of 2^{14} keys each with respect to the key register state after extraction of the round key RK^{30}. Since the key schedule of PRESENT is a bijective mapping of every register state to a unique value of the secret key, this partitioning covers the full secret-key space.
The base keys $K[0,0]$ of our sets are all 80-bit secret keys with 14 bits fixed to zero, whereas the remaining 66 bits iterate over all possible values. The keys in a set $\{K[i,j]\}$ are defined relative to the base key $K[0,0]$ and two differences Δ_i^K and ∇_j^K, where $i, j \in \{0, \ldots, 2^7 - 1\}$. Hereby, we chose the key differences Δ_i^K to iterate over all values of the bits $(k_{61}, k_{60}, k_{59}, k_{58}, k_{57}, k_{56}, k_{55})$ in the forward trails, and the differences ∇_j^K over all values for the bits $(k_{36}, k_{35}, k_{34}, k_{33}, k_{32}, k_{31}, k_{30})$ in the backward trails.

4.2 3-Round Biclique Of Dimension 7

We limited our bicliques to cover only three rounds, and we chose the key differences Δ_i^K so that we do not have active bits in RK^{29}. Therefore, we could obtain a relatively low data complexity for this attack. As a result, we construct bicliques of dimension seven over the rounds 29-31. Figure 5 visualizes the Δ_i- and ∇_j-trails with red and blue lines, respectively. As one can see, the trails do not share active non-linear components (here, S-boxes), and are therefore independent. Additionally, we stress that the red trails affect only 25 bits of the ciphertexts C_i. Hence, an adversary who fixes the ciphertexts C_0 over all bicliques, will have to collect at most 2^{25} chosen ciphertexts to mount this attack.

4.3 Matching Over 28 Rounds

In the following, we apply a matching-with-precomputations procedure over the rounds 1-28. Hereby, we locate the states $\overrightarrow{v_{i,j}}$ after Round 14 and the states $\overleftarrow{v_{i,j}}$ before the S-box layer of Round 16. At each of these states, we want to reconstruct 12 out of 16 bits which serve as input to our sieve E_s. Considering the complexity of the attack, we are mostly interested in the number of operations which have to be recomputed. In order to have a single number which refers best to the total effort, we interpret PRESENT as a nibble-wise

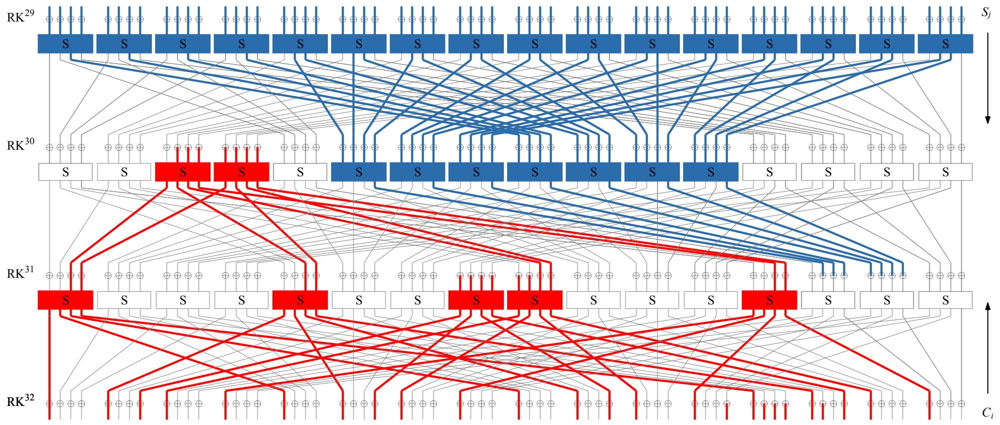


Fig. 5. The biclique over the rounds 29-31 in our attack on full PRESENT-80. The red colored trails indicate bits affected by the differences Δ_i^K, the blue trails bits affected by the differences ∇_j^K.

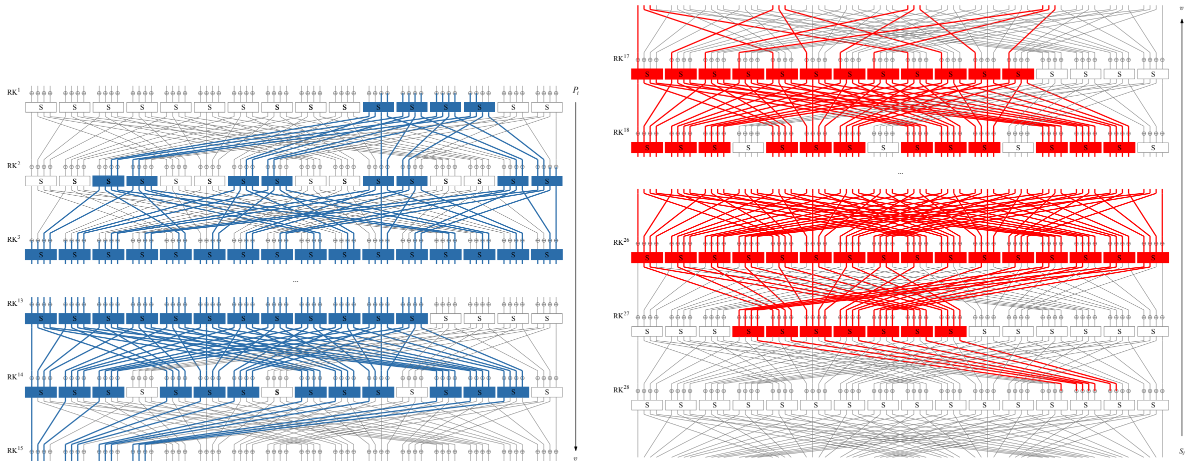


Fig. 6. Matching for the attack on full PRESENT-80 in forward direction from the plaintexts P_i to $v_{i,j}$ (left), and in backward direction from the states S_j to $v_{i,j}$ (right).

operating cipher and approximate the recomputation costs in the matching part by counting the number of required S-box operations in the round transformation and the key schedule. These operations are visualized by the blue trails in forward direction in Figure 6. All non-highlighted parts of the states and round keys can be used from the precomputed values. As one can see, there are four active S-boxes in the first round, eight in the second round, $10 \cdot 16$ in the rounds 3-12, 12 S-boxes in Round 13, 12 S-boxes in Round 14, and four in Round 15. In every forward computation, this sums up to 196 active S-boxes.
The operations which have to be recomputed in backward direction are shown by the red trails in Figure 6. Again, all non-highlighted trails in the round transformation can be used from the precomputed values. This time, there are seven active S-boxes in Round 27, $8 \cdot 16$ in the rounds 19-26, 12 in Round 18, 12 in Round 17, and four in Round 16, which yields 159 S-boxes for this part. In addition, we have to consider the S-box operations which have to be recomputed in the key schedule. Concerning the Δ_i^K-differences, we have to invoke the S-box to compute the round keys RK^{25}, RK^{21}, and RK^{17}. For the ∇_j^K-differences, we have to recompute five S-boxes, namely to obtain the updated values of the RK^3, RK^3, RK^7, RK^{11}, RK^{24}, and RK^{28}. In total, the recomputations sum up to 363 S-boxes.

4.4 Complexity

The computational complexity of the attack is given by our equation

$$C_{full} = 2^{k-2d}\left(C_{biclique} + C_{precomp} + C_{recomp} + C_{falsepos}\right).$$

The full cipher consists of 31 rounds, where every round has 16 S-box operations in the input transformation and one in the key schedule. Hence, the recomputation costs can be approximated by $C_{recomp} \approx 2^{14} \cdot \frac{363}{31\cdot(16+1)} \approx 2^{13.46}$ full encryptions. The effort for constructing a biclique requires the computation of 2^8 times three out of 31 rounds, or approximately $C_{biclique} \approx 2^{4.63}$ full encryptions. The precomputational costs are given by computing 2^7 times 28 out of 31 rounds, which is equivalent to $C_{precomp} \approx 2^{6.85}$ encryptions. Note, that we can ask for the decryption of our 2^{25} chosen ciphertexts before the attack, so we have to consider the effort for $C_{decrypt}$ only once.
It remains to clarify the complexity to eliminate false positives. As mentioned above, we reconstruct 12 bits of $\overrightarrow{v_{i,j}}$ and 12 bits of $\overleftarrow{v_{i,j}}$. Since four input bits (from $\overrightarrow{v_{i,j}}$) to the sieve are unknown, there are at most $2^{16-12} = 2^4$ possible output values for $\overleftarrow{v_{i,j}}$. The chance that a false positive key $K[i,j]$ matches in all 12 known bits of $\overleftarrow{v_{i,j}}$ is therefore $2^4 \cdot 2^{-12} = 2^{-8}$. For a set of 2^{14} keys, we can expect to obtain $C_{falsepos} = 2^{14-8} = 2^6$ false positives in average, that have to be tested with a full encryption operation each. In total, the time complexity of the attack is given by

$$C_{full} = 2^{66} \cdot (2^{4.63} + 2^{6.85} + 2^{13.46} + 2^6) + 2^{25} \approx 2^{79.49} \text{ encryptions.}$$

Concerning the memory complexity, we have to store 2^{25} states, or 2^{28} bytes for the attack, and $3 \cdot 2^{33}$ bytes for the sieve, which sums up to approximately $2^{34.6}$ bytes.

4.5 Independent-Biclique Attack On 17 Rounds Of PRESENT-80

In general, it is desirable to have an advantage of at least one power of two compared to exhaustive search for the computational complexity. Therefore, we can mount an attack on a reduced version of PRESENT-80, consisting of the rounds 15-31, by using the same biclique structure and matching procedure as above. This time, we locate the matching states $\overrightarrow{v_{i,j}}$, i.e., the inputs to the sieve-in-the-middle at the states after Round 19, and the states $\overleftarrow{v_{i,j}}$, i.e., the outputs of the sieve, at the states before the S-box layer of Round 21. Note, that we use the same input and ouput bits as in the attack above for matching.
As highlighted by the blue trails in Figure 7, in the forward part, we now have to recompute two S-boxes in Round 16, ten in Round 17, 12 in Round 18, and 12 in Round 19. Hence, the forward recomputations requires 36 S-box operations.
Considering the recomputations in the backward part, one can see in the red trails in Figure 7 that there are still seven active S-boxes in Round 27, $3 \cdot 16$ S-boxes in the rounds 24-26, 12 in Round 23, and 12 in Round 22, which sums up to 79 S-box operations for this part. Concerning the key schedule, we have to recompute only three S-boxes; for the differences Δ_i^K one for the key RK^{25}, and two for the differences ∇_j^K, namely to obtain the round keys RK^{24} and RK^{28}. Thus, we have to recompute 118 S-box operations in total.
Hence, C_{recomp} is equal to $2^{14} \cdot \frac{118}{17\cdot(16+1)} \approx 2^{12.71}$ encryptions. The construction of the biclique requires to compute 2^8 times three out of 17 rounds or $2^{5.50}$ encryptions, $C_{precomp}$ is given by 2^7 computations of 14 out of 17 or $2^{6.72}$ encryptions, and $C_{falsepos}$ can be expected to be 2^6 in average. The total time complexity then results from

$$C_{full} = 2^{66} \cdot (2^{5.50} + 2^{6.72} + 2^{12.71} + 2^6) + 2^{25} \approx 2^{78.76} \text{ encryptions.}$$

The data and memory complexities remain the same as in the attack on full PRESENT-80.

5 Independent-Biclique Attack On Full PRESENT-128

This section describes our attack on the full version of PRESENT-128.

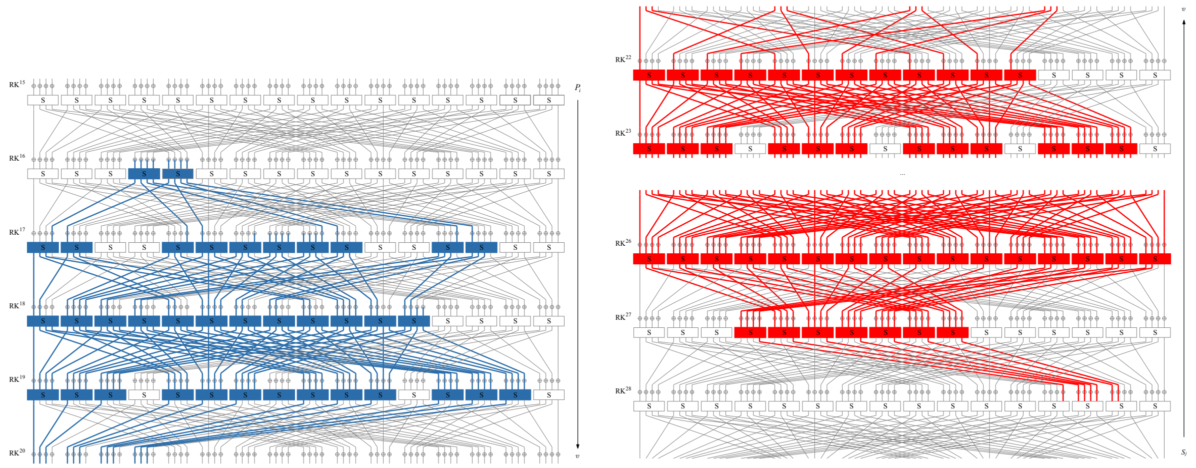


Fig. 7. Matching for the attack on 17-round PRESENT-80 in forward direction from the plaintexts P_i to $v_{i,j}$ (left), and in backward direction from the states S_j to $v_{i,j}$ (right). The colored trails indicate the parts of the state that have to be recomputed.

5.1 Key Space Partitioning

This time, we divide the key space into 2^{117} sets of 2^{11} keys each, again with respect to the key register state after extraction of the round key RK^{30}. The base keys $K[0,0]$ of our sets are all 128-bit keys with 11 bits fixed to zero, whereas the remaining 117 bits iterate over all possible values. The keys in a set $\{K[i,j]\}$ are defined relative to the base key $K[0,0]$ and two differences Δ_i^K and ∇_j^K, where $i \in \{0,\ldots,2^3-1\}$ and $j \in \{2^7-1\}$. We adapted from [15] the choice of the key differences Δ_i^K to iterate over all values of the bits (k_{19}, k_{18}, k_{17}). In addition, for the key differences ∇_j^K, we iterate over the bits $(k_{55}, k_{54}, \ldots, k_{48})$.

5.2 4-Round Biclique Of Dimension $(3, 8)$

Here, we construct bicliques of dimension $(3, 8)$ over the final four rounds. By this choice, we profit from the low number of active bits in the differences Δ_i^K, which allows us to have a low data complexity in the attack. On the same time, the higher number of active bits in the differences ∇_j^K allows us to test more keys for one biclique than with a 3-dimensional biclique. Therefore, the effort for precomputations and decryptions have a lower influence on the total time complexity of the attack.
Figure 8 shows the independent Δ_i- and ∇_j-differentials as red and blue trails. As one can see from the figure, the red trails affect only 23 bits of the ciphertexts. Hence, an adversary who fixes the ciphertexts C_0 over all bicliques, will have to collect at most 2^{23} chosen ciphertexts to mount this attack.

5.3 Matching Over 27 Rounds

We apply a matching-with-precomputations procedure over the rounds 1-27, where the states $\overrightarrow{v_{i,j}}$ are placed after Round 19 and the states $\overleftarrow{v_{i,j}}$ before the S-box layer of Round 21. Again, we want to reconstruct 12 out of 16 bits of each state $\overrightarrow{v_{i,j}}$ and $\overleftarrow{v_{i,j}}$ which serve as input to our sieve E_s.
The operations which have to be recomputed in forward and backward direction are highlighted in Figure 9 by the blue and red trails, respectively. From the figure, we can see that there are four active S-boxes in the second round, eight in the third round, $14 \cdot 16$ in the rounds 4-17, 12 in Round 18, and 12 in Round 19, which yields 260 active S-boxes in forward direction. In the backward computations, we have to take into account three active S-boxes in Round 26, 12 in Round 25, 16 in Round 24, 12 in Round 23, and 12 in Round 22, summing up to 55 S-boxes for this part. In addition, we have to consider five S-boxes in the key schedule:

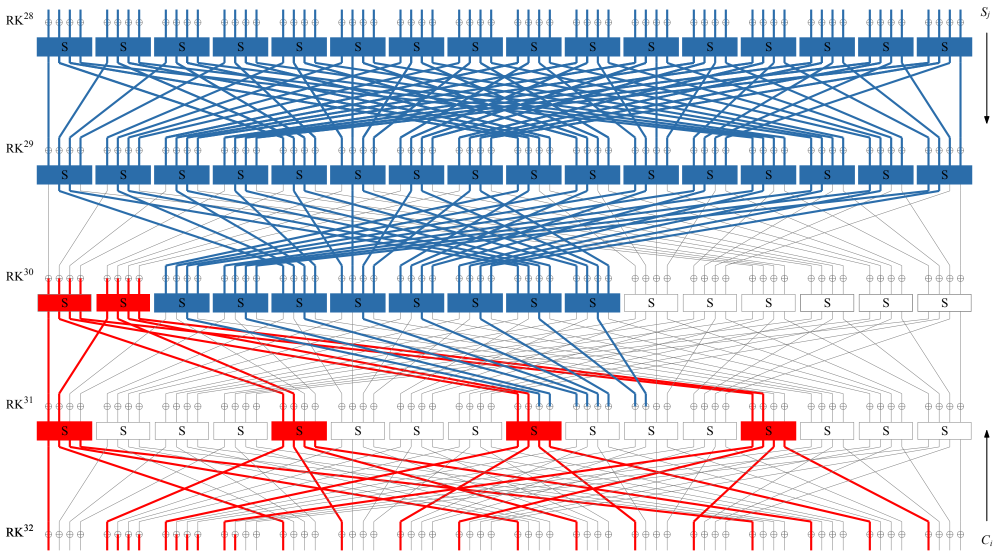


Fig. 8. The biclique over the rounds 28-31 in our attack on full PRESENT-128. The red colored trails indicate bits affected by the differences Δ_i^K, the blue trails bits affected by the differences ∇_j^K.

one S-box to recompute the round key RK^{28} in the differences Δ_i^K. In addition, there are four S-boxes to recompute the round keys for the differences ∇_j^K; one S-box to recompute RK^{21}, two S-boxes to recompute RK^{19}, and one to recompute RK^{17}. Hence, this sums up to $260 + 55 + 5 = 320$ S-box operations.

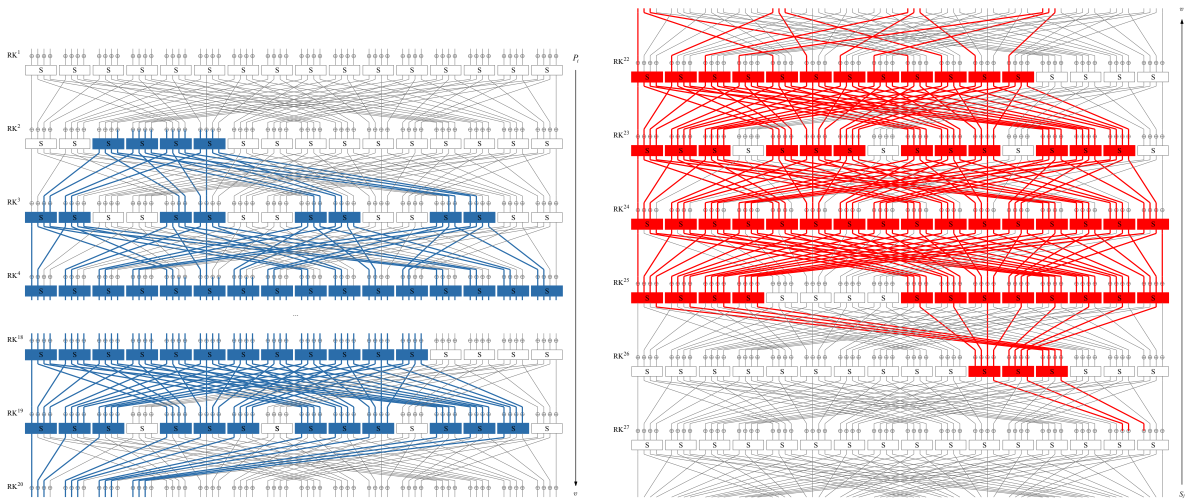


Fig. 9. Matching for the attack on full PRESENT-80 in forward direction from the plaintexts P_i to $v_{i,j}$ (top), and in backward direction from the states S_j to $v_{i,j}$ (bottom). The colored trails indicate the parts of the state that have to be recomputed.

5.4 Complexity

PRESENT-128 invokes the S-box 16 times in each of its 31 rounds, and two times in every iteration of the key schedule. Hence, we can approximate the recomputation costs C_{recomp} by $2^{11} \cdot \frac{320}{31 \cdot (16+2)} \approx 2^{10.20}$ full encryptions. The effort for constructing a biclique, $C_{biclique}$, concerns the computation of $(1+2^3-1+2^8-1) = 263$ times four out of 31 rounds, or approximately $2^{5.09}$ full encryptions. Furthermore, $C_{precomp}$ is given by computing 2^3 times 20 out of 31 rounds, and 2^8 times seven out of 31 rounds, which is equivalent to $2^3 \cdot \frac{20}{31} + 2^8 \cdot \frac{7}{31} \approx 2^6$ encryptions. Like in the attack on PRESENT-80, the probability for a false positive is 2^{-8} for each key, as in our attack on PRESENT-80. Therefore, we can expect to obtain $C_{falsepos} = 2^3$ false positives in average. All together, the time complexity of this attack is given by

$$C_{full} = 2^{117} \cdot (2^{5.09} + 2^6 + 2^{10.20} + 2^3) + 2^{23} \approx 2^{127.32} \text{ encryptions.}$$

We ask a decryption oracle only once for the decryptions of all occuring ciphertexts C_i in the attack, and store them. Concerning the memory complexity, we have to store 2^{23} plaintexts or 2^{26} bytes for the attack, and $3 \cdot 2^{33}$ bytes for the sieve, which sums up to $2^{34.6}$ bytes.

5.5 Independent-Biclique Attack On 19 Rounds Of PRESENT-128

To obtain an advantage of at least one half of the total effort, we mount an attack on a version of PRESENT-128 reduced to the rounds 13-31 with the same biclique and matching procedure as in the attack on full PRESENT-128. As highlighted by the blue trails in Figure 10, in the forward part, we now have to recompute

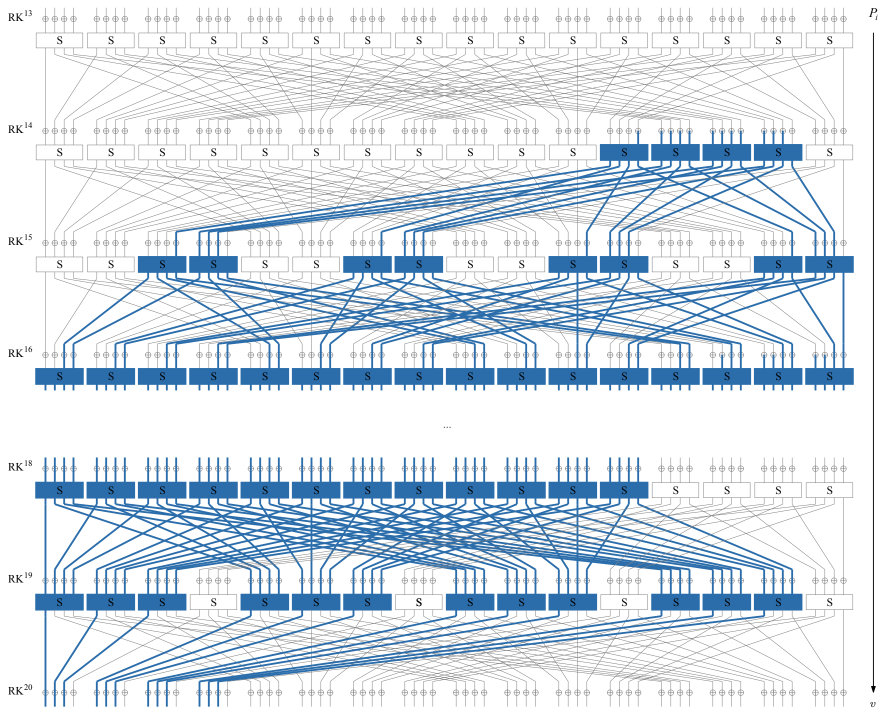


Fig. 10. Forward part of the matching for the attack on 19-round PRESENT-128 from the plaintexts P_i to $v_{i,j}$. The colored trails indicate the parts of the state that have to be recomputed.

four S-boxes in Round 14, eight in Round 15, $2 \cdot 16$ in the rounds 16-17, plus 12 in Round 18 and 13 in Round

19, which sums up to 68 S-box operations. Considering the backward part, there are still 55 active S-boxes in the rounds 22-27, and still five S-boxes in the key schedule, which sums up to 128 S-boxes in total. The construction of the biclique requires to compute 263 times four out of 19 rounds or $2^{5.79}$ encryptions, and we have to compute 2^3 times seven out of 19 rounds, and 2^8 times seven out of 19 rounds, which is equivalent to $2^{6.61}$ encryptions. The recomputations costs can be approximated by $2^{11} \cdot \frac{128}{19 \cdot (16+2)} \approx 2^{9.58}$ full encryptions. And we can expect, again, 2^3 false positives in average. Therefore, the full computational effort results of the attack can be approximated by

$$C_{full} = 2^{117} \cdot (2^{5.79} + 2^{6.61} + 2^{9.58} + 2^3) + 2^{23} \approx 2^{78.76} \text{ encryptions.}$$

The data and memory complexities remain the same as in the attack on full PRESENT-128.

6 Brief Description Of LED

LED is an AES-like substitution-permutation network, which transforms a 64-bit text input in 32 rounds for LED-64, and in 48 rounds for LED-128. The internal state of the cipher is represented by a 4×4-matrix where every cell in the matrix represents a nibble. The secret key is filled into one 4×4-word K or two words K_1 and K_2, depending on the key length. For the key lengths from 65 to 128 bits, the first 64 bits of the given key are used for K_1 and the remaining key is padded with zeroes to fill up K_2.

6.1 Round Transformation

The encryption process of LED consists of two operations, `AddRoundKey` ($\texttt{AK}[K_i]$) and `step`, as shown in Figure 12. The `step` operation itself contains four AES-like rounds, where each round includes the operations `AddConstants` (`AC`), `SubCells` (`SC`), `ShiftRows` (`SR`), and `MixColumnsSerial` (`MCS`) (see Figure 11):

$$\begin{aligned} \texttt{round} &= \texttt{MCS} \circ \texttt{SR} \circ \texttt{SC} \circ \texttt{AC} \\ \texttt{step} &= \texttt{round} \circ \texttt{round} \circ \texttt{round} \circ \texttt{round} \\ \texttt{LED} &= \texttt{AK}[K_1] \circ \texttt{step} \circ \texttt{AK}[K_2] \circ \texttt{step} \ldots \texttt{step} \circ \texttt{AK}[K_1]. \end{aligned}$$

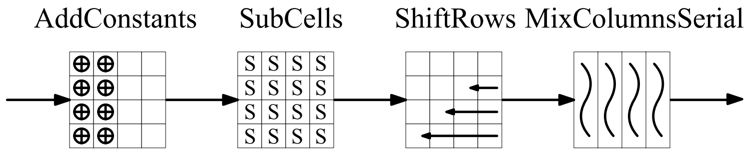


Fig. 11. One round in LED.

6.2 Key Schedule

LED omits a key schedule. In the 64-bit version, the secret key is used in simply every `AddRoundKey` operation, while in all larger versions, the key words K_1 and K_2 are used alternatingly. For more details on the individual operations, we would like to refer the interested reader to the original proposal of LED [12].

7 Independent-Biclique Attack On Full LED-64

This part includes the description of an attack on full LED-64.

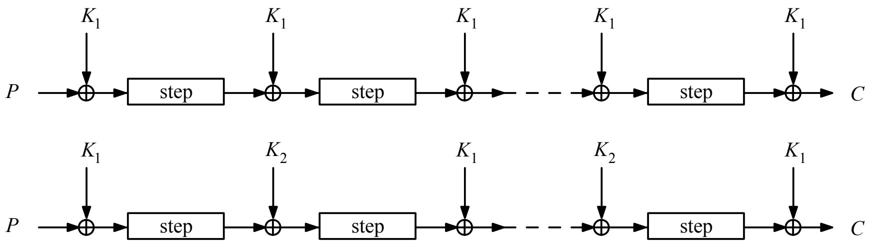
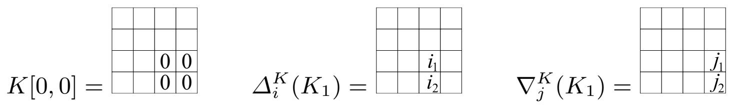


Fig. 12. Round structure of LED with 64-bit key (top) and 128-bit key (bottom).

7.1 Key Space Partitioning

We divide the 64-bit key space into sets of 2^{16} keys with respect to the secret key. The base keys $K[0,0]$ are all 64-bit secret keys with 16 bits fixed to zero, whereas the remaining 48 bits iterate over all possible values. The 2^{16} keys in a set $\{K[i,j]\}$ are defined relative to the base key $K[0,0]$ and two differences Δ_i^K and ∇_j^K, where $i, j \in \{0, \ldots, 255\}$ and $i = (i_1 \| i_2)$ and $j = (j_1 \| j_2)$.



7.2 4-Round Biclique Of Dimension 8

Our biclique covers the rounds 29-32, including the final key addition, as shown in Figure 13. Obviously, the Δ_i- and ∇_j-trails are independent, since the key addition is located at the end of the differential trails. One can see, that only two nibbles are active in the ciphertexts of the Δ_i-differentials. Thus, by fixing the ciphertext C_0 over all bicliques, we need at most 2^8 chosen ciphertexts for the attack.

7.3 Matching Over 28 Rounds

We locate the matching state v after Round 3 and match in two nibbles, as shown in Figure 14. The round transformation of LED employs constant additions, key additions, S-boxes, row shifts and column-wise multiplications. Following the argumentation from Bogdanov et al. [4], the bottleneck of AES-like cipher implementations is given by the number of S-box calls. We have a negligible number of key and constant additions, compared to the number of S-box calls. So, we can neglect the XOR and shift operations and consider the number of `MixColumnsSerial` and `SubCell` operations. Since the number of S-box calls is the larger summand compared to the number of mixing operations, we consider only the number of S-boxes.
In the first three rounds, we have to recompute $2+4+4 = 10$ S-boxes in forward direction. In addition, there are $2 + 8 + 22 \cdot 16 + 4 = 366$ S-boxes which have to be recomputed in backward direction, which sum up to 376 S-boxes in the full matching phase.

7.4 Complexity

There are $32 \cdot 16 = 512$ S-boxes in the 32 rounds of the full cipher. Hence, C_{recomp} is equivalent to $2^{16} \cdot \frac{376}{512} \approx 2^{15.55}$ encryptions. The precomputation effort $C_{precomp}$ is given by 2^8 computations of 28 out of 32 rounds,

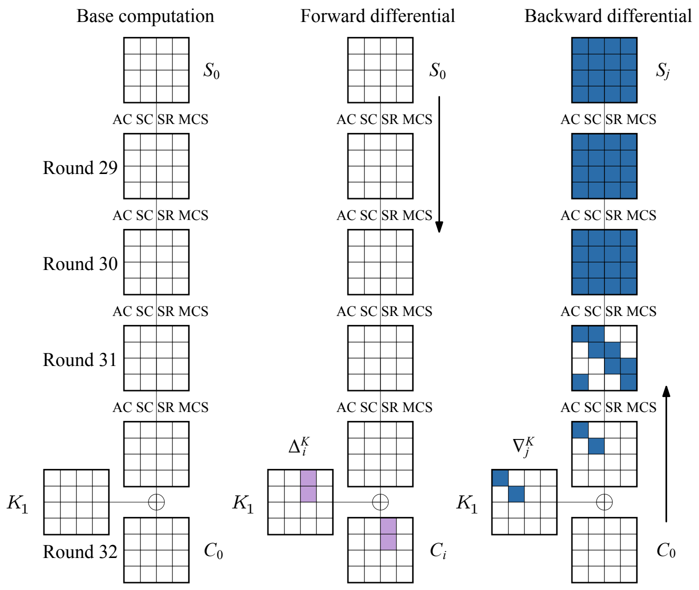


Fig. 13. The biclique on LED-64 over the rounds 29-32 with Δ_i- and ∇_j-differentials.

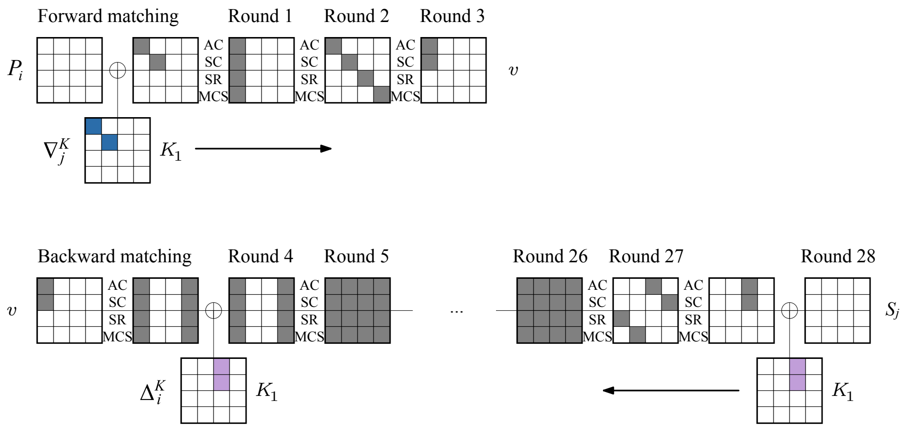


Fig. 14. Recomputations for LED-64 in forward and backward direction.

or $2^{7.81}$ full encryptions, and $C_{biclique}$ represents the costs for computing 2^9 times four out of 32 rounds, or $2^{6.19}$ encryptions. Thus, the total computational complexity of this attack results from

$$C_{full} = 2^{48} \cdot (2^{6.19} + 2^{7.81} + 2^{15.55} + 2^8 + 2^8) \approx 2^{63.58} \text{ encryptions.}$$

The attack requires memory to store 2^8 states, or 2^{11} bytes.

7.5 Independent-Biclique Attack On 16-Round LED-64

We can mount an attack on a reduced version of the first 16 of LED-64 by using the same key space partitioning, biclique, and partial matching procedure as before. This time we locate the biclique to cover the rounds 13-16. Then, we require to recompute 16 full rounds less. We still have to recompute $2+4+4 = 10$ S-boxes in forward direction; though, the recomputation effort in the backward direction reduces to $2+8+6\cdot 16+4 = 110$ S-boxes, which gives a total number of 120 S-boxes to recompute.

Since there are 256 S-boxes in the 16 rounds of the cipher, C_{recomp} is equivalent to $2^{16} \cdot \frac{120}{256} \approx 2^{14.91}$ encryptions. The effort for constructing one biclique can be approximated by computing 2^9 times four out of 16 rounds or $2^{7.42}$ full encryptions. $C_{precomp}$ is given by 2^8 computations of 12 out of 16 rounds or $2^{7.58}$ encryptions. The full computational complexity therefore sums up to

$$C_{full} = 2^{48} \cdot (2^{7.42} + 2^{7.58} + 2^{14.91} + 2^8 + 2^8) \approx 2^{62.95} \text{ encryptions.}$$

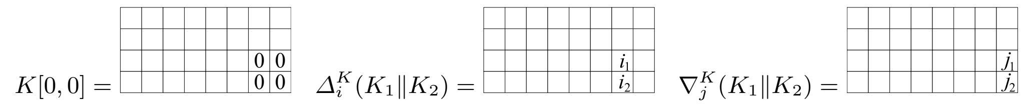
As before, this attack requires the adversary to collect 2^8 chosen plaintexts, and memory to store 2^{11} bytes.

8 Independent-Biclique Attack On Full LED-128

In this part, we describe an independent-biclique attack on full LED-128.

8.1 Key Space Partitioning

This time, we divide the key space into 2^{112} sets of 2^{16} keys. The base keys $K[0,0]$ are all 128-bit secret keys with 16 bits fixed to zero, whereas the remaining 112 bits iterate over all possible values. The 2^{16} keys in a set $\{K[i,j]\}$ are defined relative to the base key $K[0,0]$ and two differences Δ_i^K and ∇_j^K, where $i, j \in \{0, \dots, 255\}$ and $i = (i_1 \| i_2)$ and $j = (j_1 \| j_2)$.



8.2 8-Round-Biclique Of Dimension 8

At most, for LED-128, one could construct bicliques over up to 4 steps, without the wrapping key additions. Since, first, we count only full steps, and second, we aim at obtaining a low data complexity, we decided to limit the biclique to two steps, covering the rounds 41-48. Figure 15 illustrates the Δ_i- and ∇_j-differentials. Since, like in the attack on LED-64, the differences Δ_i^K affect the state only in the very last operation, both trails are independent from each other.

8.3 Matching Over 32 Rounds

The matching then covers the rounds 1-40. We locate v in the state after Round 7 and match in two nibbles, as shown in Figure 16. One can see from there, that the rounds 1 through 4 are not affected by active bits from the ∇_j^K-differences. Thus, regarding the recomputations in the forward part of the matching, we have to consider only $2 + 4 + 4 = 10$ S-boxes in the rounds 5, 6, and 7. In backward direction, there are $2 + 8 + 30 \cdot 16 + 4 = 494$ S-boxes which have to be recomputed in the rounds 8-40, which sums up to 504 S-boxes in total

8.4 Complexity

There are $48 \cdot 16 = 768$ S-boxes in the 48 rounds of the cipher. Thus, for 2^{16} keys in one set, C_{recomp} is equal to $2^{16} \cdot \frac{504}{768} \approx 2^{15.39}$ full encryptions. In the precomputations step, we require to compute 2^8 times 40 out of 48 rounds, which is equivalent to $2^{7.74}$ 40-round encryptions. The costs to create one biclique are given by 2^9 computations of eight out of 48 rounds, or $2^{6.68}$ encryptions. The full computational complexity can be approximated by

$$C_{full} = 2^{112} \cdot (2^{6.68} + 2^{7.74} + 2^{15.39} + 2^8 + 2^8) \approx 2^{127.42} \text{ encryptions.}$$

encryptions. The attack requires the adversary to store 2^{11} bytes, and, when fixing C_0 over all key sets, to collect 2^8 chosen plaintexts.

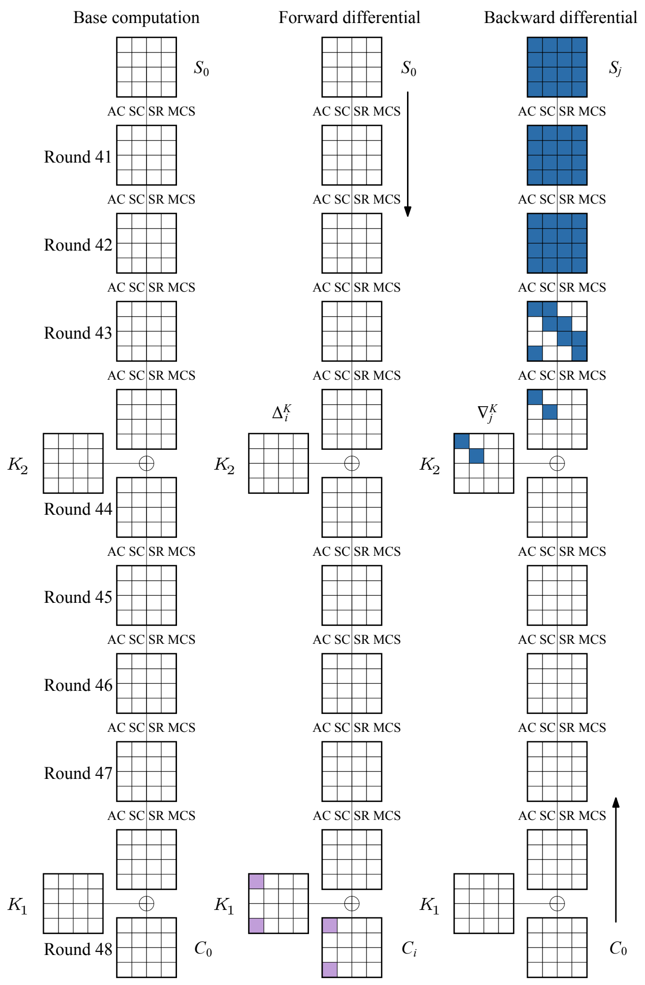


Fig. 15. The biclique on full LED-128 over the rounds 41-48 with $\mathit{\Delta}_i$- and ∇_j-differentials.

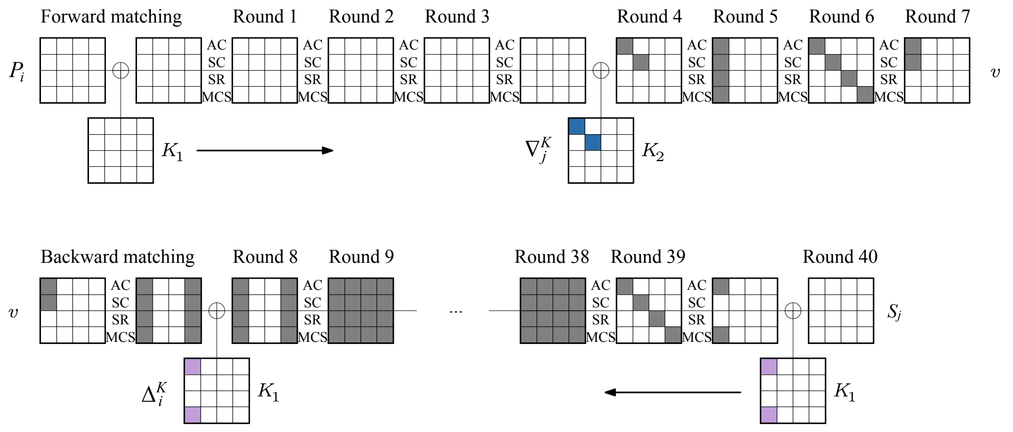


Fig. 16. Recomputations for LED-128 in forward and backward direction.

8.5 Independent-Biclique Attack On 32-Round LED-128

We can mount an attack on a reduced version of LED-128, which covers the first 32 rounds. number of operations, with the same key space partitioning, biclique, and partial matching procedure as before (see Section 8). In contrast to the previous attack, the matching phase requires to recompute 16 full rounds less, and the biclique covers the rounds 25-32. This time, we have to recompute $2+4+4=10$ S-boxes in forward direction; the effort in backward direction becomes $2+8+14\cdot 16+4=238$ S-boxes, which sums up to 248 S-boxes in total.
There are 512 S-boxes in the 32 rounds of the reduced cipher. Hence, C_{recomp} is given by $2^{16}\cdot\frac{248}{512}\approx 2^{14.95}$ encryptions. $C_{precomp}$ represents the effort of computing 2^8 times 24 out of 32 rounds, or $2^{7.58}$ 32-round encryptions. The costs for constructing a biclique, $C_{biclique}$ are given by 2^9 computations of eight out of 32 rounds, or $2^{7.58}$ encryptions. Thus, the total computational complexity of this attack is given by

$$C_{full}=2^{112}\cdot(2^{7.42}+2^{7.58}+2^{14.95}+2^{8}+2^{8})\approx 2^{126.99}\text{ encryptions.}$$

32-round encryptions. Again, the attack requires 2^8 chosen plaintexts and memory to store 2^{11} bytes.

9 KLEIN

9.1 Round Transformation

The structure of KLEIN is a typical substitution-permutation network which combines ideas from the round transformation of the AES with the small 4×4-S-box from PRESENT to have a small implementation footprint. KLEIN has a fixed block length of 64 bits and supports key lengths of 64, 80 and 96 bits. Depending on the key length, KLEIN processes the plaintext in $N_R=12/16/20$ rounds, where each round consists of four operations.

- `AddRoundKey` (`AK`(sk^i)): A 64-bit round key sk^i is XORed with the state.
- `SubNibbles` (`SN`): The nibbles in the state are replaced using a 4×4-S-box.
- `RotateNibbles` (`RN`): The state is rotated by two nibbles to the left.
- `MixNibbles` (`MN`): The state is split into two 32-bit halves, and each half is multiplied with the MDS matrix of the AES in the $GF(2^4)$.

After the final round, a final round key sk^{N_R+1} is XORed with the state to generate the ciphertext.

9.2 Key Schedule

The key schedule of KLEIN expands the secret key to N_R+1 round keys of 64-bits. The first round key sk^1 is initialized with the secret key. A round key sk^{i+1} is then derived from its previous round key sk^i as follows:

1. Divide the subkey sk^i into two halves, named a and b. For KLEIN-64, one obtains $a=sk^i_0,sk^i_1,sk^i_2,sk^i_3$ and $b=sk^i_4,sk^i_5,sk^i_6,sk^i_7$, where sk^i_j denotes the j-th byte.
2. Rotate a and b by one byte to the left to obtain a',b': $a'=sk^i_1,sk^i_2,sk^i_3,sk^i_0$ and $b'=sk^i_5,sk^i_6,sk^i_7,sk^i_4$.
3. XOR b' to a' and swap both halves to obtain a'',b'': $a''=b'$ and $b''=a'\oplus b'$.
4. XOR the round counter i with the third byte of a'', and substitute the second and third byte of b'' using the KLEIN S-box.
5. Output the 64 leftmost bits of sk^i as sk^{i+1}.

Figure 17 illustrates the key schedule of KLEIN-64. For further details on the specification of KLEIN, we refer to the original proposal [11].

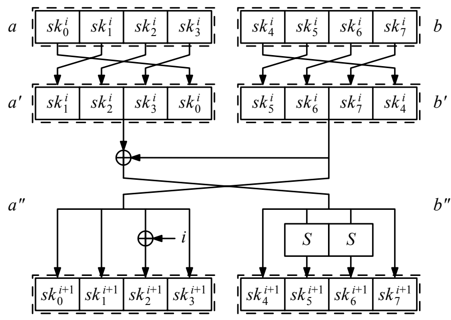


Fig. 17. The key schedule of KLEIN-64.

9.3 Sieve-in-the-Middle

For KLEIN, we can construct a table $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ which stores the transitions $\overrightarrow{v_{i,j}} \leftrightarrow \overleftarrow{v_{i,j}}$ over the operation sequence $E_s = \texttt{SN} \circ \texttt{AK(}sk^i\texttt{)} \circ \texttt{MN} \circ \texttt{RN} \circ \texttt{SN}$. Note, that over this sequence, we can separate the state of KLEIN into two distinct halves of 32 bit, where each half depends only on 32 bit of the state and 32 bits of the round key sk^i. The halves are illustrated by the white and darkened cells in Figure 18. In our concrete attacks on KLEIN, we construct a table for that half that is visualized by the darkened cells. However, to lower the memory complexity, we only consider 24 bits of the key sk^i, i.e., those nibbles in the figure, which are not struck through. The effort for constructing the table $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ is therefore given by 2^{32} computations of the sequence $\texttt{MN} \circ \texttt{RN} \circ \texttt{SN}$, and 2^{56} computations of $\texttt{SN} \circ \texttt{AK(}sk^i\texttt{)}$, both which are negligible in the total effort. We require to store 2^{56} entries under the index $(\overrightarrow{v_{i,j}} \| \overleftarrow{v_{i,j}} \| k_s)$, which means that we require $2^{56} \cdot \frac{32+24+24}{8} < 2^{60}$ bytes.

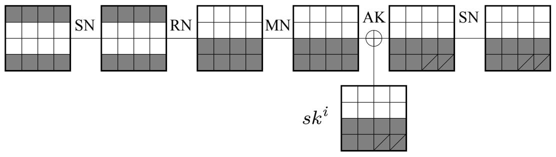
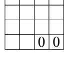
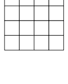
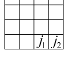


Fig. 18. Sieve over the steps $\texttt{SN} \circ \texttt{AK(}sk^i\texttt{)} \circ \texttt{MN} \circ \texttt{RN} \circ \texttt{SN}$ for KLEIN.

10 Independent-Biclique Attack on Full KLEIN-80

10.1 Key Space Partitioning

The partitioning procedure is very similar to that for KLEIN-80. First, we divide the key space into 2^{64} groups of 2^{16} keys each with respect to the 80-bit key register before the extraction of the second round key sk^2. The base keys $K[0,0]$ are all 20-nibble values with four nibbles fixed to zero, whereas the remaining nibbles running over all other possible values.

$K[0,0](sk^2) =$  $\Delta_i^K(sk^2) =$  $\nabla_j^K(sk^2) =$ 

10.2 3-Round Biclique Of Dimension 8

Since four-round bicliques lead to fully active states at both ends of the differentials, we construct a biclique over three rounds as depicted in Figure 19. From there, one can see that the ∇_j-differentials, the plaintexts P_j are only affected twelve out of 16 nibbles. By fixing the plaintexts P_0 over all bicliques in the attack, the data complexity for the bicliques does not exceed 2^{48} chosen plaintexts.

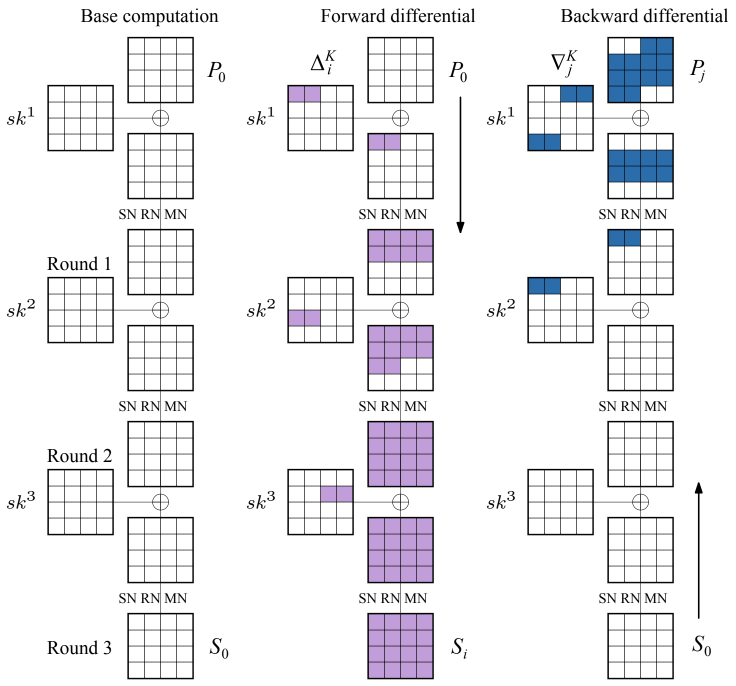


Fig. 19. The biclique over the rounds 1-3 in our attack on full KLEIN-80. The light-blue colored trails indicate bits affected by the differences Δ_i^K, the dark-blue trails bits affected by the differences ∇_j^K.

10.3 Matching Over 13 Rounds

We perform a matching-with-precomputations over the rounds 4-16. We locate the states $\overrightarrow{v_{i,j}}$ after the key addition with sk^8, and the states $\overleftarrow{v_{i,j}}$ before the S-box layer of Round 9. We construct a sieve, as described in Section 9.3, over Round 8 and the key addition and S-box layer of Round 9. The exact matching procedure is illustrated in Figure 20; the darkened cells indicate those nibbles which have to be recomputed in the states. In forward direction, we have to recompute two S-boxes in Round 4, ten in Round 5, 16 in Round 6, and eight in Round 7, which sums up to 36 S-box operations. In backward direction, we have to recompute eight S-boxes in Round 16, and $6 \cdot 16$ S-box operations in the rounds 10-15, which gives 104 S-boxes for the backward part. In addition, we take into account the bytes which have to be recomputed and that serve as input to the S-box during the key schedule. Concerning the differences ∇_j^K, we have to recompute two S-boxes in sk^4, and two S-boxes in sk^5. Concerning the differences Δ_i^K, we need to consider two S-boxes in sk^8, two in sk^9, two in sk^{13}, and another two S-boxes in sk^{14}. All in all, there are 152 S-boxes.

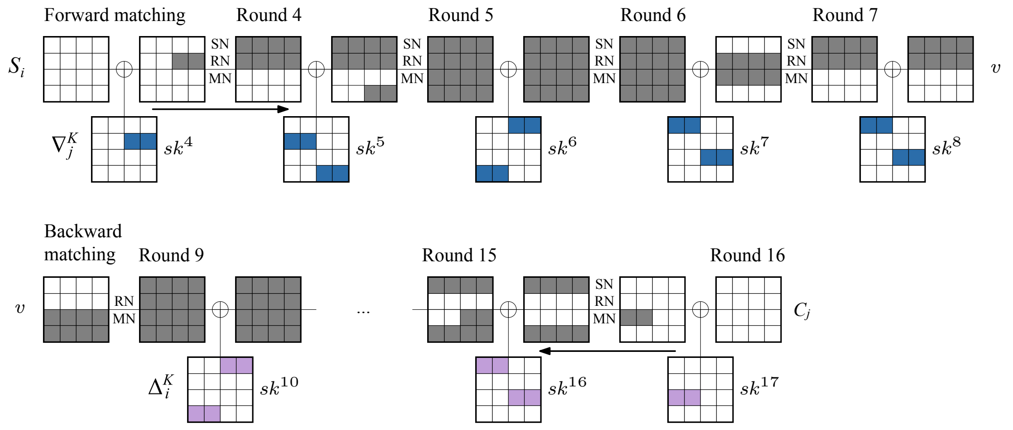


Fig. 20. Matching for the attack on full KLEIN-80 in forward direction from the states S_i to $v_{i,j}$ (top), and in backward direction from the ciphertexts C_j to $v_{i,j}$ (bottom).

10.4 Complexity

In KLEIN-80, there are $16 \cdot 16 = 256$ S-boxes in the round transformation and $16 \cdot 4 = 64$ S-boxes in the key-schedule. C_{recomp} is then equal to $2^{16} \cdot \frac{152}{320} \approx 2^{14.97}$ full encryptions. $C_{biclique}$ is equivalent to $2^{6.58}$ encryptions and $C_{precomp}$ includes 2^8 computations of 13 out of 16 rounds, or $2^{7.7}$ encryptions.
One can see from Figure 20, we know four nibbles of the input to our sieve from the first row of $\overrightarrow{v_{i,j}}$. In addition, we know the full lower half of the state $\overleftarrow{v_{i,j}}$, i.e., all output bits of the sieve, and, given $K[i,j]$, full sk^9. Given our four out of eight nibbles (16 out of 32 bits) of the input $\overrightarrow{v_{i,j}}$, there may be 2^{16} possible output transitions, due to the unknown input bits. The probability, that we find a valid transition $\overrightarrow{v_{i,j}} \leftrightarrow \overleftarrow{v_{i,j}}$ in our precomputed table $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ for a wrong key, is $2^{16} \cdot 2^{-24} = 2^{-8}$, since it must match in the 24 bits of the lower half of $\overleftarrow{v_{i,j}}$, which we have stored as outputs of $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ (cf. Section 9.3). Hence, we can expect to have $C_{falsepos} = 2^{16} \cdot 2^{-8} = 2^8$ false positives for every biclique. The full computational complexity is then given by

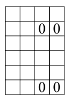
$$C_{full} = 2^{64} \cdot (2^{6.58} + 2^8 + 2^{7.7} + 2^{14.97} + 2^8) \approx 2^{79.00} \text{ encryptions.}$$

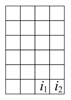
The memory complexity is given by storing 2^8 states and the table $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ or 2^{60} bytes.

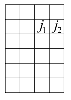
11 Independent-Biclique Attack on Full KLEIN-96

11.1 Key Space Partitioning

The key-partitioning procedure is similar to that in our attack on KLEIN-80. We split the key space into 2^{80} sets of 2^{16} keys each, where the base keys $K[0,0]$ are all 24-nibble values with four nibbles fixed to 0 all other nibbles running over all possible values. The keys in a group $\{K[i,j]\}$ are enumerated by all possible differences $i = (i_1 \| i_2)$ and $j = (j_1 \| j_2)$ with respect to $K[0,0]$.

$K[0,0](sk^2) =$ 

$\Delta_i^K(sk^2) =$ 

$\nabla_j^K(sk^2) =$ 

11.2 3-Round Biclique Of Dimension 8

Again, we construct a biclique over the first three rounds, as depicted in Figure 21. The plaintexts P_j are affected by the key differences in 8 out of 16 nibbles. So, the data complexity does not exceed 2^{32}.

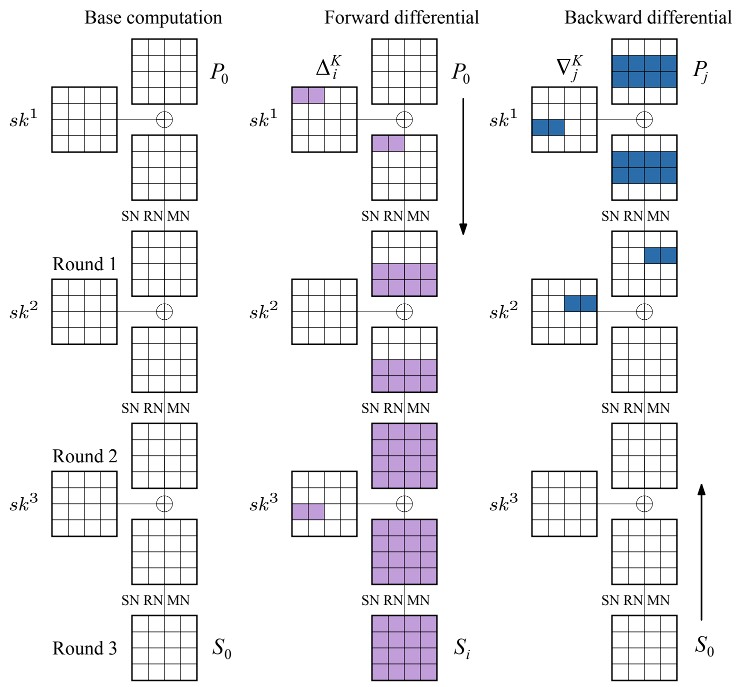


Fig. 21. The biclique over the rounds 1-3 in our attack on full KLEIN-96. The light-blue colored trails indicate bits affected by the differences Δ_i^K, the dark-blue trails bits affected by the differences ∇_j^K.

11.3 Matching Over 17 Rounds

We apply a matching-with-precomputations to the rounds 4-20, where we locate the input states to our sieve $\overrightarrow{v_{i,j}}$ in the states after the key injection with sk^6, and the output states $\overleftarrow{v_{i,j}}$ in the states before the S-box layer of Round 7. The recomputations are illustrated in Figure 22. In the forward part of the round transformation, we have to recompute four S-boxes in Round 4, and eight in Round 5, which yield 12 for this part. In backward direction, we have to consider eight S-boxes in Round 20, and $12 \cdot 16$ in the rounds 8-19, which sums up to 200. In the key schedule, we need to consider two S-boxes each in the computation of sk^4, sk^{11}, sk^{12}, sk^{17}, and sk^{18}. Hence, the recomputation costs sum up to 222.

11.4 Complexity

There are $20 \cdot 16 = 320$ S-boxes in the round transformation and $20 \cdot 4 = 80$ S-boxes in the key-schedule. Thus, C_{recomp} is equal to $2^{16} \cdot \frac{222}{400} \approx 2^{15.15}$ full encryptions. The biclique effort is given by computing 3 out of 20 rounds 2^9 times. The precomputational costs include 2^8 computations of 17 out of 20 rounds, or $2^{7.77}$ encryptions. With the same argumentation as in the attack on KLEIN-80, we can expect $C_{falsepos} = 2^8$ false positives for each biclique. The full computational complexity is then equivalent to

$$C_{full} = 2^{80} \cdot (2^{6.26} + 2^8 + 2^{7.77} + 2^{15.15} + 2^8) \approx 2^{95.18} \text{ encryptions.}$$

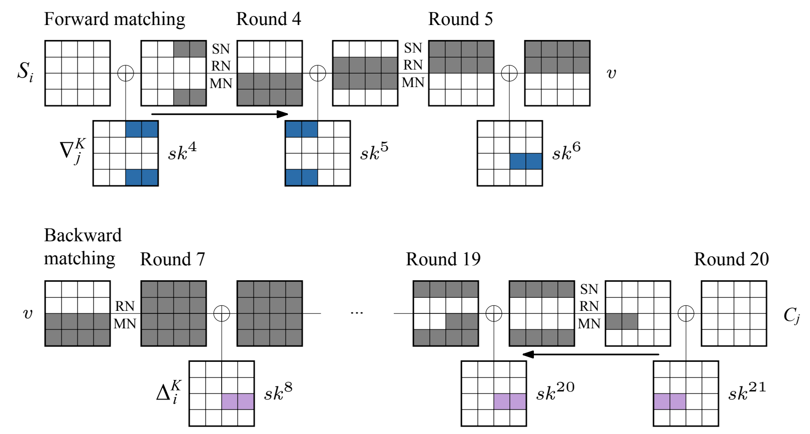


Fig. 22. Matching for the attack on full KLEIN-96 in forward direction from the states S_i to $v_{i,j}$ (top), and in backward direction from the ciphertexts C_j to $v_{i,j}$ (bottom).

The data complexity of in this attack is upper bounded by 2^{32} chosen plaintexts and the memory complexity is given by storing 2^8 states and $T_{\overrightarrow{v_{i,j}},\overleftarrow{v_{i,j}}}$ or 2^{60} bytes.
As we can see from our analysis of full KLEIN-80, an attack on the first 16 of KLEIN-96– including a final wrapping key addition with sk^{17} – using the same biclique and matching procedure as above would yield a time complexity of $2^{95.00}$ encryptions; thus, allowing an advantage factor of two for the adversary.

12 Conclusion

In this paper, we proposed biclique attacks on full and reduced versions of PRESENT and LED, and full KLEIN-80 and KLEIN-96. In our attacks on full PRESENT-80 and PRESENT-128, the time complexities are equivalent to $2^{79.49}$ and $2^{127.32}$ full encryptions and the data complexities are given by 2^{25} and 2^{23} chosen plaintexts, respectively. In our attacks on full LED-64 and LED-128, we obtained computational complexities of $2^{63.58}$ and $2^{127.42}$ encryptions. In these attacks, the adversary has to collect only a number of 2^8 chosen plaintexts. In the attacks on KLEIN-80 and KLEIN-96, the adversary requires $2^{79.00}$ and $2^{95.18}$ encryptions and 2^{48} and 2^{32} chosen plaintexts, respecitively.
Our results on the full-round versions established new lower security margins for these ciphers. Due to the high number of rounds in PRESENT and LED, our analyses of their full versions gain a relatively low advantage for the adversary, i.e., a factor of only 0.42 to 0.58 bits. For KLEIN and PRESENT, we can profit from precomputing a table over one round in the middle. We could achieve the best attacks of the considered ciphers at the time of writing this paper. As a consequence, our work contributes to establish new lower bounds on them. Authors of future lightweight ciphers can learn from our results the impact of combining many rounds decreases the effect of bruteforce-like cryptanalysis effectively.

References

1. Zahra Ahmadian, Mahmoud Salmasizadeh, and Mohammad Reza Aref. Biclique Cryptanalysis of the Full-Round KLEIN Block Cipher. Cryptology ePrint Archive, Report 2013/097, 2013. `http://eprint.iacr.org/`.
2. Martin R. Albrecht and Carlos Cid. Algebraic Techniques in Differential Cryptanalysis. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2009.
3. Jean-Philippe Aumasson, María Naya-Plasencia, and Markku-Juhani O. Saarinen. Practical Attack on 8 Rounds of the Lightweight Block Cipher KLEIN. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *INDOCRYPT*, volume 7107 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 2011.

4. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2011.
5. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
6. Anne Canteaut, Maria Naya-Plasencia, and B. Vayssiere. Sieve-in-the-Middle: Improved MITM Attacks. To be published, January 2013.
7. Shaozhen Chen and Tianmin Xu. Biclique Attack of the Full ARIA-256. *IACR Cryptology ePrint Archive*, 2012:11, 2012.
8. Joo Yeon Cho. Linear Cryptanalysis of Reduced-Round PRESENT. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2010.
9. Mustafa Coban, Ferhat Karakoc, and Özkan Boztacs. Biclique Cryptanalysis of TWINE. Cryptology ePrint Archive, Report 2012/422, 2012. `http://eprint.iacr.org/`.
10. Baudoin Collard and Fran*C*cois-Xavier Standaert. A Statistical Saturation Attack against the Block Cipher PRESENT. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2009.
11. Zheng Gong, Svetla Nikova, and Yee Wei Law. KLEIN: A New Family of Lightweight Block Ciphers. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
12. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
13. Deukjo Hong, Bonwook Koo, and Daesung Kwon. Biclique Attack on the Full HIGHT. In Howon Kim, editor, *ICISC*, volume 7259 of *Lecture Notes in Computer Science*, pages 365–374. Springer, 2011.
14. Takanori Isobe and Kyoji Shibutani. Security Analysis of the Lightweight Block Ciphers XTEA, LED and Piccolo. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP*, volume 7372 of *Lecture Notes in Computer Science*, pages 71–86. Springer, 2012.
15. Kitae Jeong, HyungChul Kang, Changhoon Lee, Jaechul Sung, and Seokhie Hong. Biclique cryptanalysis of lightweight block ciphers present, piccolo and led. Cryptology ePrint Archive, Report 2012/621, 2012. `http://eprint.iacr.org/`.
16. Dmitry Khovratovich, Gaëtan Leurent, and Christian Rechberger. Narrow-Bicliques: Cryptanalysis of Full IDEA. In *EUROCRYPT*, pages 392–410, 2012.
17. Dmitry Khovratovich and Christian Rechberger. A Splice-and-Cut Cryptanalysis of the AES. *IACR Cryptology ePrint Archive*, 2011:274, 2011. `http://eprint.iacr.org/2011/274`.
18. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. Cryptology ePrint Archive, Report 2011/286, 2011. `http://eprint.iacr.org/`.
19. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 244–263. Springer, 2012.
20. Hamid Mala. Biclique Cryptanalysis of the Block Cipher SQUARE. Cryptology ePrint Archive, Report 2011/500, 2011. `http://eprint.iacr.org/`.
21. Florian Mendel, Vincent Rijmen, Deniz Toz, and Kerem Varici. Differential Analysis of the LED Block Cipher. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 190–207. Springer, 2012.
22. Ivica Nikolic, Lei Wang, and Shuang Wu. Cryptanalysis of Round-Reduced LED. In Shiho Moriai, editor, *Fast Software Encryption, 20th International Workshop – FSE 2013*, Lecture Notes in Computer Science. Springer, 2013.
23. Kenji Ohkuma. Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2009.
24. M. Shakiba, M. Dakhilalian, and H. Mala. Non-isomorphic Biclique Cryptanalysis and Its Application to Full-Round mCrypton. Cryptology ePrint Archive, Report 2013/141, 2013. `http://eprint.iacr.org/`.
25. Meiqin Wang. Differential Cryptanalysis of Reduced-Round PRESENT. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 2008.

26. Yanfeng Wang, Wenling Wu, and Xiaoli Yu. Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher. In Mark Dermot Ryan, Ben Smyth, and Guilin Wang, editors, *ISPEC*, volume 7232 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2012.
27. Yanfeng Wang, Wenling Wu, Xiaoli Yu, and Lei Zhang. Security on LBlock against Biclique Cryptanalysis. In *WISA2012*, Lecture Notes in Computer Science (LNCS), 8 2012.
28. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved Meet-in-the-Middle Cryptanalysis of KTANTAN. Cryptology ePrint Archive, Report 2011/201, 2011.
29. Xiaoli Yu, Wenling Wu, Yanjun Li, and Lei Zhang. Cryptanalysis of Reduced-Round KLEIN Block Cipher. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 7537 of *Lecture Notes in Computer Science*, pages 237–250. Springer, 2011.

A Round-Key Dependencies Of PRESENT

Table 2 summarizes the secret-key bits on which the round keys RK^i depend.

Secret-key bits in round keys of PRESENT-80									Round key	Secret-key bits in round keys of PRESENT-128								
79	78	77	76	…	19	18	17	16	RK^1	127	126	125	124	…	67	66	65	64
18	17	16	15	…	38	37	36	35	RK^2	66	65	64	63	…	6	5	4	3
37	36	35	34	…	57	56	55	54	RK^3	5	4	3	2	…	73	72	71	70
56	55	54	53	…	76	75	74	73	RK^4	72	71	70	69	…	12	11	10	9
75	74	73	72	…	15	14	13	12	RK^5	11	10	9	8	…	79	78	77	76
14	13	12	11	…	34	33	32	31	RK^6	78	77	76	75	…	18	17	16	15
33	32	31	30	…	53	52	51	50	RK^7	17	16	15	14	…	85	84	83	82
52	51	50	49	…	72	71	70	69	RK^8	84	83	82	81	…	24	23	22	21
71	70	69	68	…	11	10	9	8	RK^9	23	22	21	20	…	91	90	89	88
10	9	8	7	…	30	29	28	27	RK^{10}	90	89	88	87	…	30	29	28	27
29	28	27	26	…	49	48	47	46	RK^{11}	29	28	27	26	…	97	96	95	94
48	47	46	45	…	68	67	66	65	RK^{12}	96	95	94	93	…	36	35	34	33
67	66	65	64	…	7	6	5	4	RK^{13}	35	34	33	32	…	103	102	101	100
6	5	4	3	…	26	25	24	23	RK^{14}	102	101	100	99	…	42	41	40	39
25	24	23	22	…	45	44	43	42	RK^{15}	41	40	39	38	…	109	108	107	106
44	43	42	41	…	64	63	62	61	RK^{16}	108	107	106	105	…	48	47	46	45
63	62	61	60	…	3	2	1	0	RK^{17}	47	46	45	44	…	115	114	113	112
2	1	0	79	…	22	21	20	19	RK^{18}	114	113	112	111	…	54	53	52	51
21	20	19	18	…	41	40	39	38	RK^{19}	53	52	51	50	…	121	120	119	118
40	39	38	37	…	60	59	58	57	RK^{20}	120	119	118	117	…	60	59	58	57
59	58	57	56	…	79	78	77	76	RK^{21}	59	58	57	56	…	127	126	125	124
78	77	76	75	…	18	17	16	15	RK^{22}	126	125	124	123	…	66	65	64	63
17	16	15	14	…	37	36	35	34	RK^{23}	65	64	63	62	…	5	4	3	2
36	35	34	33	…	56	55	54	53	RK^{24}	4	3	2	1	…	72	71	70	69
55	54	53	52	…	75	74	73	72	RK^{25}	71	70	69	68	…	11	10	9	8
74	73	72	71	…	14	13	12	11	RK^{26}	10	9	8	7	…	78	77	76	75
13	12	11	10	…	33	32	31	30	RK^{27}	77	76	75	74	…	17	16	15	14
32	31	30	29	…	52	51	50	49	RK^{28}	16	15	14	13	…	84	83	82	81
51	50	49	48	…	71	70	69	68	RK^{29}	83	82	81	80	…	23	22	21	20
70	69	68	67	…	10	9	8	7	RK^{30}	22	21	20	19	…	90	89	88	87
9	8	7	6	…	29	28	27	26	RK^{31}	89	88	87	86	…	29	28	27	26
28	27	26	25	…	48	47	46	45	RK^{32}	28	27	26	25	…	96	95	94	93

Table 2. Secret-key bits k_j, which affect the individual round keys of PRESENT.