Security Evaluation of Block Ciphers AES, Camellia, CLEFIA and SC2000 using Two New Techniques: Biclique Attacks and Zero-Correlation Linear Attacks

Final Report
Version 1.0, November 2012

Andrey Bogdanov

Abstract. CRYPTREC is planning to update the current e-Government Recommended Ciphers List published in 2003 by March 2013. The candidate algorithms for 128-bit block ciphers are AES, Camellia, CIPHERUNICORN-A, Hierocrypt-3, SC2000, and CLEFIA. For this purpose, CRYPTREC aims to have the security of these algorithms reevaluated using some novel techniques of cryptanalysis.
The document deals with two new techniques of symmetric-key cryptanalysis: biclique-based MITM cryptanalysis and zero-correlation linear cryptanalysis that have recently turned out to be capable of attacking stronger ciphers than previously believed. We concentrate on block ciphers AES, Camellia, CLEFIA and SC2000 here. More specifically, we evaluate AES, Camellia, CLEFIA and SC2000 with respect to biclique-based MITM cryptanalysis as well as Camellia and CLEFIA with respect to zero-correlation linear cryptanalysis.
We conclude that all for all full ciphers in question AES, Camellia, CLEFIA and SC2000 applying exhaustive MITM attacks can reduce the complexity of key recovery with respect to brute force. Zero-correlation linear cryptanalysis turns out applicable to round-reduced variants of Camellia and CLEFIA.

Keywords: AES, Camellia, CLEFIA, SC2000, biclique cryptanalysis, zero-correlation linear cryptanalysis

Acknowledgements. This work was supported in part by the Cryptography Research and Evaluation Committees (CRYPTREC). I would like to thank Meiqin Wang for her kind permission to include some of the results on zero correlation from our unpublished joint work into this report.

1 Short Specifications of Ciphers

1.1 AES-128

For the sake of clarity, we will be trying to reuse as much notation as possible from [13] – the work introducing biclique cryptanalysis for block ciphers and applying it to AES. This also applies to the notations regarding the description of AES-128.

AES-128 is a block cipher with 128-bit internal state and 128-bit key K. The internal state is represented by a 4×4 byte matrix, and the key is represented by a 4×4 matrix. The plaintext is xored with the key, and then undergoes a sequence of 10 rounds. Each round consists of four transformations: nonlinear bytewise SubBytes, the byte permutation ShiftRows, linear transformation MixColumns, and the addition with a subkey AddRoundKey. MixColumns is omitted in the last round.

SubBytes is a nonlinear transformation operating on 8-bit S-boxes. The operation ShiftRows rotates bytes in row r by r positions to the left. The MixColumns is a linear transformation with branch number 5, i.e. in the column equation $(y_0, y_1, y_2, y_3) = MC(x_0, x_1, x_2, x_3)$ only 5 and more variables can be non-zero.

We address two internal states in each round as follows: #1 is the state before SubBytes in round 1, #2 is the state after MixColumns in round 1, #3 is the state before SubBytes in round 2, ..., #19 is the state before SubBytes in round 10, #20 is the state after ShiftRows in round 10.

The key K is expanded to a sequence of keys $K^0, K^1, K^2, \ldots, K^{10}$, which form a 4×60 byte array. Then the 128-bit subkeys $\$0, \$1, \$2, \ldots, \$14$ come out of the sliding window with a 4-column step. The keys in the expanded key are formed as follows. First, $K^0 = K$. Then, column 0 of K^r is the column 0 of K^{r-1} xored with the nonlinear function (SK) of the last column of K^{r-1}. Subsequently, column i of K^r is the xor of column $i-1$ of K^r and of column i of K^{r-1}.

1.2 Camellia

Camellia [1] is a block cipher designed by researchers from Mitsubishi and NTT. It is part of the ISO/IEC encryption mechanisms as well as NESSIE and CRYPTREC portfolios. The block size of Camellia is 128 bits. The key size of Camellia is either 128, 192, or 256 bits. Thus, the interfaces are compatible to those of AES.

Camellia is a balanced Feistel network of 24 rounds organized as 3 blocks of 6 rounds for Camellia-128 and 4 blocks of 6 rounds for Camellia-192 and Camellia-256. The 6-round blocks are separated by the applications of FL and FL^{-1}

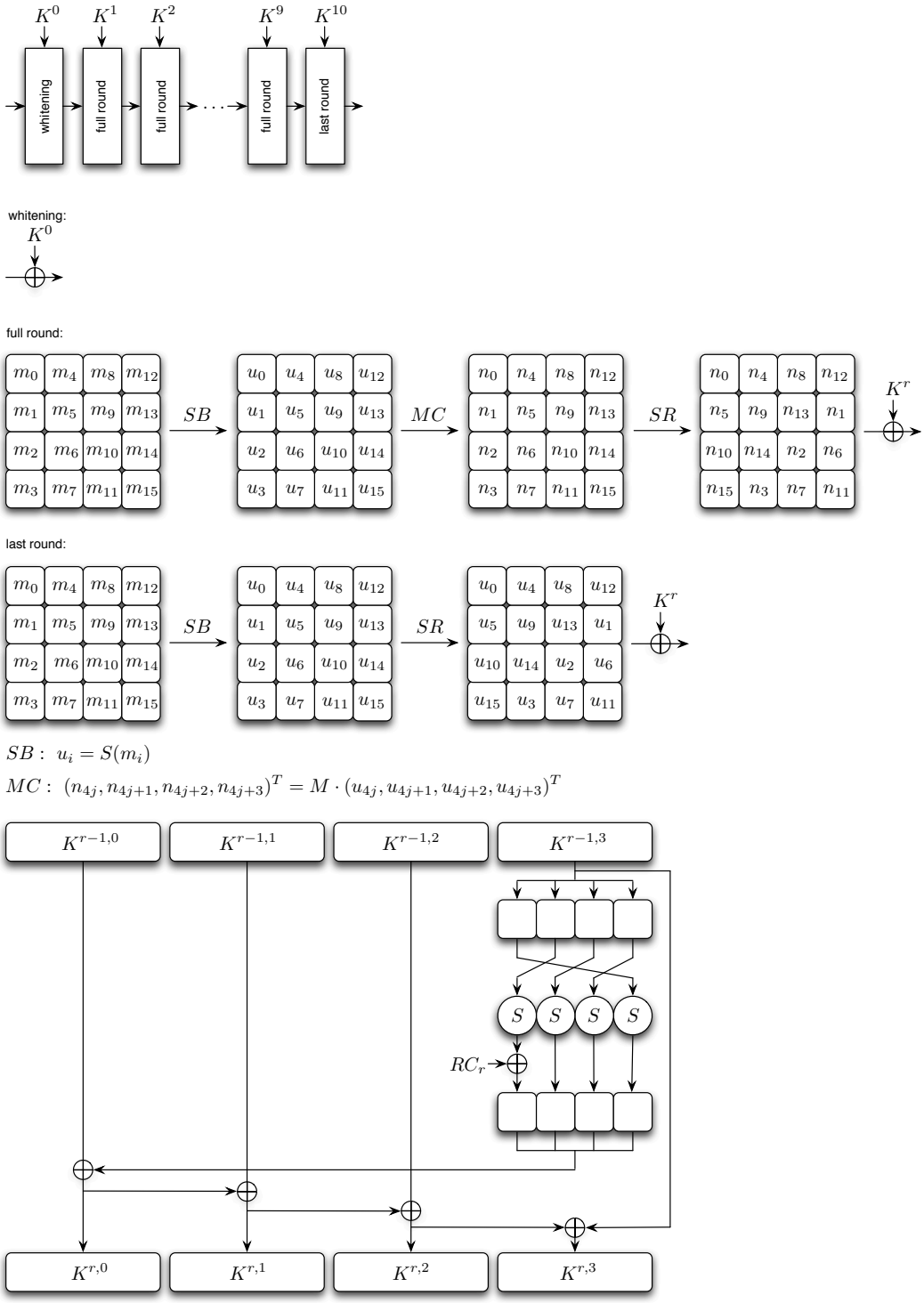

K^0 K^1 K^2 K^9 K^{10}
whitening
full round
full round
full round
last round
whitening:
K^0
full round:
SB MC SR K^r
last round:
SB SR K^r
$SB: \ u_i = S(m_i)$
$MC: \ (n_{4j}, n_{4j+1}, n_{4j+2}, n_{4j+3})^T = M \cdot (u_{4j}, u_{4j+1}, u_{4j+2}, u_{4j+3})^T$
$K^{r-1,0}$ $K^{r-1,1}$ $K^{r-1,2}$ $K^{r-1,3}$
S S S S
RC_r
$K^{r,0}$ $K^{r,1}$ $K^{r,2}$ $K^{r,3}$

Fig. 1. AES-128

functions on the left-hand and right-hand halves of the Feistel state, correspondingly. Before the first 6-round block and after the last 6-round block whitening keys are added. The F-function maps 64 bits to 64 bits. The input is divided into 8 bytes. Each byte is first added with the corresponding byte of the expanded key and then is passed through an 8-bit S-box. This is followed by a simple diffusion layer which can be described as a multiplication by a binary matrix over the field of 256 elements. The FL/FL^{-1} functions linear in the data input (consisting of shifts and XORs) but are nonlinear in the key input, it being introduced per bitwise AND and OR. See Figure 2.

1.3 CLEFIA

CLEFIA [69] is a 128-bit block cipher with its key length being 128,192 and 256 bits, which is compatible to AES. It employs a generalized Feistel structure with four data lines, and the width of each data line is 32 bits. Additionally, there are key whitening parts at the beginning and the end of the cipher. The numbers of rounds of CLEFIA are 18, 22 and 26 for 128-bit, 192-bit and 256-bit keys, respectively.

The F-function inside the GFN structure bears similarities to the buidling blocks of AES: an addition of 4 key bytes (RK_i) to the input is followed 4 8-bit S-boxes and a matrix-vector multiplication using a 4x4 MDS matrix (branch number 5) over bytes. Before the first round and after the last round, prewhitening and postwhitening is applied. See Figure 3 for an illustration.

1.4 SC2000

SC2000 [68] is a block cipher designed by Fujitsu. It was considered by the NESSIE project, which did not result in a selection though. SC2000 is part of the CRYPTREC portfolio. The key size of SC2000 is 128, 192, or 256 bits and the block size of SC2000 is 128 bits. Thus, its interfaces are compatible to those of AES.

The round transform can be outlined in the following way. In a round, addition with subkey (I function) is followed by a layer of 4-bit S-boxes (B function) and by another addition with subkey (I function again). Then a round of balanced Feistel-type transform (R function) is applied with an unkeyed F-function consisting of a layer of 5-bit and 6-bit S-boxes followed by a linear transform (superposition of a M and L), the swap of state halves (cross connection $\times$) and by the Feistel-type transform of R function again. The number of rounds is either 6.5 (for SC2000-128) or 7.5 (for SC2000-192 and SC2000-256). Where the last half round consists of functions I, B and I only, omitting the R function.

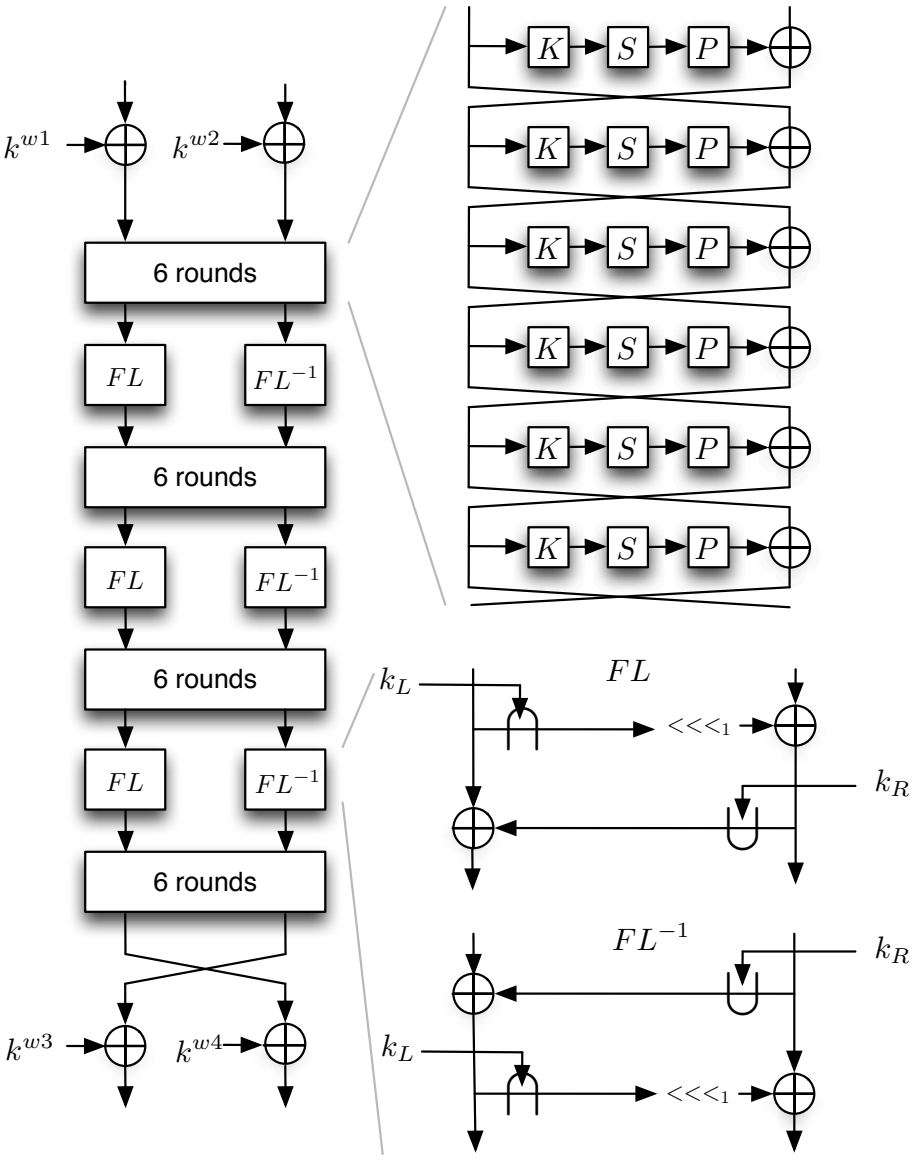


Fig. 2. Camellia-192/256

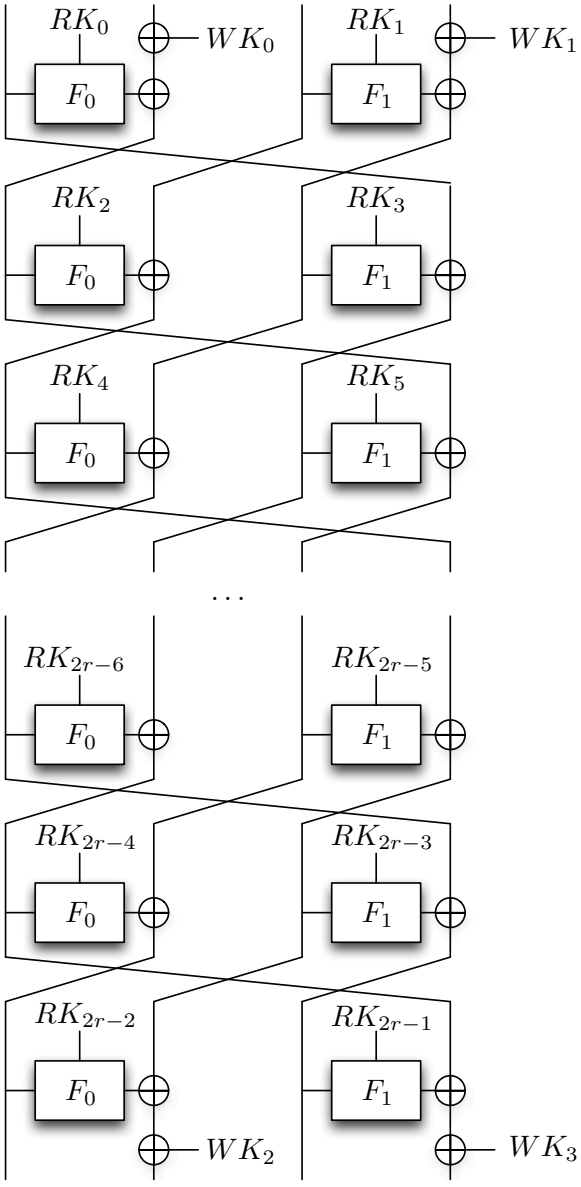

RK_0
WK_0
RK_1
WK_1
F_0
F_1
RK_2
RK_3
RK_4
RK_5
...
RK_{2r-6}
RK_{2r-5}
RK_{2r-4}
RK_{2r-3}
RK_{2r-2}
RK_{2r-1}
WK_2
WK_3

Fig. 3. CLEFIA

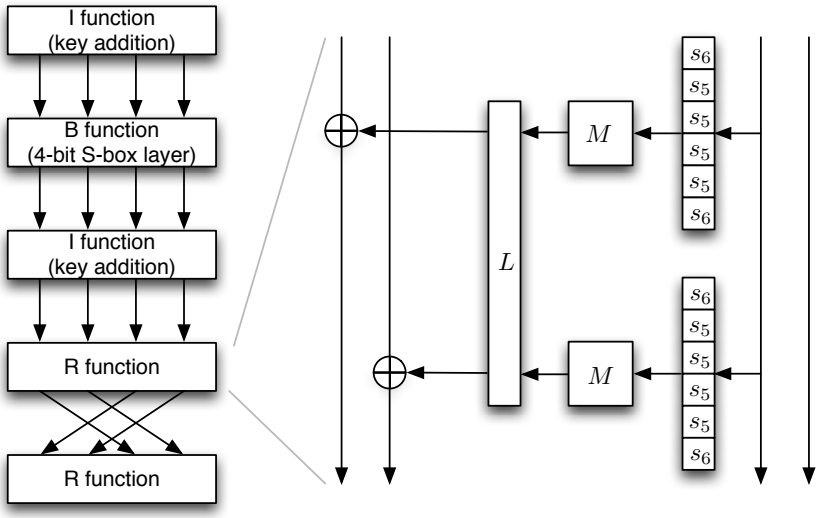


Fig. 4. SC2000

2 Basic Techniques

Block ciphers are essential to cryptography, both theoretically and practically. Most bulk encryption and data authentication performed in the field relies on an efficient block cipher, which is employed after the key has been negotiated in a security protocol. The introduction of cryptanalytic techniques that are based on new principles is a rather rare event in the area of block ciphers. Recently, two new techniques have been proposed for block ciphers: bicliques and zero correlation. We start with a brief introduction in the context of these techniques.

2.1 Bicliques

Today, the most frequently used block cipher is AES (Advanced Encryption Standard) [29] — the current U.S. encryption standard selected by NIST in an open competition. AES is the only publicly known cipher that is NSA-approved for protecting secret and top secret government information in the U.S. Recently, the theoretical security of AES has been challenged by biclique cryptanalysis [13]. Biclique key recovery is based on the meet-in-the-middle approach at its core but borrows an important twist – *initial structures* – from the domain of hash function cryptanalysis. Namely, biclique cryptanalysis uses the fact that, for some ciphers such as AES, the adversary can efficiently prepare structures of internal states

that cover many keys. The work [13] scrutinizes the notion of initial structures for block ciphers and formalizes it to *bicliques* (complete bipartite graphs) which are especially efficient to construct. As a result, the authors of [13] propose key recovery on all three variants of the full AES with computational complexities below brute force, though rather high data complexities. It works in the standard single secret key model. Biclique cryptanalysis has been successfully applied to AES resulting in key recovery faster than brute force. For AES-128, the best attack so far has computational complexity $2^{126.16}$ and data complexity 2^{88}.

Biclique key recovery may be considered as an advance in the field of symmetric-key cryptography but it has been prepared by a considerable number of works in the area of meet-in-the-middle (MITM) attacks on block ciphers [15,22,23,34,39] and hash function cryptanalysis [2, 3, 38] including the introduction of initial structures [65] and bicliques for preimage search in hash functions [45]. Since the introduction of bicliques, an entire line of research emerged aiming to apply the technique to various block ciphers [26, 40, 44, 59, 75].

Here we provide a brief overview of the existing biclique techniques as applied to AES.

Biclique-based preimage finding for hash was proposed in [45] by scrutinizing the concept of initial structures introduced in [65]. Biclique cryptanalysis for ciphers was proposed by [13] in two different paradigms: independent-biclique and long-biclique approaches. It is the independent-biclique technique that has resulted in key recovery for full AES. For the purposes of this report, we will be focusing on independent bicliques. Correspondingly, for simplicity of presentation, the description of biclique techniques in the sequel is explicitly tailored to independent-bicliques for AES-128. For a general introduction into bicliques, we refer to [13].

In the key recovery procedure of independent-biclique cryptanalysis for AES-128, the entire space of 2^{128} keys is divided into non-overlapping groups of keys. In each group of keys, every key is tested using MITM techniques faster than with one AES computation. This accounts for the computational advantage of the technique over brute force.

Balanced bicliques. Each group of keys has the structure of a *biclique* – a complete bipartite graph. A biclique consists of two sets S_x and S_y of intermediate cipher states (two disjoint sets of vertices of the biclique) and a set of keys $\mathcal{K}$ which maps each element in one set of states to each element in the other one (edges of the biclique).

In the standard independent-biclique cryptanalysis, the biclique is defined such that each set of states has $|S_x| = |S_y| = 2^d$ elements and the set of keys

contains $|\mathcal{K}| = 2^{2d}$ keys, which actually corresponds to a *balanced biclique* – a biclique whose vertex sets have equal cardinalities. The structure of the (balanced) biclique is shown in Figure 8. To fix the notation, we denote $S_x = \{x_j\}$, $S_y = \{y_i\}$, and $\mathcal{K} = \{K[i,j]\}$ with $i,j \in \{0, 2^d - 1\}$. d is called the biclique *dimension*.

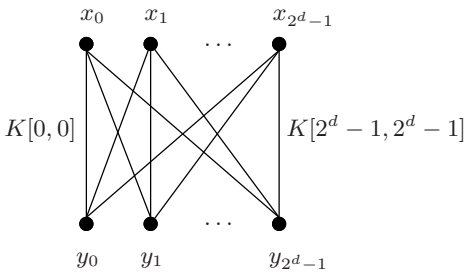


Fig. 5. Balanced biclique of dimension d

Bicliques via independent differentials. So far the most efficient way of building bicliques for AES is from independent related-key differentials. Consider two families of related-key differentials over the rounds covered by biclique (from values x_j to y_i): $2^d - 1$ distinct Δ-*differentials*

$$(0, \Delta_i^K) \longmapsto \Delta_i$$

with input state difference 0, input key difference Δ_i^K and output state difference Δ_i as well as $2^d - 1$ distinct ∇-*differentials*

$$(\nabla_j, \nabla_j^K) \longmapsto 0$$

with input state difference ∇_j, input key difference ∇_j^K and output state difference 0. Now we assume that the Δ- and ∇-differentials do not share any active nonlinear components (S-boxes in the case for AES). Then, it has been shown in [13] that if input x_0, output y_0 and key $K[0,0]$ conforms to both Δ- and ∇-differentials, then the values

$$\begin{aligned} x_j &= x_0 \oplus \nabla_j, \\ y_i &= y_0 \oplus \Delta_i, \\ K[i,j] &= K[0,0] \oplus \nabla_j^K \oplus \Delta_i^K \end{aligned}$$

form a balanced biclique of dimension d, with $\Delta_0 = \nabla_0 = \Delta_0^K = \nabla_0^K = 0$. The computation over the biclique rounds from input x_0 to output y_0 with key $K[0,0]$ is called the *base computation*. $K[0,0]$ is called the *base key*.

For AES-128, the work [13] provides an 8-dimensional biclique over 2.5 last rounds. That is, the vertex set S_y of the biclique coincides with the set of ciphertexts in the given key group. This biclique is illustrated in Figure 6.

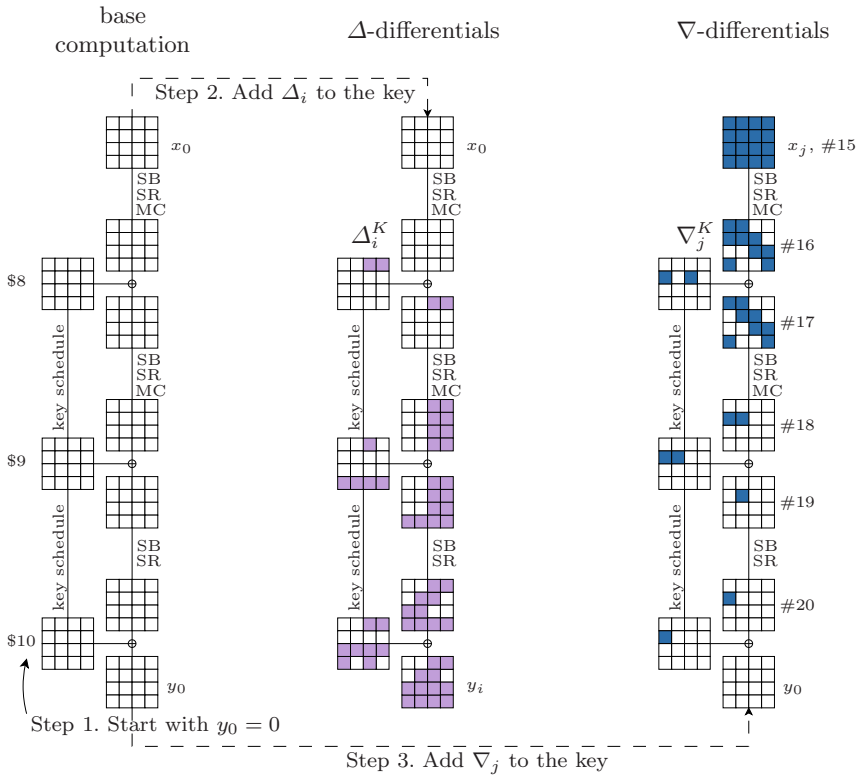


Fig. 6. Biclique over 3 last rounds of AES-128 from independent differentials [13]

Key recovery. For AES-128, the entire space of 2^{128} keys is divided into 2^{112} non-overlapping groups (bicliques) of $2^{2d} = 2^{16}$ keys each. In each biclique, the base key is fixed.

Now a meet-in-the-middle key recovery with partial matching (in one intermediate state byte) and splice-and-cut technique (going over the decryption oracle from the generated ciphertexts to the corresponding plaintexts) is applied. For

each combination of x_j and y_i (corresponding to $K[i,j]$), it is tested if there is a match.

For each biclique, the computational complexity consists of the complexity $C_{precomp}$ of preparing Δ- and ∇-propagations (including preparing the biclique), the complexity C_{recomp} of recomputing states for each key $K[i,j]$, and the complexity $C_{falsepos}$ of checking the key candidates surviving the partial matching:

$$C_{full} = 2^{128-2d}(C_{precomp} + C_{recomp} + C_{falsepos}).$$

The computational complexity is dominated by C_{recomp}. The data complexity is determined by the number of state differences Δ_j in all key groups, since the ciphertext of the base computation remains the same in all key groups.

As regards C_{recomp}, most computational advantage over brute force originates from the fact that the adversary saves the computation of the 3 (strictly speaking, only 2.5 rounds, since MixColumns is omitted in the final round) last rounds of AES-128. In fact, using about 2^8 partial evaluations of the cipher and some minor storage for Δ- and ∇-trails, one covers 2^{16} combinations of x_j and y_i (keys). Another part of computational advantage comes from the partial matching procedure that allows one to save almost 1.5 more rounds of computations. Finally, the intermediate states x_j do not diffuse too fast under $K[i,j]$ in the backward direction. Also the plaintexts obtained from ciphertext y_i using the decryption oracle with the right key $E^{-1}(y_i)$, do not diffuse immediately under $K[i,j]$. This allows the adversary to save more than one round of computations on top of that.

Combining all this yields a computational complexity of $2^{126.16}$ and a data complexity of 2^{88} chosen ciphertexts. Since the adversary cannot reject the right key, the success probability is 1.

2.2 Zero Correlation

For the security evaluation of block ciphers, numerous techniques have been proposed over the last decades. However, although various attacks, in particular differential cryptanalysis and linear cryptanalysis and their variants, have been extensively studied, new interesting results are still being discovered. Among others, the fact that *zero-correlation linear cryptanalysis* [18, 19] has been proposed only very recently – while impossible differential cryptanalysis [7, 20] (its counterpart in the domain of differential cryptanalysis) has been known for about 15 years now – speaks for itself.

Zero-correlation cryptanalysis [18] is a novel promising attack technique for block ciphers. The distinguishing property used in zero-correlation cryptanalysis is the existence of *zero-correlation linear approximations* over (a part of) the

cipher. Those are linear approximations that hold true with a probability p of exactly $1/2$, that is, strictly unbiased approximations having a *correlation* $c = 2p - 1$ equal to 0. This gives the adversary the possibility to gain information from perfectly uncorrelated events.

We use the state-of-the-art techniques [**?**, 19] of zero correlation to decrease the data complexity, in our cryptanalysis of Camellia, CLEFIA, that allow one to take advantage of a maximum number of zero-correlation linear approximations available without relying unnecessary assumptions.

Linear approximations with correlation zero Zero correlation linear cryptanalysis has been introduced in [16]. Below we briefly review its basic ideas and methods.

Consider an n-bit block cipher f_K with key K. Let P denote a plaintext which is mapped to ciphertext C under key K, $C = f_K(P)$. If Γ_P and Γ_C are nonzero plaintext and ciphertext linear masks of n bit each, we denote by $\Gamma_P \to \Gamma_C$ the linear approximation

$$\Gamma_P^T P \oplus \Gamma_C^T C = 0.$$

Here, $\Gamma_A^T A$ denotes the multiplication of the transposed bit vector Γ_A (linear mask for A) by a column bit vector A over $\mathbb{F}_2$. The linear approximation $\Gamma_P \to \Gamma_C$ has probability

$$p_{\Gamma_P,\Gamma_C} = \Pr_{P \in \mathbb{F}_2^n}\{\Gamma_P^T P \oplus \Gamma_C^T C = 0\}. \tag{1}$$

The value

$$c_{\Gamma_P,\Gamma_C} = 2p_{\Gamma_P,\Gamma_C} - 1 \tag{2}$$

is called the *correlation (or bias)* of linear approximation $\Gamma_P \to \Gamma_C$. Note that $p_{\Gamma_P,\Gamma_C} = 1/2$ is equivalent to *zero correlation* $c_{\Gamma_P,\Gamma_C} = 0$:

$$p_{\Gamma_P,\Gamma_C} = \Pr_{P \in \mathbb{F}_2^n}\{\Gamma_P^T P \oplus \Gamma_C^T C = 0\} = 1/2. \tag{3}$$

In fact, for a randomly drawn permutation of sufficiently large bit size n, zero is the most frequent single value of correlation for a nontrivial linear approximation. Correlation goes to small values for increasing n, the probability to get exactly zero decreases as a function of n though. More precisely, the probability of the linear approximation $\Gamma_P \to \Gamma_C$ with $\Gamma_P, \Gamma_C \neq 0$ to have zero correlation can be approximated by

$$\frac{1}{\sqrt{2\pi}} 2^{\frac{4-n}{2}}. \tag{4}$$

Two examples Given a randomly chosen permutation, however, it is hard to tell a priori which of its nontrivial linear approximations in particular has zero correlation. At the same time, it is often possible to identify groups of zero correlation linear approximations for a block cipher f_K once it has compact description with a distinct structure. Moreover, in many interesting cases, these linear approximations will have zero correlation *for any key* K. Here are two examples:

- **AES:** The data transform of AES has a set of zero correlation linear approximations over 4 rounds (3 full rounds appended by 1 incomplete rounds with MixColumns omitted). If Γ and Γ' are 4-byte column linear masks with exactly one nonzero byte, then each of the linear approximations $(\Gamma, 0, 0, 0) \rightarrow (\Gamma', 0, 0, 0)$ over 4 AES rounds has zero correlation.
- **CLEFIA-type GFNs:** CLEFIA-type generalized Feistel networks (also known as type-II GFNs with 4 lines) have zero correlation linear approximations over 9 rounds, if the underlying F-functions of the Feistel construction are invertible. For $a \neq 0$, the linear approximations $(a, 0, 0, 0) \rightarrow (0, 0, 0, a)$ and $(0, 0, a, 0) \rightarrow (0, a, 0, 0)$ over 9 rounds have zero correlation.

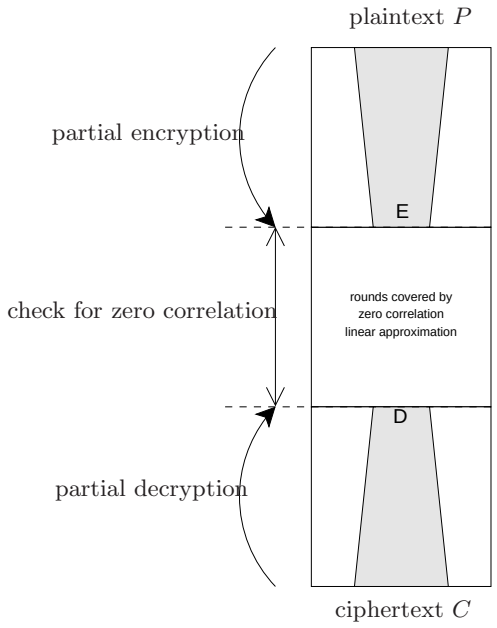


Fig. 7. High-level view of key recovery in zero correlation linear cryptanalysis

Key recovery with zero correlation linear approximations Based on linear approximations of correlation zero, a technique similar to Matsui's Algorithm 2 can be used for key recovery. Let the adversary have N known plaintext-ciphertexts and ℓ zero correlation linear approximations $\{\Gamma_E \to \Gamma_D\}$ for a part of the cipher, with $\ell = |\{\Gamma_E \to \Gamma_D\}|$. The linear approximations $\{\Gamma_E \to \Gamma_D\}$ are placed in the middle of the attacked cipher. Let E and D be the partial intermediate states of the data transform at the boundaries of the linear approximations.

Then the key can be recovered using the following approach (see also Figure 7):

1. Guess the bits of the key needed to compute E and D. For each guess:
   (a) Partially encrypt the plaintexts and partially decrypt the ciphertexts up to the boundaries of the zero correlation linear approximation $\Gamma_E \to \Gamma_D$.
   (b) Estimate the correlations $\{\hat{c}_{\Gamma_E,\Gamma_D}\}$ of all linear approximations in $\{\Gamma_E \to \Gamma_D\}$ for the key guess using the partially encrypted and decrypted values E and D by counting how many times $\Gamma_E^T E \oplus \Gamma_D^T D$ is zero over N input/output pairs, see (1) and (2).
   (c) Perform a test on the estimated correlations $\{\hat{c}_{\Gamma_E,\Gamma_D}\}$ for $\{\Gamma_E \to \Gamma_D\}$ to tell of the estimated values of $\{\hat{c}_{\Gamma_E,\Gamma_D}\}$ are compatible with the hypothesis that all of the actual values of $\{c_{\Gamma_E,\Gamma_D}\}$ are zero.
2. Test the surviving key candidates against a necessary number of plaintext-ciphertext pairs according to the unicity distance for the attacked cipher.

Step 1(c) of the technique above relies on an efficient test distinguishing between the hypothesis that $\{c_{\Gamma_E,\Gamma_D}\}$ are all zero and the alternative hypothesis. The work [16] requires the exact evaluation of the correlation value (defined by the probability of a linear approximation) and the data complexity is restricted to $N = 2^n$ in [16]. Thus, a small number ℓ of linear approximations is usually enough in [16] and $\hat{c}_{\Gamma_E,\Gamma_D} = c_{\Gamma_E,\Gamma_D}$, though the data complexity of the full codebook is too restrictive.

For most ciphers (including the examples of Subsection 2.2), however, a large number ℓ of zero correlation linear approximations is available. This freedom is not used in [16]. At the same time, it has been shown in the experimental work [27] that any value of correlation can be used for key recovery in a linear attack with reduced data complexity, once enough linear approximations are available. Despite its convincing experimental evidence, [27] gives no theoretical data complexity estimations and does not provide any ways of constructing linear approximations with certain properties.

3 New Techniques

3.1 Stars

For the purposes of this analysis, we propose to use *stars* (unbalanced bicliques which are trees with one node and many leaves) for low data complexity key recovery. Those are unbalanced bicliques which are trees with one node and many leaves – for the analysis of block ciphers. We further apply this concept to discover a star-based independent-biclique key recovery for the full AES-128 which requires only one or two known plaintext-ciphertext pairs and works with computational complexity $2^{126.7}$.

3.2 The concept of stars

We start with the observation that the biclique does not have to be balanced – i.e. contain 2^d states in each of its two vertex sets – to cover 2^{2d} keys. Indeed, there is a biclique with just one state in one vertex set and 2^{2d} states in the other one: $S_x = \{x\}$, $S_y = \{y_{i,j}\}$, $i, j \in \{0, 2^d - 1\}$, where each $y_{i,j}$ is obtained by encrypting x with key $K[i, j]$, covering 2^{2d} keys. This biclique is called a star of dimension d.

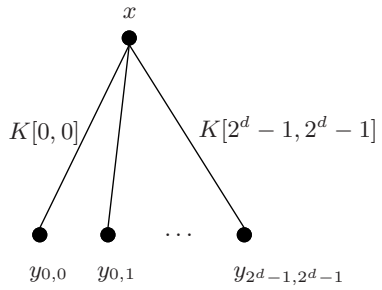


Fig. 8. Star: maximally unbalanced biclique of dimension d for the minimum data complexity

Now, if we place the star at the beginning of the cipher, and let x be the *plaintext* (or ciphertext) the data complexity of the MITM part of the key recovery will be exactly 1. Note that x can be any value and, thus, we deal with a known-plaintext key recovery here. The overall data complexity will be solely defined by the unicity distance of the cipher and, therefore, minimal theoretically attainable.

3.3 Stars from independent differentials

Similarly to balanced bicliques, stars can be constructed efficiently from independent sets of differentials. Unlike balanced bicliques, however, the necessary form of differentials is different. Suppose we have a set of $2^d - 1$ distinct related-key Δ-differentials from x to $y_{i,j}$:

$$(0, \Delta_i^K) \longmapsto \Delta_i$$

and a set of $2^d - 1$ distinct related-key ∇-differentials from over the same part of the cipher:

$$(0, \nabla_j^K) \longmapsto \nabla_j.$$

We assume that the Δ-differentials and ∇-differentials do not share any active nonlinear components. If input x, output $y_{0,0}$ and key $K[0,0]$ conform to both Δ- and ∇-differentials, then the values

$$\begin{array}{l} x \\ y_{i,j} = y_{0,0} \oplus \Delta_i \oplus \nabla_j, \text{ and} \\ K[i,j] = K[0,0] \oplus \Delta_i^K \oplus \nabla_j^K \end{array}$$

form a star of dimension d, with $\Delta_0 = \nabla_0 = \Delta_0^K = \nabla_0^K = 0$.

4 Improved Biclique Cryptanalysis of AES

4.1 Introduction

Today, the most frequently used block cipher is AES (Advanced Encryption Standard) [29] — the current U.S. encryption standard selected by NIST in an open competition. AES is the only publicly known cipher that is NSA-approved for protecting secret and top secret government information in the U.S.

The original work [13] introducing biclique key recovery leaves several questions unanswered though, which are crucial to judging about the real-world security of AES and implications of the biclique cryptanalysis in general:

– *Is there much potential in minimizing the data complexity* of the biclique attacks? In fact, it is low data complexity attacks that are most relevant in practice, especially in the context of efficient implementation of the attacks – the point clearly made in [12]. Actually, the data complexity of the original biclique attacks makes any practical implementation of them highly unreasonable since the standard brute force is very likely to be both cheaper and faster in reality (mainly due to the high requirements in terms of storage or oracle access).

– Though the new technique has been coined after bicliques, the initial structures are explicitly limited to balanced bicliques only, i.e. complete bipartite graphs the two set of vertices of which have exactly the same cardinality. So the question remains: Can one take any advantage of using *other types of bicliques* as initial structures?
– Finally, no comprehensive investigation of *attack optimality* in terms of computational complexity, data complexity or both has been performed. So it is still not clear if there are *faster biclique attacks*, even in the same class of the attacks as proposed in [13].

In this report, we aim to bridge these gaps and answer all three questions in the positive for the case of AES-128:

– As regards more general initial structures, we drop the balancedness requirement for bicliques. We propose to use *stars*, which are the most unbalanced bicliques, having only one vertex in one of the two disjoint sets of biclique vertices (see Section 3.1). This allows us to come up with a star-based biclique key recovery technique for block ciphers that inherently has the minimal theoretically attainable data complexity – the one due to the unicity distance.
– In terms of the attack space exploration for biclique cryptanalysis, we limit ourselves to the most promising class of attacks as applied to AES-128: Namely, we enumerate all truncated independent balanced bicliques and stars whose key modification trails have a single active byte in some state of the expanded key. For the sake of conciseness, we will refer to this class of attacks as based on *tight truncated independent bicliques and stars* (see Subsection 4.2). Clearly, this exploration does not cover many advanced and inherently harder-to-analyze attack vectors such as long-bicliques (bicliques of a lower dimension whose key modification differentials share active S-boxes) or narrow bicliques [44]. That is why, one cannot claim any formal bounds on the complexity of biclique attack complexity in general.
  Nonetheless, it is the tight truncated independent-biclique approach to key recovery that has resulted in the fastest attack on the full AES-128 so far, though requiring a part of the state to be recomputed for each key. However, we believe that this investigation does provide important new insight into the limits of the current techniques of biclique cryptanalysis when applied to AES-128.
– Using stars as initial structures, we propose the first key recovery on AES-128 with the minimal theoretically possible data complexity and faster than brute force. It requires 1 or 2 known plaintexts and has computational complexity $2^{126.7}$ (Subsection 4.3). Note that this computational complexity is only slightly higher than that of the original attack (of computational complexity

$2^{126.16}$ and data complexity 2^{88}). Moreover, both the time and data complexities of this new attack are actually lower than those of the attack in [12] (requiring $2^{128.9}$ time and 2^4 chosen plaintexts).

– Next, we exhaustively enumerate all attacks based on tight truncated independent bicliques and stars for AES-128 which have a data complexity lower than the full codebook. It turns out that among them the ones of computational complexity $2^{126.16}$ are fastest. Interestingly, this exactly corresponds to the original key recovery on AES-128 [13]. We further investigate the data complexity of these attacks for the biclique dimension $d = 8$ and show that the minimum data complexity is 2^{64} (cf. 2^{88} in the original attack). This implies that the original attack did not have the optimal data complexity.
– To investigate the limits of this class of biclique cryptanalysis, we abandon all restrictions on the data complexity and search for the fastest attack on AES-128 in this class. We find that the one with computational complexity $2^{125.6}$ is fastest (though requiring the full code book). It is interesting that this attack is based on independent-bicliques of length of 3 full AES rounds. This is the longest independent-biclique constructed so far for AES-128.

With respect to AES-128, the cryptanalytic results of the report are summarized in Table 1.

Table 1. Secret key recovery with bicliques for full AES-128 (10 rounds)

data	computations	memory	success prob	biclique length (rounds)	property shown	reference
2^{88} CC	$2^{126.16}$	2^8	1	2.5	-	[13]
2^{88} CP	$2^{126.89}$	2^8	1	2	-	[12]
Unic. dist: 1 or 2 KP	$2^{126.7}$	2^8	1	1	fastest with minimum data	Subsection 4.3
2^{64} CC	$2^{126.16}$	2^8	1	2.5	fastest with $< 2^{128}$ data	Subsection 4.4
2^{128}	$2^{125.6}$	2^8	1	3	fastest	Subsection 4.4

4.2 A search tool for biclique attacks on AES-128

In this section, we describe how we enumerate all biclique key recoveries in a large promising class of biclique attacks.

Enumerating bicliques. Clearly, going over all possible initial structures, even without enumerating possibilities for the actual key recovery, would be infeasible for the AES. So we have to confine the search space of attacks by imposing

some limitations. In here, we describe the search space along with the respective justifications:

- First, we consider bicliques (complete bipartite graphs) as initial structures. We stress that we include *both balanced bicliques and stars* in our search, unlike the attacks of [13] and [12] that consider only balanced bicliques.
- Second, we restrict the search to *independent-bicliques only*. Those are exactly the bicliques that can be constructed from independent related-key differentials, that is, related-key differentials that do not share active non-linear components. The principle of independent-biclique construction is described in Subsection 2.1.
  This constraint excludes such bicliques as long-bicliques [13] and narrow-bicliques [44], which are especially challenging to enumerate. However, though not optimal in the number of rounds covered, it is the independent-bicliques that attain the highest advantages over brute force for full AES-128 so far.
- Third, for AES-128, we confine the search to independent related-key differentials that have a key state in their trails with exactly one active byte. Note that this byte does not have to be the byte where the key difference is injected and the key difference still can be injected in multiple bytes. Actually, the best biclique key recoveries on AES-128 proposed so far are based on such bicliques.
- Finally, to keep the search space from exploding, we have to consider the trails of the bilciques in a *truncated* manner: We do not differentiate between the active values of the key modification trails in our bicliques (values of differences in the related-key differentials). In particular, this means that, once activated, a difference in a byte of a trail cannot be cancelled out. This is a significant but necessary limitation since we believe it is infeasible to run the exhaustive search otherwise, for excessively high computational complexities.

We implement these restrictions in a C program and are able to successfully enumerate all *tight truncated independent balanced bicliques and stars* of AES-128 within a very limited time on a standard PC.

Searching for key recoveries. Having enumerated all the bilciques as described above exhaustively, we apply MITM techniques to each of the obtained initial structures to evaluate their time and data complexities. This is done as follows.

First of all, we set the optimization goal as *minimizing the time complexity for a given data complexity restriction*. That is, in each search for a key recovery, we fix an upper bound on the data complexity. Then we perform the exhaustive search over all possibilities for matching. In terms of key enumeration, we impose

the restriction that the forward and backward key modifications should have at least one state of linear intersection. This stipulates the full key space coverage and success probability of 1. The MITM techniques used include partial matching (the matching is performed on a byte of the state to save computations) and the cut-and-splice technique (so that trails can go over the encryption/decryption oracles to win degrees of freedom).

To evaluate the time complexity of a key recovery efficiently on-the-fly, the tool uses the computational model proposed in [13]: All linear operations (AddRoundKey, ShiftRows, and MixColumns) are ignored and one counts only the number of S-box computations that have to be performed. One full AES-128 computation is, thus, equivalent to 200 S-box computations – a metric that proved to be meaningful in practice [12]. The evaluated time complexity is output in the numbers of S-box computations that have to be performed per key tested. Again, this is the parameter that has lead to the fastest attacks so far since it makes the key group larger and minimizes the impact of biclique construction on the total complexity.

Depending on the data complexity restriction, the tool can find the optimal attack (the attack with lowest evaluated time complexity under the data complexity restriction) within a very limited lime on our PC.

As a second optimization goal, we have *minimizing the data complexity for a given time complexity*. This second optimization is applied once a key recovery has been found in the previous step. At this point, we already know that there are no faster key recoveries in our search space. So we check if the data complexity of the fastest attack identified can be reduced. This task typically requires much less computations and can usually be completed within an hour on our computational platform.

Applications to find attacks with minimal data and time complexities. We applied the tool to search for three data complexity restrictions:

- *Minimum data complexity:* In fact, the minimum data complexity attack of Subsection 4.3 was discovered using this tool by setting the upper bounds of the key recovery to its theoretical minimum of the unicity distance. So we can claim that this is the fastest biclique key recovery with the minimal data complexity of exactly the unicity distance in the class of biclique and key recoveries covered by our tool. To recall its details, this key recovery requires $2^{126.7}$ time and 1 or 2 KPs of data. Note that the tool suggests that going for a star as the initial structure is optimal in this case.
- *Data complexity strictly lower than the full codebook:* This restriction is a standard line that is informally drawn between interesting attacks – that

require less that the full codebook of texts - and less interesting attacks – that can only work with the full codebook. The tool demonstrates that the fastest biclique key recoveries in the covered class with these restrictions require $2^{126.16}$ time. Optimizing for data complexity among all attacks with this time complexity yields that the lowest data complexity is 2^{64} CCs. This attack is summarized in Subsection 4.4. This attack is based on a balanced biclique of the exactly same lenth as the AES-128 biclique of [13]. However, the form of the biclique in [13] was not optimal in terms of data complexity.

– *No data complexity constraint:* The tool finds that the fastest biclique key recovery in the entire class of biclique attacks covered is the one requiring $2^{125.6}$ time and the full codebook of data. This attack provides an important insight into the limits of the independent-biclique approach as developed so far. This attack is outlined in Subsection 4.4 and relies on balanced bicliques.

4.3 Minimum data complexity key recovery for AES-128

Stars of dimension 8 over 1 round of AES-128 In AES-128, it is possible to construct a star of dimension 8 over the first round. Δ-trail activates byte 0 of key \$0. ∇-trail activates byte 1 of key \$0, see Figure 9(a). Difference propagation in those differentials over one round is non-overlapping till the end of round 1. In state #4, there is a linear overlap between those and, already in round 2, one has to recompute 2 S-boxes for each key. See Figure 9(a).

Rather surprisingly, even if the length of the star is just one round, the form of its trails is such that this short biclique still allows the adversary to obtain a reasonable computational advantage over brute force.

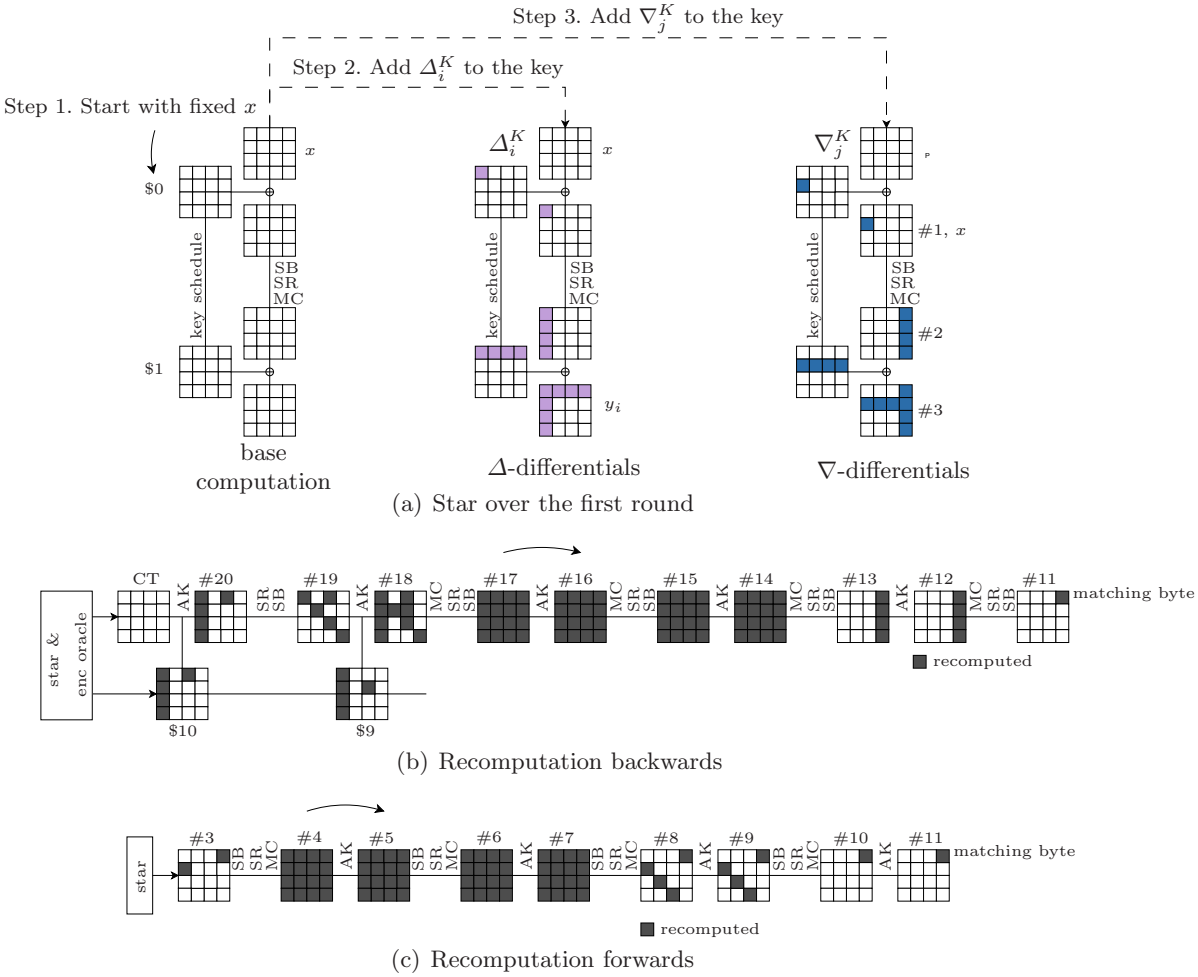
Key enumeration. We define the 2^{112} groups of keys with respect to \$0, i.e. the first subkey which equals the master key. The base keys are all keys of the form

,

where bytes 0 and 1 are fixed to zero. The other 14 bytes are fixed anew for each key group. The keys in each group are enumerated with respect to the base key by applying difference:

,

which yields 2^{16} keys in a group.



(a) Star over the first round

(b) Recomputation backwards

(c) Recomputation forwards

Fig. 9. Fastest biclique attack on AES-128 with minimum data: time $2^{126.7}$ and data 1 or 2 KPs

Matching and recomputations. In the forward direction of matching, starting in round 2, a part of the state has to be recomputed for each key. In round 2, only 2 S-boxes have to be recomputed. Starting in round 3 and forwards, the propagation affects the whole state. See Figure 9(c).

In the backward direction of matching, one starts with the ciphertext obtained using the encryption oracle under the right key for plaintext x. The Δ- and ∇-propagations in the key schedule are such that only 5 bytes of the \$10 depend on both Δ and ∇. This means that only 5 S-boxes have to be recomputed in round

10. Starting in round 9 and backwards, the propagation affects the full state. See Figure 9(b).

We match on byte 12 in state #11 of round 5, in which only one S-boxes need recomputation. In round 4 and round 6, only 4 S-boxes, respectively, are recomputed. The S-boxes in the four remaining rounds need to be recomputed completely (another 64 S-boxes). No S-box recomputations are needed in the key schedule.

Complexities. The matching yields a recomputation of 80 out of 200 S-boxes. Thus, $C_{recomp} \approx 2^{14.678}$ in one key group. About 2^8 keys will be suggested in each key group and need to be tested on more bytes of matching. This requires at most 4 S-boxes to be recomputed in both rounds 4 and 6 as well as 1 S-box in round 5, i.e. at most $C_{falsepos} \approx 2^{3.526}$. The complexity of precomputations and star generation is upper-bounded by $C_{precomput} \approx 2^{8.5}$ full AES computations. Thus, $C_{full} \approx 2^{126.698}$.

The data complexity exactly corresponds to the unicity distance of AES-128 – the minimal data complexity theoretically attainable. 1 known plaintext-ciphertext pair can sometimes be enough (with success probability of $1/e \approx 0.3679$). 2 known plaintext-ciphertext pairs yield a success probability of practically 1.

4.4 Fastest key recoveries for AES-128

Fastest biclique key recovery with less than the full codebook of data. This attack is based on a balanced biclique of dimension 8 of the last 2.5 rounds of AES-128. It is depicted in Figure 10(a). The forward and backwards trails in the biclique have an intersection in byte 0 of \$8. However, this intersection is in a linear operation (xor) of the key schedule and does not affect the biclique property. Note that the forward and backward trails in original attack did not have any intersection at all over the biclique. This is the additional degree of freedom used in our key recovery. See Figure 10(a).

The key is enumerated in \$9 which is the only key state linear in the key modification both in forward and backward trails. The bytes of key enumeration with i and j are non-intersecting. i is placed in bytes 0,4,8, and 12. j is put in bytes 5 and 9. The base key in each group is chosen such that the key coverage is complete and there are no intersections between the key groups: The bytes not affected by key modification run over all possibilities. In the bytes affected by i-modification, byte 0 is always set to zero and bytes 4,8, and 12 run over all possibilities. Similarly, in the bytes of j-modification, byte 5 is always 0 and byte 9 accepts all possible values. Thus, the key group size is 2^{16}.

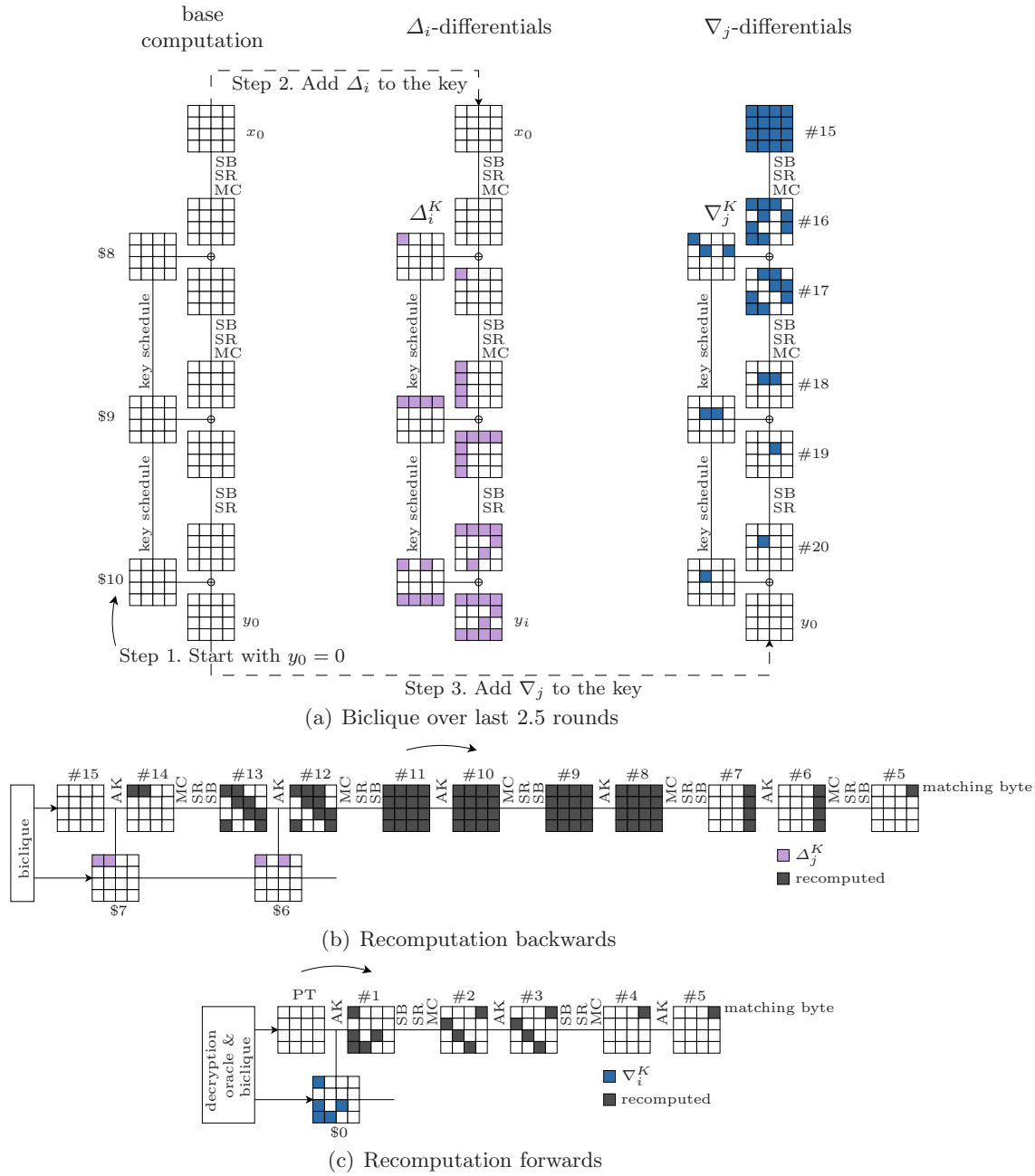


Fig. 10. Fastest biclique attack on AES-128 with less than full codebook: time $2^{126.16}$ and data 2^{64}

The key recovery is MITM with partial matching in byte 12 of data state #5 of round 3, where only one S-box needs recomputation. In round 2 and round 4, only 4 S-boxes, respectively, are recomputed. In round 7, only 8 S-boxes are recomputed. In round 1, 5 S-boxes are recomputed as the plaintext is influenced by 5 active bytes of the backward key modification through the key schedule. The S-boxes of rounds 5 and 6 have to be recomputed completely. See Figures 10(b) and 10(c). Rounds 8-10 are covered by the biclique.

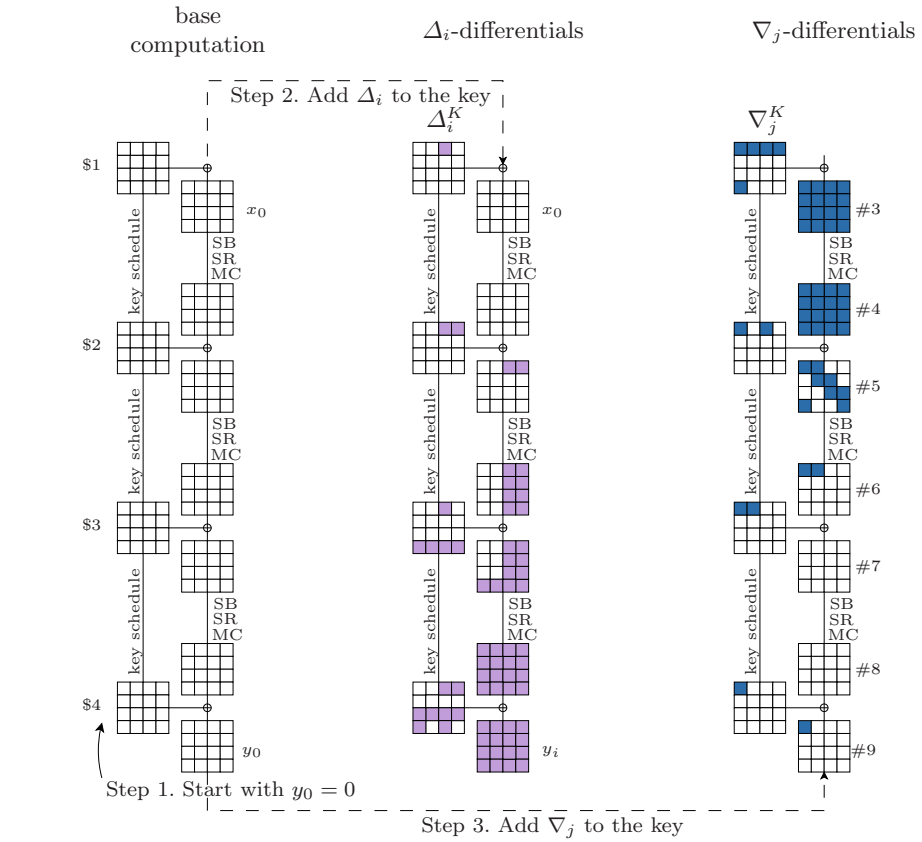
All in all, also counting the necessary recomputations in the key schedule we arrive at 55 S-boxes that have to be recomputed for each key, resulting in $C_{recomp} \approx 2^{14.137}$. As in the previous attacks, $C_{falsepos} \approx 2^{3.526}$ and $C_{precomp} \approx 2^{8.5}$. This yields $C_{full} \approx 2^{126.16}$. The data complexity as defined by the form of the biclique is 2^{64} chosen ciphertexts. As in all our attacks, the success probability is 1.

Fastest biclique key recovery. When we drop the constraint of data complexity being below the full codebook, we can construct a balanced biclique of dimension over 3 full AES-128 rounds and with the minimal recomputation of just one S-box in the fourth round right after the biclique. The biclique is placed in rounds 2-4 which implies the data complexity of 2^{128} for the backward trail. See 11(a).

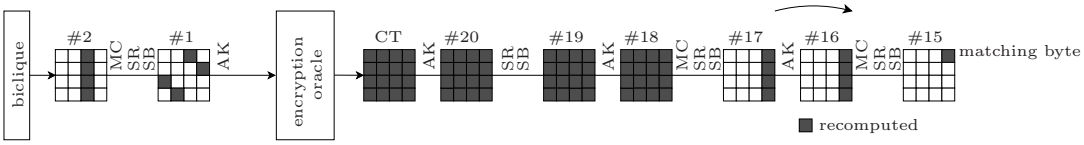
In the forward recomputation, one byte in round 5 is affected. See Figure 11(b). In the backward direction, four bytes in round 1 are recomputed. See Figure 11(c). The remainder of the rounds is the matching on one byte in round 8. This involves recomputation of 2 full rounds (amounting to 32 S-boxes), invocation of the encryption oracle, and recomputation of 9 S-boxes in rounds 7,8 and 9.

Thus, we have to recompute 37 S-boxes altogether for each key, resulting in $C_{recomp} \approx 2^{13.56}$. For all attacks of this type (matching on one byte), we have $C_{falsepos} \approx 2^{3.526}$. The precomputations are approximately the same with $C_{precomp} \approx 2^{8.5}$. This yields $C_{full} \approx 2^{125.6}$. The data complexity in this attack is the full codebook. The success probability is again 1 since key coverage is complete.

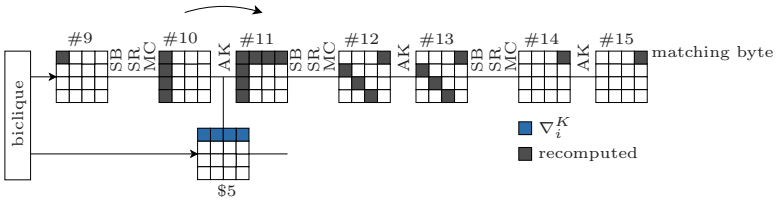
This key recovery can be converted into a preimage search for the compression function constituted by AES-128 in Davies-Meyer mode. Here the attack works offline and does not have to make any online queries. This preimage attack requires $2^{125.6}$ AES-128 operations and finds a preimage with probability about 0.632. The generic preimage search would require 2^{128} time to succeed with probability 0.632.



(a) Biclique over 3 rounds in the middle



(b) Recomputation backwards



(c) Recomputation forwards

Fig. 11. Fastest biclique attack on AES-128: time $2^{125.6}$ and full codebook

5 Biclique Cryptanalysis of Camellia

5.1 Summary

For the purpose of this analysis, we confine ourselves to the case of independent bicliques with at most a single-byte key modification. This is the type of biclique cryptanalysis that proved most successful for AES. Because of the SPN structure used in the round function of Camellia, we can presume that this will be also a good choice for Camellia. Moreover, since the purpose of this document is an indication of a security level for these ciphers, we take the approach of letting the data complexity of such biclique attacks be unlimited. This might result in biclique attacks that require full code book, though not necessarily. However, their computational complexity can be seen as a lower bound of what can be attained for Camellia with these techniques. In this preliminary document, we only report results for the full Camellia since those are most interesting.

The properties of the key schedule are highly important to make a biclique attack work. In Camellia, one can observe that the key schedule is designed to make it robust to related-key differential attacks. This is done by maintaining different types of intermediate keys: K_L and K_A for Camellia-128 as well as K_L, K_R, K_A and K_B for Camellia-192 and Camellia-256, where K_A and K_B depend in a complex way on K_L or K_L and K_R, respectively. Moreover, the key schedule is such that no three adjacent rounds use subkeys from either K_L and K_R or K_A and K_B. This complicates the biclique construction and makes its effective length shorter. It is also the reason to enumerate the keys with respect to a base key of $K_L|K_R$ in each key group.

Applying this approach to Camellia, we report key recovery results as follows. For the full Camellia-128, we can recover the key with a complexity of $2^{127.6}$, data complexity 2^{128}, negligible memory complexity and success probability 1. We have found a key recovery for the full Camellia-192 with computational complexity $2^{191.7}$, data complexity 2^{128}, negligible memory complexity and success probability 1. For the full Camellia-256, we report a key recovery with a computational complexity of $2^{255.7}$, data complexity 2^{128}, negligible memory complexity and success probability 1. Note that if we just apply exhaustive MITM key recovery without constructing a biclique structure, it is possible to convert these attacks to lower data complexity attacks at the expense of an increase in the computational complexity. We choose not to do so since the time complexities of the key recoveries are quite high even with bicliques.

Table 2. Summary of our biclique key recoveries for full Camellia

cipher	rounds	attack type	data	time	memory
Camellia-128	18 (of 18)	biclique MITM	2^{128}	$2^{127.6}$	small
Camellia-192	24 (of 24)	biclique MITM	2^{128}	$2^{191.7}$	small
Camellia-256	24 (of 24)	biclique MITM	2^{128}	$2^{255.7}$	small

5.2 Base line: complexity of brute force

We understand brute force key recovery for a cipher with a k-bit key as the complete computation of the cipher encryption on each of the 2^k keys. The first and most time-consuming filtering stage can be performed using one plaintext-ciphertext pair then, followed by subsequent and much less frequent tests if needed. So before discussing advantages of any other key recovery procedure, one needs to establish a base line for the respective computation of relative advantage – namely, quantify the complexity of the brute force key recovery with success probability 1.

For Camellia, just in line with the computational model introduced in the biclique attacks for AES, the unit will be the application of an 8-bit S-box. This is arguably the most consuming component in terms of implementations — both software (table lookups or bitslice) or hardware (ASIC or FPGA).

Table 3. Computational base line for Camellia (brute force)

cipher	data transform (rounds)	key schedule (rounds)	total (rounds)	total (S-boxes)
Camellia-128	18R	4R	22R	176
Camellia-192	24R	6R	30R	240
Camellia-256	24R	6R	30R	240

As indicated in Table 3, one Camellia-128 computation requires in total 176 S-box applications, since each round transform includes 8 S-box applications and there is an equivalent of 22 rounds in the full Camellia-128. Similar applies to Camellia-192 and Camellia-256 yielding a total of 240 S-box applications.

5.3 Independent bicliques over 2 rounds

We describe here how to construct an independent biclique of dimension 8 over 2 rounds of a 6-round block in Camellia. For this, one needs to specify rounds

such that round 3 and 4 of the 6-round block use key K_L or K_R. In all Camellia versions, there are such pairs of numbers even using K_L only. So without unnecessary loss of generality, we construct bicliques in the rounds as specified in Table 4.

Table 4. 2-round bicliques and key modification for Camellia

cipher	type	key modification	subkey	round number
Camellia-128	forward	one byte	$(K_L <<< 15)_{L(64)}$	3
	backward	one byte	$(K_L <<< 15)_{R(64)}$	4
Camellia-192	forward	one byte	$(K_L <<< 45)_{L(64)}$	9
	backward	one byte	$(K_L <<< 45)_{R(64)}$	10
Camellia-256	forward	one byte	$(K_L <<< 45)_{L(64)}$	9
	backward	one byte	$(K_L <<< 45)_{R(64)}$	10

The key modification in both forward and backward directions is in 8 bits which corresponds to an independent biclique of dimension 8. The actual biclique construction is outlined in Figure 12.

In the backward direction, one byte key modification in round 4 of the 6-round block in the backward direction activates 5 bytes at the output of the F-function due to the specific choice of the byte position according to the specification of P. This, in turn, activates 5 bytes at the input of the F-function in round 3 of the 6-round block, which can activate 8 bytes at the output of this function.

In the forward direction, a similar key modification is applied. In round 3 of the 6-round block, a single byte of the round subkey is activated which gives 5 active bytes at the output of the F-function in that round. In the next round, this yields 5 active S-boxes and an 8-byte active output pattern.

We show now how this construction forms an independent 2-round biclique of dimension 8. First, we require that the input difference at the input of round-3 F-function in the backward propagation does not include the byte of key modification in the forward direction. And vice versa, we require that the input difference at the input of round-4 F-function in the forward propagation does not cover the byte of key modification of the backward propagation. Obviously, this can be easily attained by inspecting the P-layer. Now we observe that though we have a lot of difference intersections in the backward and forward directions over the 2 rounds, they never intersect in the non-linear components (S-boxes). This suffices for the construction of an independent biclique. Then, we see that we can

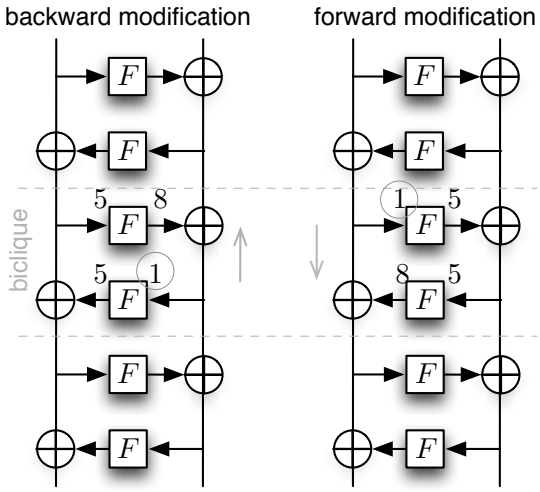


Fig. 12. Independent biclique construction over 2 rounds of Camellia: Numbers indicate the number of active bytes in the F-function right after the subkey addition and right after the P-layer. Numbers in grey circles indicate spots of key difference injections

inject up to an 8-bit difference in each direction. This defines the dimension of the biclique to be 8.

5.4 Partial matching over 3 rounds

Having construction a biclique, we can apply recomputation with precomputation over the biclique area to save operations. Moreover, we can have more savings due to partial matching which is illustrated in Figure 13.

We match in round 4 of a 6-round block in one byte of a 8-byte half state. This requires the recomputation of only 5 bytes in rounds 3 and 5 but full recomputations of rounds 1,2, and 6 (8 S-boxes each).

5.5 Recomputation over the remaining rounds and in the key schedule

All S-boxes in the remaining rounds of the data transform need to be recomputed for every key. In the key schedule though, some more savings are possible. A glance at the key schedule of Camellia (see Figure 16) suggests that no full recomputation is necessary given the type of key modification we have employed.

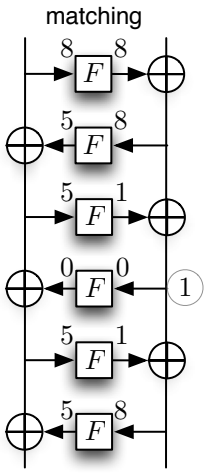


Fig. 13. Partial matching over 3 rounds for Camellia: Numbers indicate the number of bytes to be recomputed in the corresponding lines. The number in grey circle indicates the byte of matching

In Camellia-128, we need to compute K_A from every modification of K_L. This requires at most 4 rounds. However, since our key modification is in two distinct bytes of K_L which are in two distinct 64-bit halves of the user-supplied key, only one S-box needs to be recomputed in the first round. 5 bytes are actived at the output of this F-function and at the input of the F-function in the second round. Thus, we need to compute only 6 S-boxes instead of 16 in the first two rounds.

In Camellia-192 and Camellia-256, both K_A and K_B need to be computed from K_L and K_R. We have no modification in K_R and only single-byte modification in two distinct 64-bit halves of K_L again. So again we have a saving of 10 S-box computations in the key schedule recomputation.

5.6 Complexity

In total, we have saved 14 S-box computations due to the construction of the biclique, 16 S-box computations due to the partial matching and another 10 S-box computations due to the optimized recomputation in the key schedule. This results in 40 S-boxes saved per key tested.

Memory complexity in the biclique construction and recomputations within the key group is negligible because of the limited biclique dimension. The computational complexity for Camellia-128 is defined by the advantage factor of $\frac{176}{176-40}$

and amounts to about $2^{127.6}$. For Camellia-192 and Camellia-256, the advantage factor is $\frac{240}{240-40}$ which brings the computational complexity to $2^{191.7}$ and $2^{256.7}$, respectively. Note that we do not take into account the complexity of biclique construction since the dimension of the biclique is large enough to make it negligible.

The data complexity of this key recovery is 2^{128}, that is, the full codebook. Due to the specification of the key schedule – both the computation of the subkeys and the order of their usage in rounds – it is challenging to have both a non-trivial biclique (accounting for more computation savings) and a lower data complexity. For instance, in Camellia-128, if key modification is in K_L, then K_A will look random for each key modification which will lead to the propagation of the key modification to the full state at the plaintext or ciphertext, depending on where the biclique is exactly put. Similar applies to Camellia-192 and Camellia-256.

6 Zero Correlation Cryptanalysis of Camellia

6.1 Summary

In the course of the key recovery in zero correlation linear cryptanalysis, the correlation of a linear approximation over a part of the cipher needs to be evaluated for many key guesses. While the length of the underlying zero correlation approximation mainly determines the number of rounds that can be attacked, it is the partial encryption and decryption of plaintexts and ciphertexts up to the boundaries of the linear approximation that dominates the computational complexity of the attack. We use the discrete Fast Fourier transform to reduce the computational complexity at this end. For data complexity reduction, the novel multidimensional zero-correlation distinguisher is applied. Though some cryptanalysts prefer to skip $\mathrm{FL}/\mathrm{FL}^{-1}$ functions for simplicity, we analyze Camellia-192 and Camellia-256 with $\mathrm{FL}/\mathrm{FL}^{-1}$ functions, ensuring that we actually cryptanalyze Camellia transform and not a related variant.

For a generic construction like Camellia (balanced Feistel network), there are numerous of zero-correlation linear approximations over only 5 rounds. However, for Camellia, we are able to identify a bunch of 7-round zero-correlation linear approximations due to the properties of the $\mathrm{FL}/\mathrm{FL}^{-1}$ functions and the F-functions of the round transformation.

For 11-round Camellia-192, we report a key recovery requiring $2^{125.1}$ data, $2^{157.79}$ operations equivalent to 11 rounds of Camellia-192 as well as 2^{101} blocks of memory. For 12-round Camellia-256, we have identified a key recovery using $2^{125.9}$ data, $2^{234.31}$ operations equivalent to 12 rounds of Camellia-192 as well as 2^{165} blocks of memory. The 7-round zero-correlation linear approximations are of

the form $(0bb0bb0b|00000000) \rightarrow (00000000|h00h0hhh)$ where b and h are nonzero bytes.

Table 5. Summary of our zero correlation attacks on round-reduced Camellia with FL/FL^{-1} functions

cipher	rounds	attack type	data	time	memory
Camellia-192	11 of 24	zero correlation	$2^{125.1}$	$2^{157.79}$	2^{101}
Camellia-256	12 of 24	zero correlation	$2^{125.9}$	$2^{234.31}$	2^{165}

6.2 7-round zero-correlation linear approximations for Camellia with FL/FL^{-1} functions

In this section, we will present some zero-correlation linear hulls for 7-round Camellia with FL/FL^{-1} function. Firstly, we will introduce some properties for FL/FL^{-1}.

Property 1: If the input mask of FL function is $IM = (0|0|0|0|0|i|0|0)$, the output mask of FL function $OM = (0|?|?|0|0|?|?|0)$, where "?" is an unknown value.

Proof. If we denote the input mask for $IM = IM_L|IM_R$ and $OM = OM_L|OM_R$, from the definition of FL, we have

$$OM_L = IM_L \oplus ((IM_R \ggg 1) \cap k_L), OM_R = (OM_L \cup k_R) \oplus IM_R.$$

As

$$IM_L = 0,$$

$$OM_L = ((IM_R \ggg 1) \cap k_L) = ((0|i|0|0) \ggg 1) \cap k_L = (0|?|?|0)$$

and

$$OM_R = ((0, ?, ?, 0) \cup k_R) \oplus (0|i|0|0) = (0|?|?|0).$$

□

Similarly, we can get the following relation between IM and OM for FL function,

$$IM = (0|0|0|0|0|0|0|i) \Rightarrow OM = (?|0|0|?|?|0|0|?).$$

Property 2: If the output mask of FL^{-1} function is $OM = (i|0|0|0|0|0|0|0)$, the input mask of FL^{-1} function $IM = (i||0|0|?||0|0)$. If $OM = (0|0|0|0|i|0|0|0)$, it holds that

$$IM = (?|?|0|0|?|?|0|0).$$

Proof. From the definition of FL^{-1}, we have

$$IM_L = OM_L \oplus (OM_R(\ggg 1) \cap k_L), IM_R = (IM_L \cup k_R) \oplus OM_R.$$

As

$$OM = (0|i|0|0|0|0|0|0),$$

one has that

$$OM_R = 0$$

and

$$OM_L = (0|i|0|0).$$

Then we have $IM_L = OM_L = (0|i|0|0)$ and $IM_R = (0|?|0|0)$, where "?" is the unknown value.

As $OM = (0|0|0|0|i|0|0|0)$, we have that $OM_L = 0$ and $OM_R = (i|0|0|0)$. Then we have

$$IM_L = (OM_R \ggg 1) = (?|?|0|0)$$

and

$$IM_R = ((?|?|0|0) \cup k_R) \oplus (0|i|0|0) = (?|?|0|0),$$

where "?" is the unknown value. □

In the similar way, we can get the following relations between IM and OM for FL^{-1},

$$OM = (0|i||0|0|0|0|0) \Rightarrow IM = (0|i||0|0|?|?|0),$$

$$OM = (0|0|i|0|0|0|0|0) \Rightarrow IM = (0|0|0|i|0|0|0|?),$$

$$OM = (0|0|0|i|0|0|0|0) \Rightarrow IM = (0|0|0|i|0|0|0|?),$$

$$OM = (0|0|0|0|0|i|0|0) \Rightarrow IM = (0|?|?|0|0|?|?|0),$$

$$OM = (0|0|0|0|0|0|i|0) \Rightarrow IM = (0|0|?|?|0|0|?|?),$$

$$OM = (0|0|0|0|0|0|0|i) \Rightarrow IM = (?|0|0|?|?|0|0|?).$$

With the above properties, we can get some zero-correlation linear hulls for 7-round Camellia.

Type 1: For 7-round Camellia consisting of

$$(F|F|F|F|FL/FL^{-1}|F|F|F),$$

if the input mask of the first round is

$$(0|b|b|0|b|b|0|b, 0|0|0|0|0|0|0|0)$$

and the output mask of the last round is

$$(0|0|0|0|0|0|0|0, h|0|0|h|0|h|h|h),$$

the correlation of the linear hull for the 7-round Camellia is zero, where $b, h \in \mathbb{F}_2^8, b \neq 0, h \neq 0$.

Proof. From Fig. 14(a), the input mask

$$(0|b|b|0|b|b|0|b, 0|0|0|0|0|0|0|0)$$

produces the input mask

$$(0|0|0|0|0|a|0|0)$$

for FL function, and the output mask

$$(0|0|0|0|0|0|0|0, h|0|0|h|0|h|h|h)$$

results in the output mask

$$(f_1|f_2|f_3|f_4|f_5|f_6|f_7|f_8 \oplus i)$$

for FL function, where

$$a, b, h, i, c_k(k \in \{2, 3, 5, 7, 8\}), g_l(l \in \{1, 4, 5, 6, 7\})$$

and $f_j(1 \leq j \leq 8)$ are non-zero value. From Property 1, if the input mask for FL function

$$IM = (0|0|0|0|0|a|0|0),$$

the output mask

$$OM = (0|?|?|0|0|?|?|0).$$

Then $f_1 = 0, f_4 = 0, f_5 = 0$ and $f_8 \oplus i = 0$. According to the relations between f_j and g_l, the following equations should hold,

$$f_4 = 0 \Rightarrow g_1 \oplus h \oplus g_5 \oplus g_6 \oplus h \oplus g_7 \oplus h = g_1 \oplus g_5 \oplus g_6 \oplus g_7 \oplus h = 0,$$

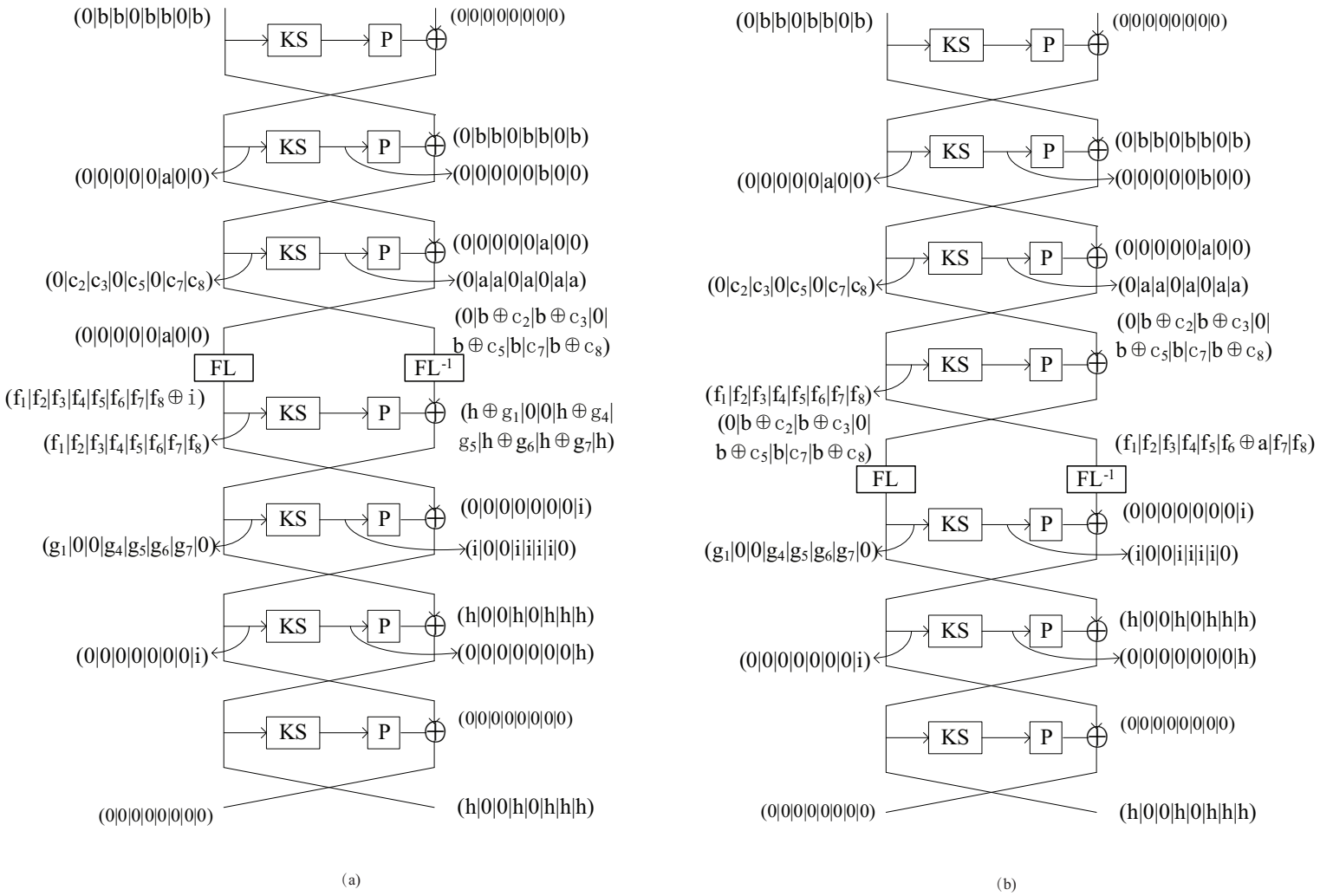


Fig. 14. Zero-correlation linear approximations for 7-round Camellia

$$f_5 = 0 \Rightarrow g_1 \oplus h \oplus g_5 \oplus g_7 \oplus h \oplus h = g_1 \oplus g_5 \oplus g_7 \oplus h = 0.$$

From the above two equations, we can deduce $g_6 = 0$, which contradicts that $g_6 \neq 0$. As a result, the linear hull is a zero-correlation linear hull.

There is another zero-correlation linear hull for 7-round Camellia which is as follows

$$(b|0|0|b|0|b|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|h|h|0|h|h|0|h).$$

The input mask

$$(b|0|0|b|0|b|b|b, 0|0|0|0|0|0|0|0)$$

and the output mask

$$(0|0|0|0|0|0|0|0, 0|h|h|0|h|h|0|h)$$

will result that the input mask is $IM = (0|0|0|0|0|0|0|a)$ and the output mask is

$$(f_1|f_2|f_3|f_4|f_5|f_6 \oplus i|f_7|f_8)$$

for FL function. From Property 1, the input mask for FL function

$$IM = (0|0|0|0|0|0|0|a),$$

and the output mask is $OM = (?|0|0|?|?|0|0|?)$. Then $f_2 = 0, f_3 = 0, f_6 \oplus i = 0$ and $f_7 = 0$. According to the relations between f_j and g_l, the following equations should hold,

$$f_2 = 0 \Rightarrow g_3 \oplus h \oplus g_5 \oplus h \oplus g_7 \oplus g_8 \oplus h = g_3 \oplus g_5 \oplus g_7 \oplus g_8 \oplus h = 0,$$

$$f_7 = 0 \Rightarrow g_3 \oplus h \oplus g_5 \oplus h \oplus h \oplus g_7 = g_3 \oplus g_5 \oplus g_7 \oplus h = 0.$$

It is easy to deduce that $g_8 = 0$, which contradicts that $g_8 \neq 0$. As a result, the linear hull is also a zero-correlation linear hull. □

Type 2: For 7-round Camellia consisting of

$$(F|F|F|FL/FL^{-1}|F|F|F|F),$$

if the input mask of the first round is

$$(0|b|b|0|b|b|0|b, 0|0|0|0|0|0|0|0)$$

and the output mask of the last round is

$$(0|0|0|0|0|0|0|0, h|0|0|h|0|h|h|h),$$

the correlation of the linear hull for the 7-round Camellia is zero, where $b, h \in \mathbb{F}_2^8, b \neq 0, h \neq 0$, see Fig. 14(b).

Proof. From Fig. 14(b), the input mask

$$(0|b|b|0|b|b|0|b, 0|0|0|0|0|0|0|0)$$

produces the input mask

$$(f_1|f_2|f_3|f_4|f_5|f_6 \oplus a|f_7|f_8)$$

for FL^{-1} function, and the output mask

$$(0|0|0|0|0|0|0|0, h|0|0|h|0|h|h|h)$$

produces the output mask

$$(0|0|0|0|0|0|0|i)$$

for FL^{-1} function, where $a, b, h, i, c_k (k \in \{2, 3, 5, 7, 8\}), g_l (l \in \{1, 4, 5, 6, 7\})$ and $f_j (1 \leq j \leq 8)$ are non-zero value. From Property 2, the output mask for FL^{-1} function $OM = (0|0|0|0|0|0|0|i)$, the input mask $IM = (?|0|0|?|?|0|0|?)$. Then $f_2 = 0, f_3 = 0, f_6 \oplus a = 0$ and $f_7 = 0$. According to the relations between f_j and g_l, the following equations should hold,

$$f_2 = 0 \Rightarrow c_3 \oplus b \oplus c_5 \oplus b \oplus c_7 \oplus c_8 \oplus b = c_3 \oplus c_5 \oplus c_7 \oplus c_8 \oplus b = 0,$$

$$f_7 = 0 \Rightarrow c_3 \oplus b \oplus c_5 \oplus b \oplus c_7 \oplus b = c_3 \oplus c_5 \oplus c_7 \oplus b = 0.$$

So we can derive that $c_8 = 0$, which contradicts that $c_8 \neq 0$. As a result, the linear hull is also a zero-correlation linear hull. □

There are some similar zero-correlation linear hulls for 7-round Camellia as follows:

$$(b|0|0|b|0|b|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|h|h|0|h|h|0|h),$$

$$(b|b|0|0|b|0|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|0|h|h|h|h|h|0),$$

$$(0|0|b|b|b|b|b|0, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|0|0|h|0|h|h),$$

$$(0|b|b|b|0|b|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|h|h|h|0|h|h|h),$$

$$(b|0|b|b|b|0|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|0|h|h|h|0|h|h),$$

$$(b|b|0|b|b|b|0|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|0|h|h|h|0|h),$$

$$(b|b|b|0|b|b|b|0, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|h|0|h|h|h|0).$$

To save on space, we will not provide the proof for these zero-correlation linear approximations here.

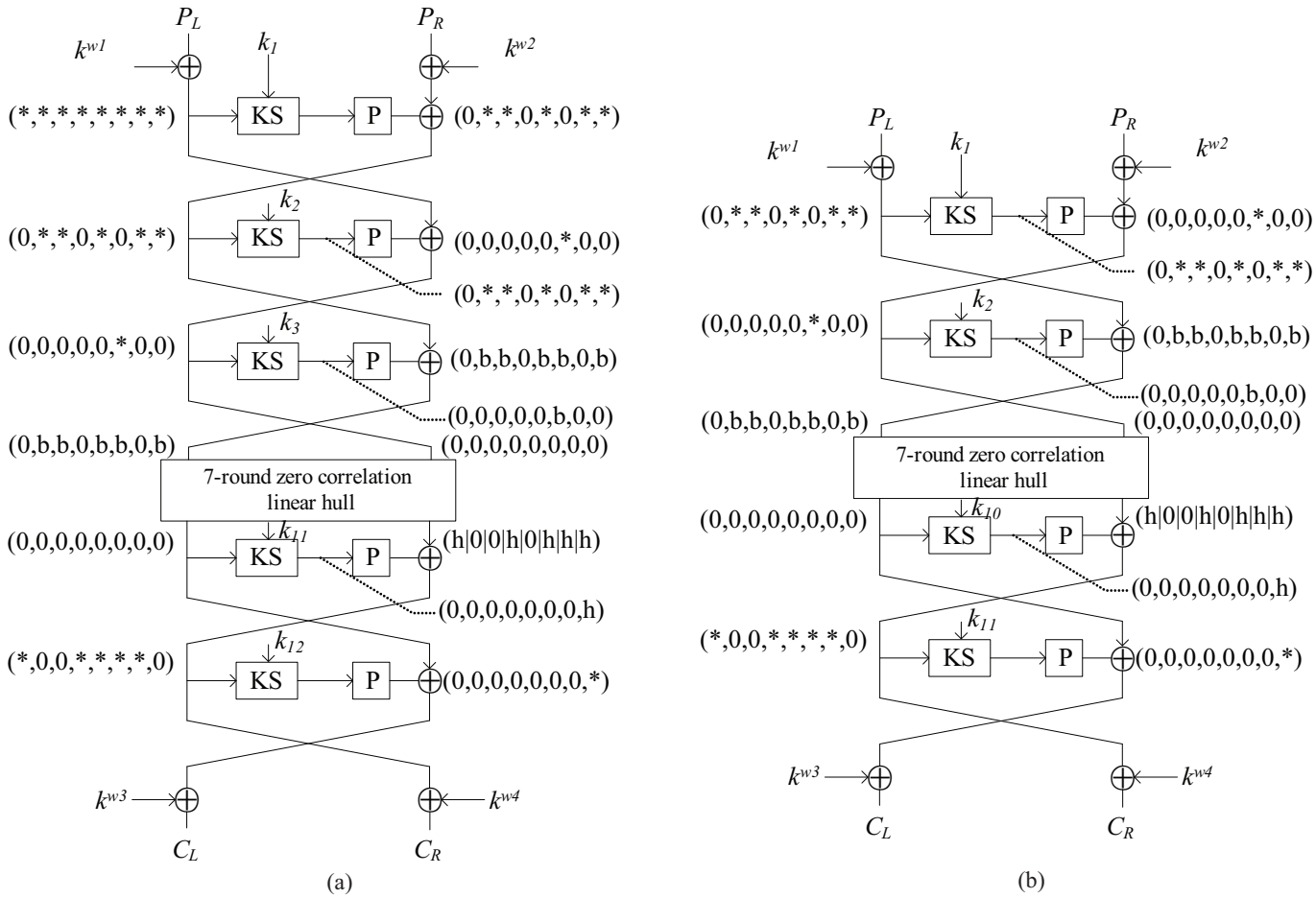


Fig. 15. Zero correlation linear attacks on 12-rounds Camellia-256 and 11-rounds Camellia-192

6.3 Attack on 12-round Camellia-256

We use the 7-round zero-correlation linear approximations to attack 12-round Camellia-256, see Fig. 15(a). In Fig. 15(a), we also try to determine the mask value before or after the 7-round zero-correlation linear hull, where $*$ denotes that the mask value for the corresponding byte position cannot be determined.

In the following, we will use some notations. $X^{i_1,i_2,\cdots}$ denotes the i_1-th, i_2-th,... byte of X and X can be the plaintext word, ciphertext word or subkey word, S^j denotes the output of the j-th S-Box, F_j denotes the output of the round function for the j-th round and F_j^l denotes the l-th output byte of the round function for the j-th round. If we denote $K_0 = k^{w1} \oplus k_1$, $K_1 = k^{w2} \oplus k_2$, $K_2 = k^{w1} \oplus k_3$, $K_3 = k^{w3} \oplus k_{11}$, and $K_4 = k^{w4} \oplus k_{12}$, we can get the following linear approximation:

$$\begin{aligned}
&\mathbf{b}^T \cdot P_R^6 \oplus \mathbf{h}^T \cdot C_R^8 \oplus \alpha^T \cdot P_R^{2,3,5,8} \oplus \beta^T \cdot C_R^{1,4,6,7} \oplus \mathbf{b}^T \cdot S^6(P_L^6 \oplus K_0^6) \\
&\oplus \mathbf{b}^T \cdot S^6\{P_L^6 \oplus K_2^6 \oplus F_2^6[P_R^{2,3,5,7,8} \oplus K_1^{2,3,5,7,8} \oplus F_1(P_L^{1,\ldots,8} \oplus K_0^{1,\ldots,8})\} \\
&\oplus \mathbf{h}^T \cdot S^8[C_L^8 \oplus K_{11}^8 \oplus F_5^8(C_R^{1,4,5,6,7} \oplus K_4^{1,4,5,6,7})] = 0.
\end{aligned} \tag{5}$$

where $\alpha^T = (\mathbf{b}, \mathbf{b}, \mathbf{b}, \mathbf{b})$, $\beta^T = (\mathbf{h}, \mathbf{h}, \mathbf{h}, \mathbf{h})$, $\mathbf{b} \in \mathbb{F}_{2^8}, \mathbf{b} \neq 0$, $\mathbf{h} \in \mathbb{F}_{2^8}, \mathbf{h} \neq 0$.

In order to express Equation (5) as a circulant matrix, we will fix some parameters to obtain the following circulant matrix:

$$\begin{aligned}
&M(P_L^{1,2,3,4,5,7,8} | C_R^5 | C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}), K_0^{1,2,3,4,5,7,8} | K_4^5 | K_4^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}) = \\
&\alpha^T \cdot P_R^{2,3,5,8} \oplus \beta^T \cdot C_R^{1,4,6,7} \oplus \mathbf{b}^T \cdot S^6(P_L^6 \oplus K_0^6) \\
&\oplus \mathbf{b}^T \cdot S^6\{P_L^6 \oplus K_2^6 \oplus F_2^6[P_R^{2,3,5,7,8} \oplus K_1^{2,3,5,7,8} \oplus F_1(P_L^{1,\ldots,8} \oplus K_0^{1,\ldots,8})]\} \\
&\oplus \mathbf{h}^T \cdot S^8[C_L^8 \oplus K_{11}^8 \oplus F_5^8(C_R^{1,4,5,6,7} \oplus K_4^{1,4,5,6,7})].
\end{aligned} \tag{6}$$

In Equation (6), P_L^6 has also been involved outside the round function F_1, so we must fix the value of P_L^6. Moreover, in order to use more zero-correlation linear hulls, more values of $\mathbf{h}$ will be taken. At the same time, in order to control the time complexity, we will take all values for $\mathbf{h}$ with the Hamming weight one or two. It means that $C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})$ have been involved in $\beta^T \cdot C_R^{1,4,6,7}$, so we must fix the value of bits $C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})$ and express the remaining bits $C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}})$ in circulant matrix. Here $\bar{\mathbf{h}} = \mathbf{h} \oplus ff_x$.

We proceed with the attack as follows:

– Set the global counter C_κ as zero for each of 2^{160} possible values of

$$\kappa = (K_0^{1,2,3,4,5,6,7,8}|K_1^{2,3,5,7,8}|K_2^6|K_3^8|K_4^{1,4,5,6,7}).$$

– For $\mathbf{h} \in S$ and $\mathbf{b} \in \mathbb{F}_{2^8}, \mathbf{b} \neq 0$, $S = \{\mathbf{x} : 1 \leq W(\mathbf{x}) \leq 2, x \in \mathbb{F}_{2^8}\}$, $|S| = \binom{8}{2} + \binom{8}{1} = 36$, where $W(\mathbf{x})$ is the Hamming weight of $\mathbf{x}$:

- Data counting phase:
  1. For N plaintext-ciphertext pairs, extract $56 + 4W(\mathbf{h})$ bits value

$$i = (P_R^{2,3,5,7,8}|P_L^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8)$$

  and the corresponding $64 + 4(8 - W(\mathbf{h}))$ bits value

$$j = (P_L^{1,2,3,4,5,7,8}|C_R^5|C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}})).$$

  2. Increment the counter j of the vector $\mathbf{x_i}$ according to the parity of

$$\mathbf{b}^T \cdot P_R^6 \oplus \mathbf{h}^T \cdot C_R^8.$$

- Key counting phase: For each possible $64 + 4W(\mathbf{h})$ bits value k of

$$(K_0^6|K_2^6|K_1^{2,3,5,7,8}|K_3^8|K_4^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})) :$$

  1. For each possible $56 + 4W(\mathbf{h})$ bits value

$$(P_R^{2,3,5,7,8}|P_L^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8) :$$

  compute

$$Z = \alpha^T \cdot P_R^{2,3,5,8} \oplus \beta^T \cdot C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h}) \oplus \mathbf{b} \cdot S^6(P_L^6 \oplus K_0^6).$$

  (a) Select the corresponding vector of counters $\mathbf{x_i}$
  (b) Compute the first column M_i of the matrix

$$M_i(P_L^{1,2,3,4,5,7,8}|C_R^5|C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}),$$
$$K_0^{1,2,3,4,5,7,8}|K_4^5|K_4^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}})) =$$

$$(-1)^{Z \oplus \mathbf{b}^T \cdot S^6\{P_L^6 \oplus K_2^6 \oplus F_2[P_R^{2,3,5,7,8} \oplus K_1^{2,3,5,7,8} \oplus F_1(P_L^{1,\ldots 8} \oplus K_0^{1,\ldots 8})]\}} \times$$

$$(-1)^{\mathbf{h}^T \cdot S^8[C_L^8 \oplus K_3^8 \oplus F_5^8(C_R^{1,4,5,6,7} \oplus K_4^{1,4,5,6,7})]}.$$

As M_i is $(64 + 4(8 - W(\mathbf{h})))$-level circulant, this information is sufficient to define M completely (requires $2^{64+4(8-W(\mathbf{h}))}$ operations which are equivalent to

$$2^{64+4(8-W(\mathbf{h}))} \cdot (1 + \tfrac{5}{8} + \tfrac{1}{8} + \tfrac{5}{8} + \tfrac{1}{8})/12 = 2^{64+4(8-W(\mathbf{h}))} \cdot 2.5/12 =$$

$$2^{64+4(8-W(\mathbf{h}))-2.26} = 2^{93.74-4W(\mathbf{h})}$$

12-round encryptions.

(c) The evaluation of the linear approximation for the subset of texts with the corresponding

$$(P_R^{2,3,5,7,8}|P_L^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8)$$

value is given by the matrix vector product $\epsilon_i = M \cdot x_i$ which requires

$$3 \cdot 2^{(64+4(8-W(\mathbf{h})))} \cdot (64 + 4(8 - W(\mathbf{h})))$$

operations[1]. This approaches to

$$3 \cdot 2^{(64+4(8-W(\mathbf{h})))} \cdot (64 + 4(8 - W(\mathbf{h})))$$

times of one-round Camellia encryption.

2. The global vector of bias $\epsilon_k^{\mathbf{b}|\mathbf{h}}$ for the particular choice of $(\mathbf{b}|\mathbf{h})$ and

$$(K_0^6|K_2^6|K_1^{2,3,5,7,8}|K_3^8|K_4^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h}))$$

is the sum of ϵ_i: $\epsilon_k^{\mathbf{b}|\mathbf{h}} = \sum_i \epsilon_i$.

3. Let $C_\kappa = C_\kappa + (\epsilon_k^{\mathbf{b}|\mathbf{h}}/N)^2$.

We obtain 2^{160} counters for C_κ corresponding to the evaluation of the bias of the linear approximation for the 160-bit subkey guess. The correct subkey is then selected among the candidates with C_κ less than the threshold

$$t = \sigma_0 \cdot z_{1-\beta_0} + \mu_0 = \frac{\sqrt{2l}}{N} \cdot z_{1-\beta_0} + \frac{l}{N} = \frac{\sqrt{2 \cdot 2^{13.16}}}{N} \cdot 2 + \frac{2^{13.16}}{N} = \frac{2^{13.20}}{N}.$$

From the section on data complexity reduction, the number of known plaintext-ciphertext pairs should satisfy the following condition:

$$N \geq \frac{2^{n+0.5}}{\sqrt{l}}(z_{1-\beta_1}\sqrt{1 + \frac{2N}{2^n}} + z_{1-\beta_0}).$$

[1] From [28], the k-level matrix vector product can be computed with 3 times of 2^k-point Fast Fourier Transform.

If we set $\beta_0 = 0.023$ and $\beta_1 = 2^{-98}$, we get $z_{1-\beta_0} = 2$ and $z_{1-\beta_1} = 11.37$. $n = 128, l = 255 \cdot 36 = 2^{13.16}$. As $N = 2^{125.9}$, the above condition can be satisfied.

The memory requirements in the data counting phase and the key counting phase are 2^{152} nibbles and 2^{92} nibbles, respectively. The memory requirements for the global counters C_κ are 2^{160} 256-bit words. Therefore, the whole memory complexity is about 2^{165} bytes.

The time complexity for the data counting phase is $2^{13.16} \cdot 2^{126.3} = 2^{139.46}$ memory accesses. In step (b) of the key counting phase, the time complexity is

$$28 \cdot 2^{72} \cdot 2^{64} \cdot 2^{93.74-4\cdot 2} + 8 \cdot 2^{68} \cdot 2^{60} \cdot 2^{93.74-4} = 2^{226.58}$$

12-round Camellia encryptions. In step (c) of the key counting phase, the time complexity is

$$8 \cdot (2^{68} \cdot 2^{60} \cdot 3 \cdot 92 \cdot 2^{92}) + 28 \cdot (2^{72} \cdot 2^{64} \cdot 3 \cdot 88 \cdot 2^{88})) = 2^{236.89}$$

times of one-round encryptions which is equivalent to $2^{233.30}$ times of 12-round encryptions.

Due to $\beta_1 = 2^{-98}$ and the total number of recovered bits is 160, the number of the remaining subkey values is $2^{-98} \cdot 2^{160} = 2^{62}$.

Then we use another 7-round zero-correlation linear hull with

$$\alpha = (b, 0, 0, b, 0, b, b, b)$$

and

$$\beta = (0, h, h, 0, h, h, 0, h)$$

to proceed the above key recovery attack, and we can recover $K_1^{1,4,6}, K_2^8, K_3^6$ and $K_4^{2,3,8}$, the time complexity and the memory complexity are equal to that in the first attack. Due to $\beta_1 = 2^{-98}$ and the total number of recovered bits is 160, the number of the remaining subkey values is $2^{-98} \cdot 2^{160} = 2^{62}$. Therefore, the total number of the remaining subkey values is about $2^{62} \cdot 2^{62} = 2^{124}$. The right key from the first attack and the second attack should has the same value for the 96 shared subkey bits, so the remaining number of subkey values should be $2^{124} \cdot 2^{-96} = 2^{28}$.

According to the key schedule for Camellia-256, its 256-bit master key is $K = K_L \| K_R$, where K_L and K_R are 128 bits. With K_L and K_R, the key schedule algorithm first calculates K_A and K_B, which is described in Fig.16. F is the round function of Camellia and $C_i (1 \leq i \leq 6)$ are constants used as the subkeys. Then the round subkey $k^{wi} (i = 1, \ldots, 4)$, $k_r (r = 1, \ldots, 24)$ and $kl^j (j = 1, \ldots, 6)$ are derived from rotating K_L, K_R, K_A or K_B.

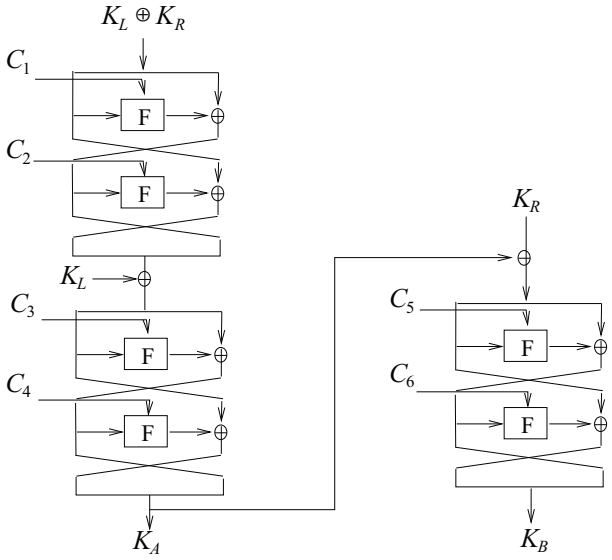


Fig. 16. The Calculation of K_A and K_B

$$K_0 = (K_L \lll 0)_L \oplus (K_B \lll 0)_L, \tag{7}$$

$$K_1 = (K_L \lll 0)_R \oplus (K_B \lll 0)_R, \tag{8}$$

$$K_2 = (K_L \lll 0)_L \oplus (K_R \lll 15)_L, \tag{9}$$

$$K_3 = (K_A \lll 45)_L \oplus (K_B \lll 111)_L, \tag{10}$$

$$K_4 = (K_A \lll 45)_R \oplus (K_B \lll 111)_R. \tag{11}$$

Now we have recovered 64-bit K_0, 64-bit K_1, 64-bit K_4, 16-bit $K_2^{6,8}$ and 16-bit $K_3^{6,8}$, we can recover the master key with the following procedure:

For each of 2^{28} remaining subkey values and for each of 2^{128} values of K_B:

1. Compute K_L from Equation (7) and Equation (8).
2. Compute $(K_A \lll 45)_R$ from Equation (11), and it means that $K_A^{0\sim44,109\sim127}$ has been deduced.

3. Compute $K_A^{85\sim92,101\sim108}$ from Equation (10) and $K_R^{55\sim62,71\sim78}$ from Equation (9).
4. Decrypt from K_B to K_A with Fig. 16, and we can derive 96-bit K_R.
5. For each possible remaining 32-bit value for $K_R^{45\sim54,63\sim70,79\sim84,93\sim100}$, deduce the master key $K = K_L \| K_R$.
6. Using two pairs of plaintext-ciphertext to verify the master key K.

The right guess for K_B and $K_R^{45\sim54,63\sim70,79\sim84,93\sim100}$ can always pass the verification for step 6, but the wrong guess will pass the verification with the probability 2^{-96}. So we can recover the master key with high probability. The time complexity to recover the master key is much less than that to recover 160-bit subkey in the previous, so it can be negligible.

In all, the data complexity is about $2^{125.9}$ known plaintexts, the time complexity is about $2^{234.31}$ 12-round Camellia-256 encryptions and the memory requirements are 2^{165} bytes.

6.4 Attack on 11-round Camellia-192

We use the 7-round zero-correlation linear approximations to attack 11-round Camellia-192, see Fig. 15(b). If we denote $K_0 = k^{w1} \oplus k_1$, $K_1 = k^{w2} \oplus k_2$, $K_2 = k^{w3} \oplus k_{10}$, and $K_3 = k^{w4} \oplus k_{11}$, the following linear approximation can be derived:

$$\begin{aligned} &\mathbf{b} \cdot P_L^6 \oplus \mathbf{h} \cdot C_R^8 \oplus \alpha^T \cdot P_L^{2,3,5,8} \oplus \\ &\beta^T \cdot C_R^{1,4,6,7} \oplus \mathbf{b}^T \cdot S^6[P_R^6 \oplus K_1^6 \oplus F_1^6(P_L^{2,3,5,7,8} \oplus K_0^{2,3,5,7,8})] \oplus \\ &\mathbf{h}^T \cdot S^8[C_L^8 \oplus K_2^8 \oplus F_4^8(C_R^{1,4,5,6,7} \oplus K_3^{1,4,5,6,7})] = 0, \end{aligned} \tag{12}$$

where $\alpha^T = (\mathbf{b}, \mathbf{b}, \mathbf{b}, \mathbf{b})$, $\beta^T = (\mathbf{h}, \mathbf{h}, \mathbf{h}, \mathbf{h})$, $\mathbf{b} \in \mathbb{F}_{2^8}, \mathbf{b} \neq 0$, $\mathbf{h} \in \mathbb{F}_{2^8}, \mathbf{h} \neq 0$.

In order to express Equation (12) as a circulant matrix, some parameters need to be fixed to obtain the following circulant matrix:

$$\begin{aligned} &M(P_L^7 | P_L^{2,3,5,8} \wedge (\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}) | C_R^5 | C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}), \\ &K_0^7 | K_0^{2,3,5,8} \wedge (\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}) | K_3^5 | K_3^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}})) = \\ &\alpha^T \cdot P_L^{2,3,5,8} \oplus \beta^T \cdot C_R^{1,4,6,7} \oplus \mathbf{b}^T \cdot S^6[P_R^6 \oplus K_1^6 \oplus F_1^6(P_L^{2,3,5,7,8} \oplus K_0^{2,3,5,7,8})] \\ &\oplus \mathbf{h}^T \cdot S^8[C_L^8 \oplus K_2^8 \oplus F_4^8(C_R^{1,4,5,6,7} \oplus K_3^{1,4,5,6,7})]. \end{aligned} \tag{13}$$

The attack can be described in the following way:

– Set the global counter C_κ as zero for each of 2^{96} values of

$$\kappa = (K_0^{2,3,5,7,8}|K_1^6|K_2^8|K_3^{1,4,5,6,7}).$$

– For $\mathbf{h} \in S$ and $\mathbf{b} \in S$ with

$$S = \{\mathbf{x} : 1 \leq W(\mathbf{x}) \leq 4, x \in \mathbb{F}_{2^8}\}$$

and

$$|S| = \binom{8}{4} + \binom{8}{3} + \binom{8}{2} + \binom{8}{1} = 162.$$

- Data counting phase:
  1. For N plaintext-ciphertext pairs, extract $16 + 4W(\mathbf{b}) + 4W(\mathbf{h})$ bits value

$$i = (P_L^{2,3,5,8} \wedge (\mathbf{b}|\mathbf{b}|\mathbf{b}|\mathbf{b})|P_R^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8)$$

  and the corresponding $16 + 4(8 - W(\mathbf{b})) + 4(8 - W(\mathbf{h}))$ bits value

$$j = (P_L^7|P_L^{2,3,5,8} \wedge (\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}})|C_R^5|C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}})).$$

  2. Increment the counter j of the vector $\mathbf{x_i}$ according to the parity of

$$\mathbf{b} \cdot P_L^6 \oplus \mathbf{h} \cdot C_R^8.$$

- Key counting phase: For each possible $16 + 4W(\mathbf{b}) + 4W(\mathbf{h})$ bits value k of

$$(K_1^6|K_0^{2,3,5,8} \wedge (\mathbf{b}|\mathbf{b}|\mathbf{b}|\mathbf{b})|K_2^8|K_3^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})) :$$

  1. For each possible $16 + 4W(\mathbf{b}) + 4W(\mathbf{h})$ bits value

$$(P_L^{2,3,5,8} \wedge (\mathbf{b}|\mathbf{b}|\mathbf{b}|\mathbf{b})|P_R^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8) :$$

     (a) Select the corresponding vector of counters $\mathbf{x_i}$
     (b) Compute the first column M_i of the matrix

$$M_i(P_L^7|P_L^{2,3,5,8} \wedge (\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}})|C_R^5|C_R^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}),$$

$$K_0^7|K_0^{2,3,5,8} \wedge (\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}}|\bar{\mathbf{b}})|K_3^8|K_3^{1,4,6,7} \wedge (\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}|\bar{\mathbf{h}}))$$

$$= (-1)^{\alpha^T \cdot P_L^{2,3,5,8} \oplus \beta^T \cdot C_R^{1,4,6,7} \oplus \mathbf{b}^T \cdot S^6[P_R^6 \oplus K_1^6 \oplus F_1^6(P_L^{2,3,5,7,8} \oplus K_0^{2,3,5,7,8})]} \times$$

$$(-1)^{\mathbf{h}^T \cdot S^8[C_L^8 \oplus K_2^8 \oplus F_4^8(C_R^{1,4,5,6,7} \oplus K_3^{1,4,5,6,7})]}.$$

As M_i is $(16 + 4(8 - W(\mathbf{b})) + 4(8 - W(\mathbf{h})))$-level circulant, this information is sufficient to define M completely. This requires

$$2^{16+4(8-W(\mathbf{b}))+4(8-W(\mathbf{h}))}$$

operations which are equivalent to

$$2^{16+4(8-W(\mathbf{b}))+4(8-W(\mathbf{h}))} \cdot (\tfrac{5}{8} + \tfrac{1}{8} + \tfrac{5}{8} + \tfrac{1}{8})/11 =$$

$$2^{16+4(8-W(\mathbf{b}))+4(8-W(\mathbf{h}))} \cdot 1.5/11 = 2^{77.13-4W(\mathbf{b})-4W(\mathbf{h})}$$

11-round encryptions.

(c) The evaluation of the linear approximation for the subset of texts with the corresponding

$$(P_L^{2,3,5,8} \wedge (\mathbf{b}|\mathbf{b}|\mathbf{b}|\mathbf{b})|P_R^6|C_R^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h})|C_L^8)$$

value is given by the matrix vector product $\epsilon_i = M \cdot x_i$. This requires

$$9 \cdot 2^{(16+4(8-W(\mathbf{b}))+4(8-W(\mathbf{h})))} \cdot (16 + 4(8 - W(\mathbf{b})) + 4(8 - W(\mathbf{h})))$$

clock cycles.

2. The global vector of bias $\epsilon_k^{\mathbf{b}|\mathbf{h}}$ for the particular choice of $(\mathbf{b}|\mathbf{h})$ and

$$(K_1^6|K_0^{2,3,5,8} \wedge (\mathbf{b}|\mathbf{b}|\mathbf{b}|\mathbf{b})|K_2^8|K_3^{1,4,6,7} \wedge (\mathbf{h}|\mathbf{h}|\mathbf{h}|\mathbf{h}))$$

is the sum of ϵ_i: $\epsilon_k^{\mathbf{b}|\mathbf{h}} = \sum_i \epsilon_i$.

3. Let $C_\kappa = C_\kappa + (\epsilon_k^{\mathbf{b}|\mathbf{h}}/N)^2$.

We obtain 2^{96} counters for C_κ corresponding to the evaluation of the bias of the linear approximation for the 96-bit subkey guess. The correct subkey is then selected among the candidates with C_κ less than the threshold

$$t = \sigma_0 \cdot Z_{1-\beta_0} + \mu_0 = \frac{\sqrt{2l}}{N} \cdot Z_{1-\beta_0} + \frac{l}{N} = \frac{\sqrt{2 \cdot 2^{14.68}}}{N} \cdot 2 + \frac{2^{14.68}}{N} = \frac{2^{14.69}}{N}.$$

From Theorem 1 in Section of Data Complexity Reduction, the number of known plaintext-ciphertext pairs should satisfy the following condition:

$$N \geq \frac{2^{n+0.5}}{\sqrt{l}}(z_{1-\beta_1}\sqrt{1 + \frac{2N}{2^n}} + z_{1-\beta_0}).$$

If we set $\beta_0 = 0.023$ and $\beta_1 = 2^{-106}$, we get $z_{1-\beta_0} = 2$ and $z_{1-\beta_1} = 11.85$. $n = 128, l = 2^{14.68}$. As $N = 2^{125.1}$, the above condition can be satisfied.

The memory requirements in the data counting phase and the key counting phase are 2^{96} 32-bit words and 2^{72} nibbles, respectively. The memory requirements for the global counters C_κ are 2^{96} 256-bit words. Therefore, the whole memory complexity is about 2^{101} bytes.

The time complexity for the data counting phase is $2^{14.68} \cdot 2^{125.1} = 2^{139.78}$ memory accesses. In step (b) of the key counting phase, the time complexity is about

$$\binom{8}{4} \cdot \binom{8}{4} \cdot 2^{77.13+16+16+4\cdot4+4\cdot4} = 2^{153.39}$$

11-round Camellia encryptions.

In step (c) of the key counting phase, the time complexity is about

$$\binom{8}{4} \cdot \binom{8}{4} \cdot 2^{16+4\cdot4+4\cdot4+16+4\cdot4+4\cdot4+16+32+32-4\cdot4-4\cdot4} \cdot 9 \cdot (16+32+32-4\cdot4-4\cdot4)) = 2^{165.01}$$

clock cycles, which is equivalent to $2^{165.01} \cdot 24/(326 \cdot 11) = 2^{157.79}$ 11-round Camellia encryptions. Therefore, the whole time complexity is about $2^{157.79}$ 11-round Camellia encryptions.

Due to $\beta_1 = 2^{-106}$, the total number of recovered bits is 96, the number of the remaining wrong subkey values is $2^{-106} \cdot 2^{96} = 2^{-10}$.

Then we use other 7-round zero-correlation linear hulls to recover other subkey bits as follows:

1. Use $(b|0|0|b|0|b|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|h|h|0|h|h|0|h)$, to recover $K_0^{1,4,6}$, K_1^8, K_2^6 and $K_3^{2,3,8}$.
2. Use $(b|b|0|0|b|0|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|0|h|h|h|h|h|0)$ to recover K_1^5 and K_2^7.
3. Use $(0|0|b|b|b|b|b|0, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|0|0|h|0|h|h)$ to recover K_1^7 and K_2^5.
4. Use $(0|b|b|b|0|b|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, 0|h|h|h|0|h|h|h)$ to recover K_1^1 and K_2^1.
5. Use $(b|0|b|b|b|0|b|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|0|h|h|h|0|h|h)$ to recover K_1^2 and K_2^2.
6. Use $(b|b|0|b|b|b|0|b, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|0|h|h|h|0|h)$ to recover K_1^3 and K_2^3.
7. Use $(b|b|b|0|b|b|b|0, 0|0|0|0|0|0|0|0) \xrightarrow{7r} (0|0|0|0|0|0|0|0, h|h|h|0|h|h|h|0)$ to recover K_1^4 and K_2^4.

The time complexity with the later zero-correlation linear hull is much less than that with the first zero-correlation linear hull attack because many subkey bits have been recovered in the first attack, so we can ignore it. Now, we have recovered K_0, K_1, K_2 and K_3.

The key schedule for Camellia-192 is similar as that for Camellia-256. The only difference is that the 192-bit master key $K = K_L|K_R$ and $K_{RR} = \overline{K_{RL}}$. According to the key schedule, we can write the following equations:

$$K_0 = (K_L \lll 0)_L \oplus (K_B \lll 0)_L, \tag{14}$$

$$K_1 = (K_L \lll 0)_R \oplus (K_B \lll 0)_R, \tag{15}$$

$$K_2 = (K_L \lll 45)_R \oplus (K_B \lll 111)_L. \tag{16}$$

$$K_3 = (K_A \lll 45)_L \oplus (K_B \lll 111)_R, \tag{17}$$

We will implement the following procedure to recover the 192-bit master key. For each of the 2^{128} values of K_B:

1. Compute K_L from Equation (14) and Equation (15).
2. Verify if K_L and K_B satisfy Equation (16), if so, go to next step.
3. Compute $(K_A \lll 45)_L = K_A^{45\sim108}$ from Equation (17).
4. For all 2^{19} possible values of $K_A^{109\sim127}$ to get K_{AR}:
   (a) Decrypt from K_B to K_{AR} with Fig. 16, and derive K_{RR} to obtain the master key $K = K_L\|\overline{K_{RR}}$.
   (b) Using one pair of plaintext-ciphertext to verify the master key K.

The right guess for K_B and $K_A^{109\sim127}$ can always pass the verification for step 4, but the wrong guess will pass the verification with the probability $2^{19+64-128} = 2^{-45}$. So we can recover the master key with high probability. The time complexity to recover the master key is much less than that to recover 96-bit subkey in the previous, so it can be negligible.

In all, the data complexity is $2^{125.1}$ known plaintexts, the time complexity is about $2^{157.79}$ 11-round Camellia-192 encryptions, and the memory requirements are about 2^{101} bytes.

7 Biclique Cryptanalysis of CLEFIA

7.1 Summary

Again, we concentrate ourselves on the independent biclique approach with at most a single-byte key modification, since this proved to be the way to go with AES so far. Since CLEFIA has MDS matrices and the local SPN structure similar to that of AES, it is reasonable to assume that such biclique attacks will do a good job also for CLEFIA. As in the case of Camellia, we adopt the approach of unlimited data complexity meaning a lower bound on the computational complexity of independent bicliques. It turns out though that the full codebook is never needed for CLEFIA.

The key schedule of CLEFIA is complex compared to that of AES. Like in Camellia, it again relies on an intermediate key L which is computed from the user-supplied key K in a highly nonlinear manner. Now the subkeys in every round (except for the whitening keys that depend on K only) depend on L. The subkeys in rounds 3 and 4 in each 4-round block additionally depend on K. This limits the length of bicliques compared to AES. We have found an efficient key enumeration only for CLEFIA-128 and CLEFIA-256. Here we enumerate the keys in each group with respect to L, since there are adjacent rounds where the computation only depends on L. For CLEFIA-192, not every value of L corresponds to a valid key K. The results reported below for CLEFIA-192 are based on partial matching computational advantage only, that is, MITM without the biclique initial structure. Here we enumerate keys with respect to K.

Our biclique results for CLEFIA are as follows. For the full CLEFIA-128, we report a key recovery with computational complexity $2^{127.7}$ CLEFIA-128 operations, data complexity 2^{64}, negligible memory complexity and success probability 1. For the full CLEFIA-192, there is a MITM key recovery with computational complexity of $2^{191.5}$ CLEFIA-192 executions, negligible memory and data complexities as well as success probability 1. For the full CLEFIA-256, a key recovery requires $2^{255.5}$ CLEFIA-256 executions, data complexity 2^{64}, negligible memory complexity and success probability 1. The low data complexity MITM-only attack on CLEFIA-128 and CLEFIA-256 is also possible which would lead to an increase in the computational complexity and is hence not reported.

7.2 Cost of brute force key recovery: establishing base line

To establish the base line for the subsequent comparisons, we derive the complexity of the brute force key recovery that involves one full computation of the cipher for every key.

Table 6. Summary of our biclique key recoveries for full CLEFIA

cipher	rounds	attack type	data	time	memory
CLEFIA-128	18 (of 18)	biclique MITM	2^{64}CP	$2^{127.7}$	small
CLEFIA-192	22 (of 22)	MITM	2KP	$2^{191.5}$	small
CLEFIA-256	26 (of 26)	biclique MITM	2^{64}CP	$2^{255.5}$	small

For CLEFIA, according to the standard computational model of biclique cryptanalysis, the unit will be again (as for AES and Camellia) the application of an 8-bit S-box. Similarly to AES, this is arguably the most consuming component in terms of implementations. As for AES and as opposed to Camellia, 4x4-byte MDS diffusion matrices are employed in CLEFIA. However, firstly, it almost always goes with S-box computations (no separate account needed) and, secondly, the contribution of the matrix-vector multiplication is rather limited compared to 4 8-bit S-boxes in most implementations.

Table 7. Cost of brute force for CLEFIA

cipher	data transform (F-functions)	key schedule (F-functions)	total (F-functions)	total (S-boxes)
CLEFIA-128	18R=36F	12R=24F	60F	240
CLEFIA-192	22R=44F	10R=40F	84F	336
CLEFIA-256	26R=52F	10R=40F	92F	368

As shown in Table 7, one CLEFIA-128 computation needs 240 S-box applications, since each F-function includes 4 S-box applications and there are equivalently 60 F-functions rounds in the full CLEFIA-128. Similar derivation holds for CLEFIA-192 and CLEFIA-256 resulting in 336 and 368 S-boxes, respectively, in total. The difference for CLEFIA-192 and CLEFIA-256 is though that instead of 12 rounds of a 4-line type-II GFN for CLEFIA-128, 10 rounds of a 8-line type-II GFN are used, involving 4 F-functions per round instead of 2 F-functions.

7.3 Constructing bicliques over 2 rounds for CLEFIA-128 and CLEFIA-256

Unlike for Camellia, we will construct an independent biclique using two forward computations. In CLEFIA-128 and CLEFIA-256, we enumerate keys in a key group with respect to L. We place the key modification in the subkeys in round

1. Subkeys RK_0 and RK_1 are just the first 64 bits of L added with a constant string of bits. We construct bicliques as specified in Table 8.

The length of bicliques is limited by the key schedule. In a block of 4 rounds, only rounds 1 and 2 depend on L only. Rounds 3 and 4 of a 4-round block, depend on both L and K. Given the complex relation between L and K, this modifies the computation in rounds 3 and 4 completely even if only a slight change has been made to L.

Table 8. 2-round bicliques and key modification for CLEFIA

cipher	type	key modification	subkey	round number
CLEFIA-128	forward 1	one byte	RK_0	1
	forward 2	one byte	RK_1	1
CLEFIA-192	-	one byte	WK_0	whitening
CLEFIA-256	forward 1	one byte	RK_0	1
	forward 2	one byte	RK_1	1

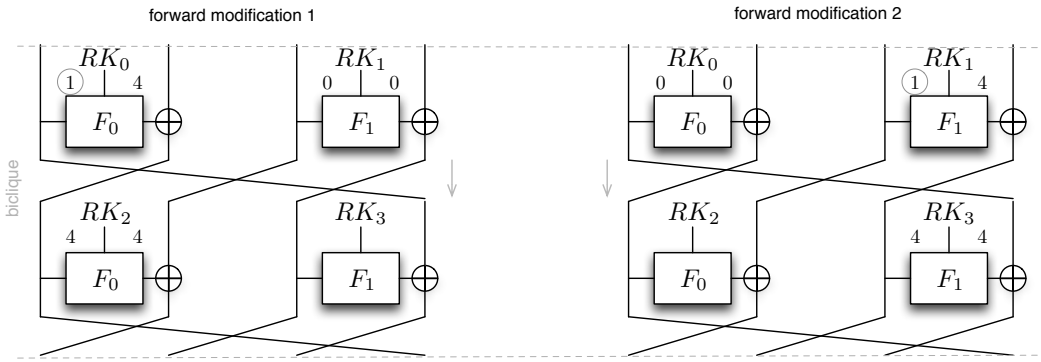


Fig. 17. Building a biclique over 2 rounds of CLEFIA-192 and CLEFIA-256: Numbers indicate the number of active bytes in the F-function right after the subkey addition and right after the diffusion layer. Numbers in grey circles indicate the spots of key difference injections

The key is modified only in the forward direction: one forward modification is in the left-hand F-function and the other one is the right-hand F-function.

Since we modify a byte, the dimension of the biclique is 8. The actual biclique construction is outlined in Figure 17.

In the left-hand modification, the injection byte in RK_0 activates the entire output of F_0 in round 1 which in turn activates the input and output of F_0 in round 2. In the right-hand modification, the injection byte in RK_1 activates the entire output of F_1 in round 1 and, thus, the input to and output of F_1 in round 2.

It is easy to see that this is an independent 2-round biclique of dimension 8: The forward key modifications are such that the propagations in both of them are entirely independent for two rounds.

7.4 Key enumeration for CLEFIA-192

In CLEFIA-192, the keys are enumerated with respect to K. We can be more precise and say that we modify one byte in K_L which corresponds to a byte modification in one of the two prewhitening keys WK_0 or WK_1. The exact position of this modification is rather not important since the operation of the first round depends on L that in turn depends on K in a complex way. This will make the computation starting from the first round as for a completely fresh key.

Note that we cannot efficiently construct a nontrivial independent biclique for CLEFIA-192. This is due to the fact that we want to enumerate the keys with respect to L since there are rounds in CLEFIA data path that just depend on L. However, not every value of L to corresponds to a valid value of K (user-supplied key) in CLEFIA-192. This is because of the redundancy that is introduced to K before it is passed to the 10-round 8-line type-II GFN for the derivation of L: The last 64 bits of the input depend on other 64 bits in K.

7.5 Partial matching over 5 rounds

The construction of the CLEFIA data transform is such that is allows for a partial matching over 5 rounds. These matching rounds can be placed in the middle of the cipher. An illustration is provided in Figure 18. As opposed to biclique construction, partial matching can be performed for all CLEFIA versions.

We match the middle of the 5 rounds in one byte of a 4-byte quarter state. It suffices to recompute 16 S-boxes instead of 40 that would be needed for matching on the full state, resulting in a saving of 24 S-boxes. Note that no recomputation is needed in round 3 of the 5-round block. Only 4 S-boxes need to be recomputed in the other 4 rounds.

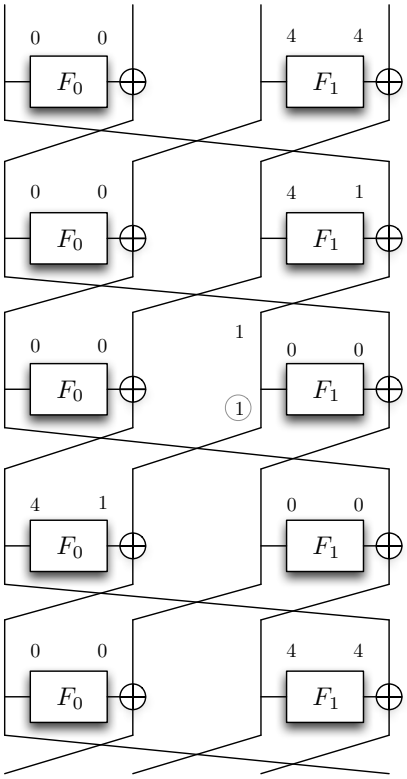


Fig. 18. Partial matching over 5 rounds of CLEFIA: Numbers indicate the number of bytes to be recomputed in the corresponding lines. The number in grey circle indicates the byte of matching

7.6 Recomputing the key schedule

Though the key recovery does not specify any further savings in the data transform of CLEFIA and all S-boxes in the remaining rounds have to be recomputed for every key, some more savings are possible in the key schedule.

In CLEFIA-128, the key schedule consists of 12 rounds of the data transform applies to K with constants as round keys to obtain L. Since we enumerate keys with respect to L, we need to apply decryption for every modification of L. Since we modify in two distinct 32-bit subkeys, this will have effect on two lines of input to the inverse 4-line GFN. The first 3 of 12 rounds do not have to be recomputed entirely. One S-box is recomputed in round 1, 4 S-boxes in round 2 and 5 S-boxes in round 3 of the inverse GFN. This results in a saving of 10 S-boxes.

In CLEFIA-192, keys are enumerated with respect to K. A single byte is modified there. So we have to check how many computations are actually needed in the forward 8-line 10-round type-II GFN if only a byte of its input gets modified. In round 1, no S-boxes need recomputation. Only 1 S-box is recomputed in round 2, followed by 4 S-boxes in round 3 and 5 in round 4. Similarly, we can see that even in round 8 only 13 out of 16 S-boxes need recomputation. In rounds 9 and 10, all 16 S-boxes are recomputed. In total, this yields 84 S-boxes instead of 160 which gives a saving of 76 S-boxes. Similarly, for CLEFIA-256, we obtain a saving of 60 S-boxes.

7.7 Complexity

In total, we have saved 16 S-box computations due to the construction of the biclique for CLEFIA-128 and CLEFIA-256 and 0 S-boxes for CLEFIA-192, 24 S-box computations due to the partial matching, as well as another 10, 76 and 60 S-box computations due to the optimized recomputation in the key schedule, in CLEFIA-128, CLEFIA-192, and CLEFIA-256, respectively. This results in coincidentally round numbers of 50, 100 and 100 S-boxes saved per key tested, correspondingly.

Thus, the computational complexity for CLEFIA-128 is

$$\frac{240 - 24 - 16 - 10}{240} 2^{128} \approx 2^{127.7}.$$

For CLEFIA-256, we have time complexity of

$$\frac{368 - 24 - 16 - 60}{368} 2^{256} \approx 2^{255.5}.$$

For CLEFIA-192, we need

$$\frac{336 - 24 - 76}{336} 2^{256} \approx 2^{191.5}$$

computations without the biclique saving. Note that we do not take into account the complexity of biclique construction since the dimension of the biclique is large enough to make it negligible. Memory complexity in the biclique construction and recomputations within the key group is negligible because of the limited biclique dimension.

The data complexity for CLEFIA-192 is just the unicity distance which is 2 known- plaintexts here, due to the absence of any initial structure. For CLEFIA-128 and CLEFIA-256, we start with a constant value for all keys after prewhitening and before round 1 where the forward modifications are applied. So in the

backward recomputation, the only parts of the plaintext affected by the modifications are the 64 bits whitened by the prewhitening which depends on K. So the data complexity is 2^{64} chosen plaintexts.

8 Zero Correlation Linear Cryptanalysis of CLEFIA

8.1 Summary

For many ciphers, especially for those with XOR addition of the round subkeys, the partial encryption and decryption in the basic zero correlation cryptanalysis is performed many times for the same value. The technique is to precompute all required intermediate values for all key bits involved into partial encryption/decryption and to store them instead of recomputing again once needed. This technique can be seen as equivalent to the discrete Fast Fourier Transform and significantly reduces the computational complexity of the zero correlation linear cryptanalysis at the expense of introducing some memory complexity.

The data transform of CLEFIA (type-II generalized Feistel network with 4 lines) exhibits a set of about 2^{32} 9-round zero-correlation linear approximations of the form $(0, 0, 0, a) \rightarrow (a, 0, 0, 0)$. So we have a budget of about 2^{32} approximations for the data complexity reduction with the multidimensional zero-correlation distinguisher. We can attain either a data complexity reduction of factor about 2^{16} or trade it off for time complexity reduction and/or more attacked rounds when combined with the discrete FFT.

Our zero correlation results for CLEFIA are the following. For the purpose of evaluation, we only report the attacks on the highest numbers of rounds we have found, even if they require the full codebook. As a matter of fact, we can attack less rounds with much lower complexities. First, as regards CLEFIA-192, we can attack 13 rounds of it using $2^{125.8}$ data and $2^{159.56}$ operations equivalent to 13 rounds of CLEFIA-192, and 2^{104} blocks of memory. Second, we can attack 14 rounds of CLEFIA-256 with 2^{128} texts, 2^{222} time, and 2^{136} blocks of memory.

Table 9. Summary of our zero correlation attacks on round-reduced CLEFIA

cipher	rounds	attack type	data	time	memory
CLEFIA-192	13	zero correlation	$2^{125.8}$	$2^{159.56}$	2^{104}
CLEFIA-256	14	zero correlation	2^{128}	2^{222}	2^{136}

8.2 9-round zero-correlation linear approximations for CLEFIA and distiguisher

In this section, we will give some zero-correlation linear hulls for 9-rounds CLEFIA. We find $(\mathbf{a}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \rightarrow (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{a})$, with input select pattern $(\mathbf{a}, \mathbf{0}, \mathbf{0}, \mathbf{0})$ and output select pattern $(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{a})$,a s described in the figure. We can use the linear hulls to attack more rounds block cipher.

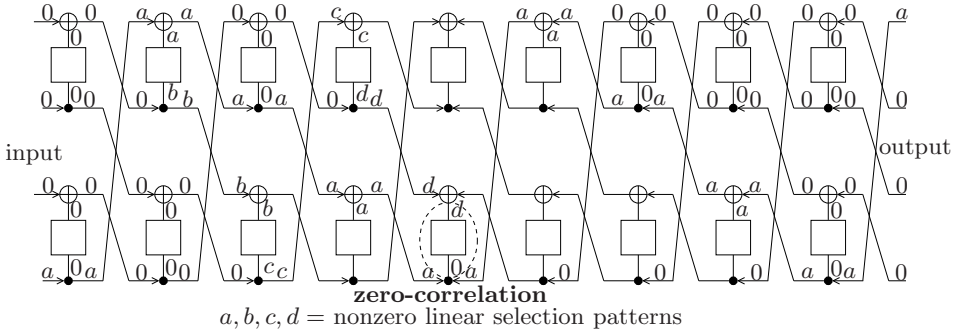


Fig. 19. 9-round zero-correlation linear approximation for CLEFIA-type ciphers (type-II generalized Feistel network) with balanced F-functions

The original zero-correlation linear analysis used only a few linear approximations and had a high data complexity close to 2^{128} for CLEFIA. That is to say, we should use all the plaintext-ciphertext pairs. If we want to reduce the data complexity we should use more linear hulls.

We will be using the following technique to reduce the data complexity of our attacks:

– Distinguishing between two normal distributions
  Consider two normal distributions, which can be seen as randomly drawn permutation and real block cipher respectively: $N(\mu_0, \sigma_0)$ with mean μ_0 and standard deviation σ_0, and $N(\mu_1, \sigma_1)$ with mean μ_1 and standard deviation σ_1. Without loss of generality, assume that $\mu_0 < \mu_1$. A sample s is drawn from either $N(\mu_0, \sigma_0)$ or $N(\mu_1, \sigma_1)$. It has to be decided if this sample is from $N(\mu_0, \sigma_0)$ or from $N(\mu_1, \sigma_1)$. The text is performed by comparing the value s to some threshold value t. If $s \leq t$, the test returns "$s \in N(\mu_0, \sigma_0)$", otherwise, if $s > t$, the test returns "$s \in N(\mu_1, \sigma_1)$". There will be error probabilities of two types:

$$\beta_0 = Pr"s \in N(\mu_1, \sigma_1)"|s \in N(\mu_0, \sigma_0)$$
$$\beta_0 = Pr"s \in N(\mu_1, \sigma_1)"|s \in N(\mu_0, \sigma_0)$$

Here conditions are given on $\mu_0, \mu_1, \sigma_0, \sigma_1$ such that the error probabilities are not higher than β_0 and β_1.

For the test to have error probabilities of at most β_0 and β_1, the parameters of the normal distributions $N(\mu_0, \sigma_0)$ and $N(\mu_1, \sigma_1)$ with $\mu_0 < \mu_1$ have to be such that

$$\frac{z_{1-\beta_1} + z_{1-\beta_0}}{\mu_1 - \mu_0} \leq 1.$$

– Known plaintext distinguisher for many zero-correlation linear hulls.
Let the adversary be given N known plaintext-ciphertext pairs and ℓ zero-correlation linear hulls for an n-bit block cipher with a fixed secret key. The adversary aims to distinguish between this cipher and a randomly drawn permutation.
The procedure is as follows. For each of the ℓ given linear hulls, the adversary computes the number T_i of times the linear hull is fulfilled on N plaintexts, $i \in 1 \cdots \ell$. Each T_i suggests an empirical correlation value $C_i = 2\frac{T_i}{N} - 1$. Then the adversary evaluates the statistic:

$$\sum_{i=1}^{\ell}(C_i^2) = \sum_{i=1}^{\ell}(2\frac{T_i}{N} - 1)^2 \tag{18}$$

It is expected that for the cipher with ℓ known zero-correlation linear hulls, the value of statistic (1) will be lower than for random ℓ linear approximations of a random drawn permutation. In a key-recovery setting, the right key will result in statistic (1) being among the lowest values for all candidate keys if ℓ is high enough.
In the sequel, we treat this more formally.
– Data complexity for the distinguisher
For ℓ zero-correlation linear hulls,each of the values C_i in (1) will be approximately distributed as $N(0, 1/\sqrt{N})$. For a random permutation and a random linear approximation, the value of C_i will be approximately distributed as $N(b_i, \frac{1}{\sqrt{N}})$, where b_i is the exact value of the correlation which is itself distributed as $N(0, 2^{-n/2})$.
Now we will derive the distribution of statistic (1) in these two cases.

1. Cipher: zero-correlation linear hulls In case we deal with ℓ zero-correlation linear hulls:

$$\sum_{i=1}^{\ell} C_i^2 \sim \sum_{i=1}^{\ell} N^2(0, 1/\sqrt{N}) = 1/N \sum_{i=1}^{\ell} N^2(0, 1) = 1/N\chi_\ell^2$$

Where χ^2_ℓ is the χ^2-distribution with ℓ degree of freedom which has mean ℓ and standard deviation $\sqrt{2\ell}$. For sufficiently large ℓ, χ^2_ℓ converges to the normal distribution. That is, χ^2_ℓ approximately follows $N(\ell, \sqrt{2\ell})$.and:

$$\sum_{i=1}^{\ell} C_i^2 \sim 1/N\chi^2_\ell \approx 1/NN(\ell, \sqrt{2\ell}) = N(\tfrac{\ell}{N}, \tfrac{\sqrt{2\ell}}{N}) \tag{19}$$

Consider ℓ nontrivial zero-correlation linear hulls for a block cipher with a fixed key. If N is the number of plaintext-ciphertext pairs, T_i is the number of times such a linear hull is fulfilled for $i = 1 \cdots \ell$, and ℓ is high enough, then the following approximate distribution holds:

$$\sum_{i=1}^{\ell} (2\frac{T_i}{N} - 1)^2 \sim N(\frac{\ell}{N}, \frac{\sqrt{2\ell}}{N})$$

2. Random permutation: random linear approximations In case we deal with a randomly drawn permutation, the ℓ given linear hulls will correspond to a set of random linear approximations. Thus, as mentioned above:

$$\sum_{i=1}^{\ell} C_i^2 \sim \sum_{i=1}^{\ell} N^2(b_i, 1/\sqrt{N}), \text{ where } b_i \sim N(0, 2^{-n/2})$$

Consider ℓ random nontrivial linear approximations for a randomly drawn permutation. If N is the number of plaintext-ciphertext pairs, T_i is the number of times such a linear hull is fulfilled for $i = 1 \cdots \ell$, and ℓ is high enough, then the following approximate distribution holds:

$$\sum_{i=1}^{\ell} (2\frac{T_i}{N} - 1)^2 \sim N(\frac{\ell}{N} + \frac{\ell}{2^n}, \frac{1}{N}\sqrt{2(\ell + 2\ell\frac{N}{2^n})})$$

3. Combining what has been said, one obtains the condition:

$$\frac{z_{1-\beta_0}\frac{1}{N}\sqrt{2(\ell + 2\ell\frac{N}{2^N})} + z_{1-\beta_1}\frac{1}{N}\sqrt{2\ell}}{(\frac{\ell}{N} + \frac{\ell}{2^n}) - \frac{\ell}{N}} \leq 1$$

For an n-bit block cipher, let ℓ nontrivial zero-correlation linear hulls be given. Then, to distinguish it from a randomly drawn permutation with probability β_1 of false positives and probability β_0 of false negatives, a number N of known plaintext-ciphertext pairs is sufficient if the following condition is fulfilled:

$$\frac{2^{n+0.5}}{N\sqrt{\ell}}(z_{1-\beta_0}\sqrt{1 + \frac{2N}{2^n}} + z_{1-\beta_1}) \leq 1$$

The success probability of an attack is defined by the probability β_0 of false negatives. The probability β_1 of false positives determines the number of surviving key candidates and, thus, the complexity of the key recovery. Making $\beta_0 \approx 0.023$ and $\beta_1 \approx 2^{-50.5}$. Thus the following statement holds:

For probability $\beta_0 \approx 0.023$ of false negatives, probability $\beta_1 \approx 2^{-50.5}$ of false positives, and $N \leq 2^{n-1}$, the condition on the data complexity is:

$$N \geq \frac{2^{n+3.94}}{\sqrt{\ell}}.$$

8.3 Attack on 13-round CLEFIA-192

We take advantage of a large number of 9-round zero-correlation linear approximations of $(\mathbf{a}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \rightarrow (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{a})$ kind to attack 13-round CLEFIA-192, the approximation covers round 3 to 11, and has 32 active bits in the first and last round. Now we find out $392 * 32$ different linear hull of this type and we also make sure all α have the following property: when they go through the inverse of the M_1 they have only one active byte, but when they go through the inverse of the M_0 they have 4 active bytes. This kinds of α can be made sure that have the lowest time complexity.

Let us denote by $\mathbf{E} = (\mathbf{E_0}, \mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3})$ the value of the encryption state at the beginning of the third round, and by $\mathbf{D} = (\mathbf{D_0}, \mathbf{D_1}, \mathbf{D_2}, \mathbf{D_3})$ the value of encryption state at the end of the 11 round. All the possible 9-round zero-correlation linear approximation used can be written as: $\mathbf{a}^T \cdot \mathbf{E_0} \oplus \mathbf{a}^T \cdot \mathbf{D_3}$. In order to evaluate the bias of this linear approximations, we have to perform a partial encryption of two first rounds and a partial decryption of the two last rounds.

According to the structure of CLEFIA, we can get the following relations:

$$\begin{aligned} &\mathbf{a}^T \cdot \mathbf{E_0} = \\ &\mathbf{a}^T \cdot \mathbf{P_2} \oplus \mathbf{a}^T \cdot \mathbf{F_0}(\mathbf{RK_2} \oplus \mathbf{WK_0} \oplus \mathbf{P_1} \oplus \mathbf{F_0}(\mathbf{RK_0} \oplus \mathbf{P_0})) \end{aligned} \tag{20}$$

$$\begin{aligned} &\mathbf{a}^T \cdot \mathbf{D_3} = \\ &\mathbf{a}^T \cdot \mathbf{C_2} \oplus \mathbf{a}^T \cdot \mathbf{F_1}(\mathbf{RK_{2r-3}} \oplus \mathbf{WK_2} \oplus \mathbf{C_1} \oplus \mathbf{F_0}(\mathbf{RK_{2r-2}} \oplus \mathbf{C_0})) \end{aligned} \tag{21}$$

For compactness, we will be using the following notation:

$$\mathbf{K_0} = \mathbf{RK_0}$$

$$\mathbf{K_1} = \mathbf{RK_2} \oplus \mathbf{WK_0}$$

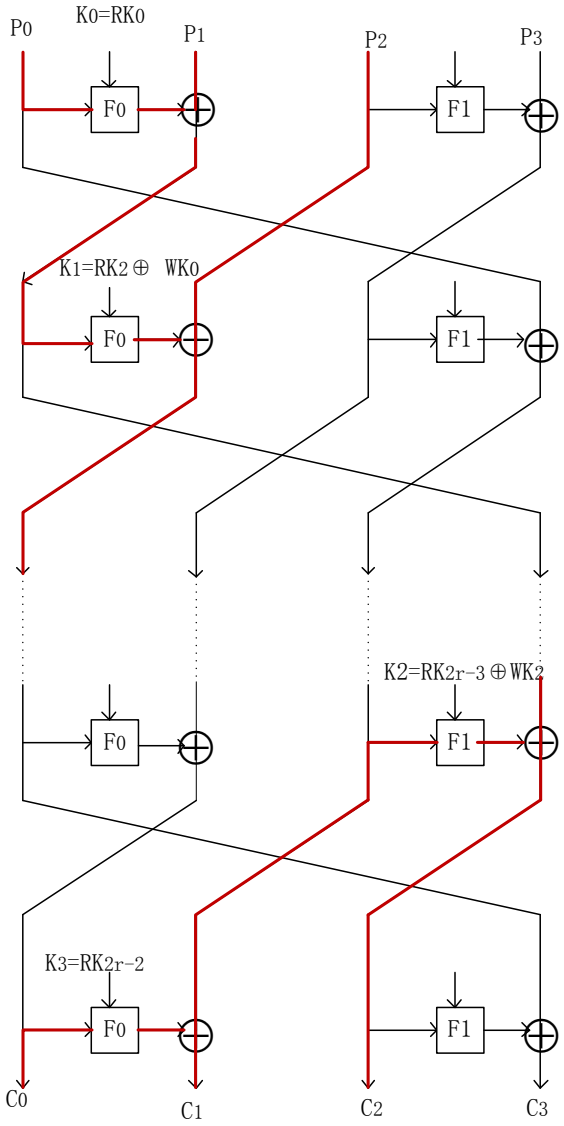


Fig. 20. Zero correlation linear attack on 13-round CLEFIA-192

$$\mathbf{K_2} = \mathbf{RK_{2r-3}} \oplus \mathbf{WK_2}$$

$$\mathbf{K_3} = \mathbf{RK_{2r-2}}$$

the linear approximation can be written as:

$$\begin{array}{l} \mathbf{a}^T \cdot (\mathbf{P_2} \oplus \mathbf{C_2}) \oplus \mathbf{a}^T \cdot \mathbf{F_0}(\mathbf{K_1} \oplus \mathbf{P_1} \oplus \mathbf{F_0}(\mathbf{K_0} \oplus \mathbf{P_0})) \\ \oplus \mathbf{a}^T \cdot \mathbf{F_1}(\mathbf{K_2} \oplus \mathbf{C_1} \oplus \mathbf{F_0}(\mathbf{K_3} \oplus \mathbf{C_0})) \end{array} \tag{22}$$

We want to use FFT methods to reduce the time complexity, which the linear approximation is required to be expressed as a circulant matrix. We can observe that neither $\mathbf{a}^T \cdot \mathbf{F_0}(\mathbf{K_1} \oplus \mathbf{P_1} \oplus \mathbf{F_0}(\mathbf{K_0} \oplus \mathbf{P_0})$ nor $\mathbf{a}^T \cdot \mathbf{F_1}(\mathbf{K_2} \oplus \mathbf{C_1} \oplus \mathbf{F_0}(\mathbf{K_3} \oplus \mathbf{C_0}))$ can be expressed as a circulant matrix because K_1 and K_2 need to be guessed and moreover, P_1 and C_1 need to be traversed. However, by guessing of fixing these parameters, it is possible to decompose the expression in smaller parts which involve the following circulant matrix:

$$\begin{array}{l} M(P_0|C_0, K_0|K_3) = \\ \mathbf{a}^T \cdot \mathbf{F_0}(\mathbf{K_1} \oplus \mathbf{P_1} \oplus \mathbf{F_0}(\mathbf{K_0} \oplus \mathbf{P_0})) \oplus \mathbf{a}^T \cdot \mathbf{F_1}(\mathbf{K_2} \oplus \mathbf{C_1} \oplus \mathbf{F_0}(\mathbf{K_3} \oplus \mathbf{C_0})). \end{array} \tag{23}$$

The attack works as follows:

- Data counting phase:
  1. For each of the 2^{40} value i of $(\mathbf{P_1}, \mathbf{C_1})$, define vector x_i of 2^{40} counters.
  2. For each of the plaintext-ciphertexts $(\mathbf{P}, \mathbf{C})$, extract the 40 bits value $\mathbf{i} = (\mathbf{P_1}, \mathbf{C_1})$ and the 64-bit value $j = (\mathbf{P_0}, \mathbf{C_0})$
  3. Increment or decrement the counter j of the vector x_i according to the parity of $\mathbf{a}^T \cdot (\mathbf{P_2} \oplus \mathbf{C_2})$.
- Key counting phase: For each possible 40-bit value k of $(K_1|K_2)$:
  1. For each possible 40-bit value of $(\mathbf{P_1}, \mathbf{C_1})$:
     (a) Select the corresponding vector of counters $\mathbf{x_i}$
     (b) Compute the first column M_i of the matrix

$$\begin{array}{l} M_i(P_0|C_0, K_0|K_3) = \\ (-1)^{\mathbf{a}^T \cdot \mathbf{F_0}(\mathbf{K_1} \oplus \mathbf{P_1} \oplus \mathbf{F_0}(\mathbf{K_0} \oplus \mathbf{P_0})) \oplus \mathbf{a}^T \cdot \mathbf{F_1}(\mathbf{K_2} \oplus \mathbf{C_1} \oplus \mathbf{F_0}(\mathbf{K_3} \oplus \mathbf{C_0}))}. \end{array}$$

     As M_i is 64-level circulant, this information is sufficient to define M completely (requires 2^{64} operations).
     (c) The evaluation of the linear approximation for the subset of texts with the corresponding $(P_1|C_1)$ value is given by the matrix vector product $\epsilon_i = M \cdot x_i$ (requires $64 \cdot 2^{64}$ operations).
  2. The global vector of bias ϵ_k for the particular choice of $(K_1|K_2)$ is the sum of ϵ_i: $\epsilon_k = \sum_i \epsilon_i$.

At the end of the key counting phase, we obtain 2^{40} vectors ϵ_k of size 2^{64} corresponding to the evaluation of the linear approximation of the bias of the linear approximation for the 104-bit key guess $(\mathbf{K_0}, \mathbf{K_1}, \mathbf{K_2}, \mathbf{K_3})$. The correct key is then selected among the candidates with a corresponding counter equal to 0. We want to reduce the data complexity down to $2^{125.8}$ from 2^{128}, we choose $2^{13.61}$ different α to evaluate the correlation. The data counting phase has a memory complexity of 2^{104} counters since we must store 2^{40} vectors of 2^{64} counters. Its time complexity is the number of plaintext-ciphertext pairs $N = 2^{125.8}$. The key counting phase is dominated by the matrix vector product, which is performed for every guess of $K_1|K_2$ and every choice of $P_1|C_1$. The time complexity of this phase is thus $2^{13.61} \cdot 2^{40} \cdot 2^{40} \cdot 64 \cdot 2^{64} \cdot 9 = 2^{166.78}$ clock cycles. CLEFIA-192 encryption need 15.8 cycles for a byte, then $15.8 \cdot 16$ cycles for 16 bytes. We also know that CLEFIA-192 is 22 rounds, then 13 rounds CLEFIA here need $252.8/22 \cdot 13 = 149$ cycles. Then the time complexity is: $2^{166.78}/149 = 2^{159.56}$.
The output linear pattern to evaluate becomes:

$$\mathbf{b}^T \cdot \mathbf{D}' = \mathbf{b}^T \cdot (\mathbf{M_1^{-1}}(\mathbf{C_2}) \oplus \mathbf{S_1}(\mathbf{K_2} \oplus \mathbf{C_1} \oplus \mathbf{F0}(\mathbf{K_3} \oplus \mathbf{C_0}))) \tag{24}$$

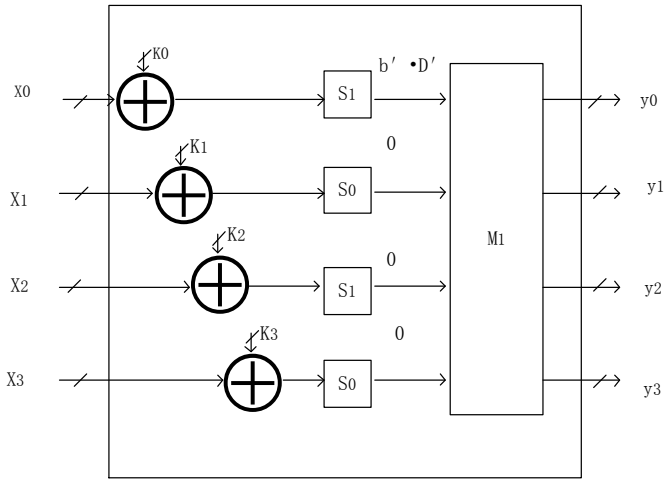


Fig. 21. Selection pattern before M_1

8.4 Attack on 14-round CLEFIA-256

We also use the 9-round zero-correlation linear approximation to attack 14-round CLEFIA-256. This approximation covers round 4 to 12, and has 32 active bits in the first and last round. Let us denote by $E = (\mathbf{E_0}, \mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3})$, the value of the encryption state at the beginning of the fourth round, and by $D = (\mathbf{D_0}, \mathbf{D_1}, \mathbf{D_2}, \mathbf{D_3})$ the value of the encryption state at the end of 12 round. In order to evaluate the bias of this approximations, we perform a partial encryption of three first rounds and a partial decryption of the two last rounds, then the 9-round zero-correlation linear approximation can by written as: $\mathbf{a}^T \cdot E_0 \oplus \mathbf{a}^T \cdot D_3$.

The relations can be derived:

$$\begin{array}{l} \mathbf{a^T} \cdot E_0 = \\ \mathbf{a}^T \cdot (P_3 \oplus F_1(k_1 \oplus P_2) \oplus F_0(k_3 \oplus P_2 \oplus F_0(k_2 \oplus P_1 \oplus F_0(k_0 \oplus P_0)))) \end{array} \tag{25}$$

$$\mathbf{a^T} \cdot D_3 = \mathbf{a}^T \cdot (C_2 \oplus F_1(k_4 \oplus C_1 \oplus F_0(k_5 \oplus C_0))) \tag{26}$$

We have:

$$\begin{array}{l} \mathbf{a}^T \cdot E_0 \oplus \\ \mathbf{a}^T \cdot D_3 = \\ \mathbf{a}^T \cdot (P_3 \oplus C_2) \oplus \\ \mathbf{a}^T \cdot (F_1(k_1 \oplus P_2) \oplus F_0(k_3 \oplus (P_2) \oplus F_0(k_2 \oplus F_0(k_0 \oplus P_0)))) \oplus \\ \mathbf{a}^T \cdot F_1(k_4 \oplus C_1 \oplus F_0(k_5 \oplus C_0)) \end{array} \tag{27}$$

Guess the key bits $(\mathbf{k_1}, \mathbf{k_2}, \mathbf{k_3}, \mathbf{k_4})$, and fixing the parameters $(\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$, it is possible to decompose the expression in smaller parts which involve the following circulant matrix:

$$\begin{array}{l} M(P_0|C_0, k_0|k_5) = \\ \\ \mathbf{a}^T \cdot (F_1 \cdot (k_1 \oplus P_2) \oplus F_0(k_3 \oplus P_2 \oplus F_0(k_2 \oplus P_1 \oplus F_0(k_0 \oplus P_0)))) \oplus \\ \mathbf{a}^T \cdot F_1(k_4 \oplus C_1 \oplus F_0(k_5 \oplus C_0)). \end{array} \tag{28}$$

The attack proceeds as follows:

– Data counting phase:
  1. For each of the 2^{72} values i of $(\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$, define vector x_i of 2^{72} counters.
  2. For each of the plaintext-ciphertext $(\mathbf{P}, \mathbf{C})$, extract the 72 bits values $\mathbf{i} = (\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$ and the 64 bits value $j = (\mathbf{P_0}, \mathbf{C_0})$.
  3. Increment or decrement the counter j of the vector x_i according to the parity of $\mathbf{a}^T \cdot (\mathbf{P_3} \oplus \mathbf{C_2})$.

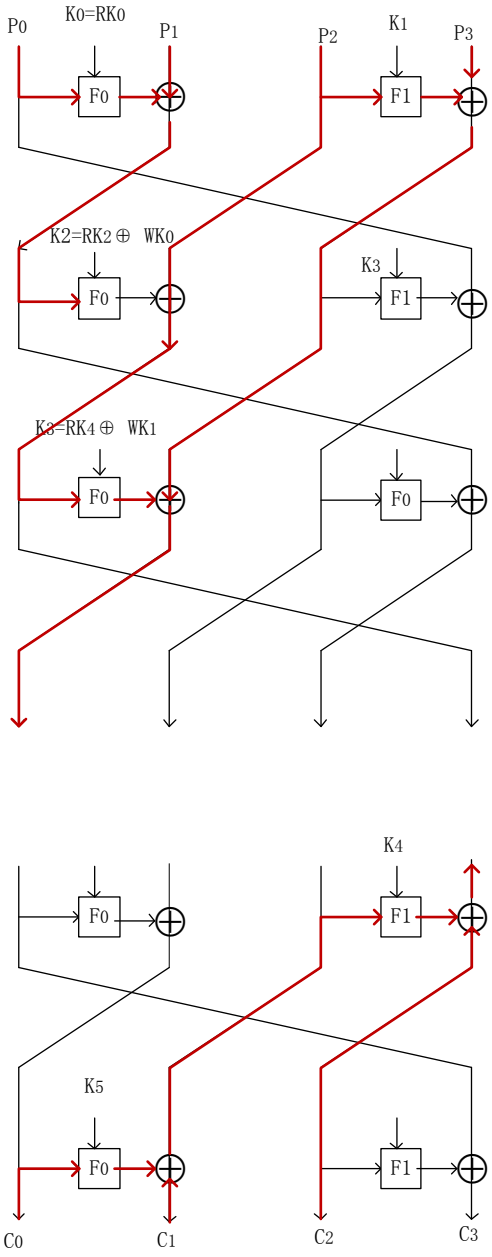


Fig. 22. Zero correlation linear attack on 14-round CLEFIA-256

– Key counting phase: for each possible 80-bit value k of $(\mathbf{k_1}, \mathbf{k_2}, \mathbf{k_3}, \mathbf{k_4})$:
  1. For each possible 72-bit value i of $(\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$:
     (a) Select the corresponding vector of counters $\mathbf{x_i}$
     (b) Compute the first column M_i of the matrix

$$M_i(P_0|C_0, k_0|k_5) =$$

$$(-1)^{\mathbf{a}^T \cdot (F_1 \cdot (k_1 \oplus P_2) \oplus F_0(k_3 \oplus P_2 \oplus F_0(k_2 \oplus P_1 \oplus F_0(k_0 \oplus P_0)))) \oplus \mathbf{a}^T \cdot F_1(k_4 \oplus C_1 \oplus F_0(k_5 \oplus C_0))}.$$

     As M_i is 64-circulant, this information is sufficient to define M completely (requires 2^{64} operations).
     (c) The evaluation of the linear approximation for the subset of texts with the corresponding $(\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$ value is given by the matrix vector product $\epsilon_i = M \cdot x_i$ (requires $64 \cdot 2^{64}$ clock cycles.)
  2. The global vector of bias ϵ_k for the particular choice of $(\mathbf{k_1}, \mathbf{k_2}, \mathbf{k_3}, \mathbf{k_4})$ is the sum of ϵ_i: $\epsilon_k = \sum_i \epsilon_i$.

At the end of the key counting phase, we obtain 2^{80} vectors ϵ_k of size of 2^{64} corresponding to the evaluation of the bias of the linear approximation for the 144-bit subkey guess. The correct subkey is then selected among the candidates with a corresponding counter equal to 0. The data counting phase has a memory complexity of 2^{136} counters since we must store 2^{72} vectors of 2^{64} counters, its time complexity is the number of plaintext-ciphertext pairs $\mathbf{N} = 2^{128}$. The key counting phase is dominated by the matrix vector product. which is performed for every guess of $(\mathbf{k_1}, \mathbf{k_2}, \mathbf{k_3}, \mathbf{k_4})$ and every choice of $(\mathbf{P_1}, \mathbf{P_2}, \mathbf{C_1})$, the time complexity of this phase is thus $2^{80} \cdot 2^{72} \cdot 64 \cdot 2^{64} = 2^{222}$ operations.

9 Biclique and MITM Cryptanalysis of SC2000

9.1 Summary

The computational advantage over brute force that can be gained with independent biclique key recovery usually originates from two major sources. First, if a biclique of a significant length can be built, the recomputation cost of the states at the boundaries of the biclique for each key covered by the biclique is relatively small. Second, if the structure of the cipher in question allows one to perform partial matching over a considerable number of rounds (which mainly depends on how fast the diffusion in the data transformation is), one obtains an additional advantage in computational complexity.

The key enumeration in biclique key recovery is usually done in some suitable round subkey or several round subkeys (for forward and backward key modification). The attacker's hope is then that some local change in the subkey(s) will only cause local modifications in the internal state (block of data being operated with) for a significant number of rounds in both forward and backward directions. Another important issue to make the key recovery successful is to ensure the near-to-full coverage of the user-supplied keys with the biclique construction.

Even for ciphers with a slow local diffusion in the data transform, constructing bicliques of significant length can be made complex by complex and/or non-invertible key schedules. This is due to the complication with the enumeration of the keys with respect to the base key in a biclique. The complication can be both due to the shortage in master key coverage that would destroy any computational advantage one can get though constructing a biclique and due to the complex dependencies of the block data to be operated with on the forward/backward modification of the subkey. The structure of the SC2000 key schedule is such that it does not allow one to efficiently construct independent bicliques.

However, even if one deals with a complex and non-invertible key schedule that would prohibit the advantage through the biclique construction, it is still possible to gain computational advantage from the partial matching in an exhaustive meet-in-the-middle key recovery. Actually, this is the case for SC2000. The results reported below are due to this type of analysis.

Our cryptanalytic results as regards SC2000 are as follows. For the full SC2000-128, we report a MITM key recovery with computational complexity equivalent to $2^{126.5}$ SC2000-128 operations including key schedule as well as negligible data and memory complexities. For the full SC2000-192, we have found a MITM key recovery with computational complexity equivalent to $2^{190.6}$ SC2000-192 operations incl. key schedule as well as negligible data and memory complexities. For the full SC2000-256, we provide a MITM key recovery with computational complexity equivalent to $2^{254.5}$ SC2000-256 operations incl. key schedule as well as negligible data and memory complexities. Note that the factors of advantage are relatively high even given the fact that we were unable to construct bicliques efficiently so far. The relatively high gain is not least due to the heavy key schedule whose cost can be leveraged in a MITM key recovery.

Table 10. Summary of our biclique key recoveries for full SC2000

cipher	rounds	attack type	data	time	memory
SC2000-128	6.5 (of 6.5)	MITM	1 or 2KP	$2^{126.5}$	small
SC2000-192	7.5 (of 7.5)	MITM	2KP	$2^{190.6}$	small
SC2000-256	7.5 (of 7.5)	MITM	2 or 3KP	$2^{254.5}$	small

9.2 Complexity of brute force key recovery

To quantify the complexity of the brute force key recovery with success probability 1, we again look at the non-linear operation of the cipher. For SC2000, there are 3 different types of S-boxes: 4-bit, 5-bit and 6-bit. For the sake of simplicity, we will consider the complexity of these S-boxes equal in our analysis. Strictly speaking, this is not true though and the 6-bit S-box is about 4 times more costly to implement than the 4-bit S-box. However, since there are a lot of S-boxes we will be counting in the complexity estimations of each type, making this distinction would not have much effect on the complexity estimations here.

Thus, according to the computational model introduced in the biclique attacks for AES, the unit will be the application of one S-box. The other operations of the data transform are linear and comparatively less costly. Also for the operation in the key schedule, the S-boxes account for a major part of computations.

Table 11. Computational base line for SC2000 (brute force)

cipher	one B (S-boxes)	$\#B$	one R (S-boxes)	$\#R$	key schedule (S-boxes)	total
SC2000-128	32	7	12	12	576	368+576=944
SC2000-192	32	8	12	14	576	424+576=1000
SC2000-256	32	8	12	14	576	424+576=1000

As indicated in Table 11, one SC2000-128 computation requires in total 944 S-box applications, since each round transform includes 7 B functions and 12 R functions and the key schedule requires another 576 iterative applications of the S-boxes. Similar applies to SC2000-192 and SC2000-256 yielding a total of 1000 S-box applications.

9.3 Key enumeration

The key schedule of SC2000 is non-invertible which complicates the construction of bicliques. The non-invertibility of the key schedule follows from the fact that the values a, b, c and d in the key schedule are obtained by xoring values derived from two adjacent master key words $uk[i]$ and $uk[i+1]$. Furthermore, the actual 32-bit subkey ek can depend on much more than 2 master key words uk due to the extended-key generation function that is also not invertible in ek.

We do not construct a biclique but still have to enumerate the keys efficiently. We divide the key space into groups and modify at most 5 bits at a time. The

modification is placed in $uk[0]$ which is the first 32 bits of the user-supplied key in all SC2000 versions. Modifying in $uk[0]$ means that we need to compute the key schedule only forwards so that its non-invertibility does not matter. In the key recovery, we start with a base key, modify it and compute forward and backward looking for a match.

Table 12. Key modification for SC2000

cipher	key modification	subkey
SC2000-128	5 bits	$uk[0]$
SC2000-192	5 bits	$uk[0]$
SC2000-256	5 bits	$uk[0]$

9.4 Partial matching over 3 rounds

SC2000 does allow for an efficient partial matching over many rounds due to the low-diffusion structure of its data transform though. If the sequence of consecutive functions consisting of I, B, I and double R with appropriate masks and connections is one round, then it is possible to perform a partial matching over 3 rounds of SC2000 (out of 6.5 rounds for SC2000-128 and 7.5 rounds for SC2000-192/256). This allows for savings which is illustrated in Figure 23.

We match in the B function of round 1 in the 3-round block on 8 bits - two rightmost 4-bit S-boxes in B. While a regular recomputation would require 56 S-box computations, the partial matching requires a recomputation of only 30 S-boxes in round 1, 14 S-boxes in round 2, and 42 5-boxes in round 3 of the 3-round block. Thus, the total saving through the matching is 82 S-box computations.

9.5 Recomputation in the key schedule

While S-boxes in the remaining rounds of the data transform need to be recomputed for every key, there is still some room for savings in the key schedule. In fact, the brute force computation of the key schedule is far from efficient.

To compute the values of $a[i]$, $b[i]$, $c[i]$ and $d[i]$, we need 4 iterations of a loop each of which straightforwardly involves 16 layers (S) of 6 S-boxes each (2 6-bit S-boxes and 4 5-bit S-boxes).

For SC2000-128, out of all those S-boxes, one needs to recompute one S-box in $S(uk[0])$ and one S-box in $S(uk[4])$ as well as the agregations (full layers of 6 S-boxes each) before the final linear diffusion to deliver $a[i]$ and $c[i]$. This amounts to 14 S-box recomputations in the first iteration and 12 S-boxes in each of the following 3 iterations. The saving is 576-50=526 S-boxes.

For SC2000-192, it is one S-box in $S(uk[0])$ and one in $S(uk[6])$ that require recomputation along with the corresponding two full layers for aggregation, giving

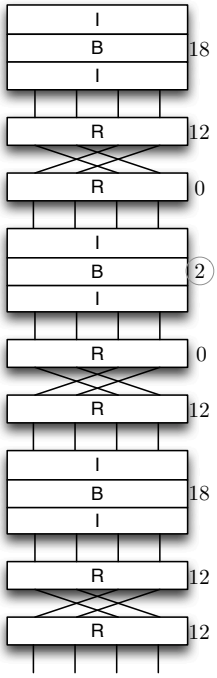


Fig. 23. Partial matching over 3 rounds for SC2000: Numbers indicate the number of S-boxes to be recomputed in the corresponding functions (B or R). The number in grey circle indicates the point of matching

again 14 S-box recomputations in the first iteration and 12 S-boxes in teach of the subsequent ones, saving again 576-50=526 S-boxes

For SC2000-256, it is only one S-box in $S(uk[0])$ and the corresponding 6 S-boxes of aggregation, yielding just 7 S-boxes, that need recomputation in the first iteration. Each subsequent iteration involves 6 S-boxes. The saving amounts to 576-25=551 S-boxes.

We notice that the significant savings in the key schedule are mostly due to the biclique-type computational model assumed and the inefficiency of the reference point. However, we are not aware of a better model here and stick to the standard one.

9.6 Complexity

The savings are as follows. We have a saving of 82 S-box computations due to the partial matching and over 500 S-box computations due to the recomputation

with precomputations in the key schedule. Due to heavy and non-invertible key schedule, it is difficult to efficiently construct non-trivial biclique to attain further savings.

The computational complexities are:

$$\frac{944-82-526}{944}2^{128} \approx 2^{126.5} \text{ for SC2000-128}$$

$$\frac{1000-82-526}{1000}2^{192} \approx 2^{190.6} \text{ for SC2000-192, and}$$
$$\frac{1000-82-551}{1000}2^{256} \approx 2^{254.5} \text{ for SC2000-256.}$$

We omit the false positive testing complexity and the precomputation complexities as we have enough matching bits and the key group is large enough and those would not have a major impact.

Memory complexity is negligible and is mainly due to the precomputations while recomputing. The data complexity of this key recovery is just the unicity distance since we did not employ any initial structure. The data complexity for SC2000-128 is, thus, 1 or 2 known plaintext-ciphertext pairs, for SC2000-192 it is 2 known plaintexts and for SC2000-256 one requires 2 or 3 known plaintexts.

10 Executive Conclusions

With respect to the novel *biclique cryptanalysis* and MITM attacks, all ciphers investigated (AES, Camellia, CLEFIA, SC2000) exhibit some properties that allow one to recover the key slightly faster than brute force. Based on our results so far, one can say that the effect of biclique cryptanalysis on Camellia, CLEFIA and SC2000 is lower than that on AES. Moreover, for AES-128 we have found a property akin to that of the complementation property for DES in terms of its effect on the key recovery: Using just the minimum number of plaintexts (1 or 2 KP), the key recovery complexity drops from 2^{128} to $2^{126.7}$ AES-128 computations. More specifically, we exhaustively analyze the most promising class of biclique cryptanalysis as applied to AES-128: Namely, under some reasonable restrictions, we enumerate all its independent bicliques and stars. In this class of bicliques for AES-128, in a computer-assisted search, we find optimal attacks towards lowest computational and data complexities:

– Among attacks with the minimal data complexity of the unicity distance (i.e. 1 known plaintext is enough with probability $1/e$ and 2 known plaintexts are sufficient for success probability 1), the one of computational complexity $2^{126.7}$ is fastest. Note that its time complexity is only marginally higher than that of the original attack, while it requires only 1 or 2 known plaintexts instead of 2^{88} chosen ciphertexts. This attack has stars instead of balanced bicliques at its core.

– Among attacks with data complexity less than the full codebook, the ones of computational complexity $2^{126.16}$ are fastest. Among those, the one with data complexity 2^{64} requires the smallest amount of data. Thus, the original attack did not have the optimal data complexity.
– Among all attacks covered, the one of computational complexity $2^{125.6}$ is fastest, though requiring the full codebook. This can be considered as an indication of the limitations of the independent-biclique attack approach as applied to AES-128.

Thus, this report explores the limits of the most promising class of bicliques so far in terms of minimizing time and data complexities.

With respect to *zero-correlation linear cryptanalysis*, we discovered some properties over a reduced number of rounds for Camellia and CLEFIA exploitable for key recovery. Based on our results, one can say that Camellia and CLEFIA look robust enough to zero correlation cryptanalysis. Whereas we could attack only less than the half of rounds for Camellia, we were successful for a higher portion of rounds in CLEFIA. Thus, Camellia looks somewhat more secure against zero-correlation cryptanalysis than CLEFIA does.

References

1. Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, Toshio Tokita: Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. Selected Areas in Cryptography 2000:39-56
2. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *Selected Areas in Cryptography'08*, volume 5381 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2008.
3. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *CRYPTO'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2009.
4. Dongxia Bai, Leibo Li: New Impossible Differential Attacks on Camellia. ISPEC 2012:80-96
5. Behran Bahrak and Mohammad Reza Aref. A novel impossible differential cryptanalysis of AES. In *Proceedings of the Western European Workshop on Research in Cryptology 2007 (WEWoRC'07)*, pages 152–156, 2007.
6. Behran Bahrak and Mohammad Reza Aref. Impossible differential attack on seven-round AES-128. *IET Inf. Secur.*, 2(2):28–32, June 2008.
7. Eli Biham, Alex Biryukov, Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. Advances in Cryptology: EUROCRYPT'99, LNCS, vol. 1592, pp. 12–23, Springer-Verlag, 1999.
8. Alex Biryukov, Ivica Nikolic: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. EUROCRYPT 2010:322-344
9. Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 299–319. Springer, 2010.

10. Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In *ASIACRYPT'09*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
11. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In *CRYPTO'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.
12. Andrey Bogdanov, Elif Bilge Kavun, Christof Paar, Christian Rechberger, and Tolga Yalcin. Better than brute-forceoptimized hardware architecture for efficient biclique attacks on AES-128, SHARCS'12 – Special-Purpose Hardware for Attacking Cryptographic Systems, March 2012, Washington D.C., USA, 2012.
13. Andrey Bogdanov, Dmitry Khovratovich and Christian Rechberger. Biclique Cryptanalysis of the Full AES. ASIACRYPT 2011, LNCS 7073, pp. 344–371, Springer-Verlag, 2011.
14. Andrey Bogdanov, Gregor Leander, Kaisa Nyberg, Meiqin Wang. Integral and Multidimensional Linear Distinguishers with Correlation Zero. Asiacrypt'12, Lecture Notes in Computer Science, Springer-Verlag, 2012.
15. Andrey Bogdanov and Christian Rechberger. A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In *SAC'10*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2010.
16. Andrey Bogdanov and Vincent Rijmen. Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. Accepted to Designs, Codes and Cryptography, in press, Springer-Verlag, 2012.
17. Andrey Bogdanov, Vincent Rijmen. Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. Preprint available as Cryptology ePrint Archive: Report 2011/123, http://eprint.iacr.org/2011/123.
18. Andrey Bogdanov, Meiqin Wang. Zero Correlation Linear Cryptanalysis with Reduced Data Complexity. FSE'12, LNCS, Anne Canteaut (ed.), Springer-Verlag, 2012.
19. Andrey Bogdanov and Meiqin Wang. Zero Correlation Linear Cryptanalysis of Some Block Ciphers, unpublished manuscript, 2012.
20. J. Borst, L. R. Knudsen, and V. Rijmen. Two Attacks on Reduced IDEA. In EUROCRYPT'97, LNCS, vol. 1233, pp. 1–13, Springer-Verlag, 1997.
21. Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Pierre-Alain Fouque, Nathan Keller, Vincent Rijmen: Low-Data Complexity Attacks on AES. IEEE Transactions on Information Theory (TIT) 58(11):7002-7017 (2012)
22. Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic Search of Attacks on Round-Reduced AES and Applications. In *CRYPTO'11*, volume 2442 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2011.
23. David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In *CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 192–211, 1986.
24. Jiazhe Chen, Leibo Li: Low Data Complexity Attack on Reduced Camellia-256. ACISP 2012:101-114
25. Jiazhe Chen, Keting Jia, Hongbo Yu, Xiaoyun Wang: New Impossible Differential Attacks of Reduced-Round Camellia-192 and Camellia-256. ACISP 2011:16-33
26. Shao-Zhen Chen, TIan-Min Xu. Biclique Attack of the Full ARIA-256. IACR Eprint Archive Report 2012/11, 2012.
27. B. Collard and F.-X. Standaert: Experimenting Linear Cryptanalysis. In P. Junod, A. Canteaut (eds.) *Advanced Linear Cryptanalysis of Block and Stream Ciphers*, vol, 7 of *Cryptology and Information Security Series*. IOS Press, 2011.

28. B. Collard, F. Standaert, J. Quisquater. Improving the time complexity of matsuis linear cryptanalysis. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 7788. Springer, Heidelberg (2007).
29. Joan Daemen and Vincent Rijmen. The Design of Rijndael. Springer-Verlag, 2002.
30. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In *FSE'08*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.
31. Hüseyin Demirci, Ihsan Taskin, Mustafa Çoban, and Adnan Baysal. Improved Meet-in-the-Middle Attacks on AES. In *INDOCRYPT'09*, volume 5922 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2009.
32. Orr Dunkelman and Nathan Keller. The effects of the omission of last round's MixColumns on AES. *Inf. Process. Lett.*, 110(8-9):304–308, 2010.
33. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In *ASIACRYPT'10*, volume 6477 of *Lecture Notes in Computer Science*, pages 158–176. Springer, 2010.
34. Orr Dunkelman and Nathan Keller. The effects of the omission of last round's MixColumns on AES. *Inf. Process. Lett.*, 110(8-9):304–308, 2010.
35. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In *FSE'00*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2000.
36. Henri Gilbert and Marine Minier. A Collision Attack on 7 Rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.
37. Henri Gilbert and Thomas Peyrin. Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. In *FSE'10*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.
38. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In *ASIACRYPT'10*, volume 6477 of *Lecture Notes in Computer Science*, pages 56–75. Springer, 2010.
39. Takanori Isobe. A single-key attack on the full GOST block cipher. In *FSE'11*, volume 6733 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2011.
40. Takanori Isobe and Kyoji Shibutani. Security Analysis of the Lightweight Block Ciphers XTEA, LED and Piccolo. ACISP 2012, LNCS, Springer-Verlag, 2012.
41. Guan Jie, Zhang Zhongya: Improved Collision Attack on Reduced Round Camellia. CANS 2006:182-190
42. Deukjo Hong, Bonwook Koo, Dong-Chan Kim: Preimage and Second-Preimage Attacks on PGV Hashing Modes of Round-Reduced ARIA, Camellia, and Serpent. IEICE Transactions (IEICET) 95-A(1):372-380 (2012)
43. Liam Keliher: Toward Provable Security Against Differential and Linear Cryptanalysis for Camellia and Related Ciphers. I. J. Network Security (IJNSEC) 5(2):167-175 (2007)
44. Dmitry Khovratovich, Gaetan Leurent, and Christian Rechberger. Narrow Bicliques: Cryptanalysis of Full IDEA. EUROCRYPT'12, LNCS, Springer-Verlag, 2012.
45. Dmitry Khovratovich, Christian Rechberger, and Alexandra Savelieva. Bicliques for preimages: attacks on Skein-512 and the SHA-2 family. Available at http://eprint.iacr.org/2011/286.pdf, 2011.
46. Duo Lei, Chao Li, Keqin Feng: Square Like Attack on Camellia. ICICS 2007:269-283
47. Duo Lei, Chao Li, Keqin Feng: New Observation on Camellia. Selected Areas in Cryptography 2005:51-64
48. Leibo Li, Jiazhe Chen, Keting Jia: New Impossible Differential Cryptanalysis of Reduced-Round Camellia. CANS 2011:26-39

49. Yanjun Li, Wenling Wu, Lei Zhang: Improved Integral Attacks on Reduced-Round CLEFIA Block Cipher. WISA 2011:28-39
50. Ya Liu, Leibo Li, Dawu Gu, Xiaoyun Wang, Zhiqiang Liu, Jiazhe Chen, Wei Li: New Observations on Impossible Differential Cryptanalysis of Reduced-Round Camellia. FSE 2012:90-109
51. Ya Liu, Dawu Gu, Zhiqiang Liu, Wei Li: Improved results on impossible differential cryptanalysis of reduced-round Camellia-192/256. Journal of Systems and Software (JSS) 85(11):2451-2458 (2012)
52. Ji-Qiang Lu: Differential Attack on Five Rounds of the SC2000 Block Cipher. J. Comput. Sci. Technol. (JCST) 26(4):722-731 (2011)
53. Jiqiang Lu: Differential Attack on Five Rounds of the SC2000 Block Cipher. Inscrypt 2009:50-59
54. Jiqiang Lu: The (related-key) impossible boomerang attack and its application to the AES block cipher. Des. Codes Cryptography (DCC) 60(2):123-143 (2011)
55. Jiqiang Lu, Orr Dunkelman, Nathan Keller, and Jongsung Kim. New impossible differential attacks on AES. In *INDOCRYPT'08*, volume 5365 of *Lecture Notes in Computer Science*, pages 279–293. Springer, 2008.
56. Jiqiang Lu, Jongsung Kim, Nathan Keller, Orr Dunkelman: Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1. CT-RSA 2008:370-386
57. Jiqiang Lu, Yongzhuang Wei, Enes Pasalic, Pierre-Alain Fouque: Meet-in-the-Middle Attack on Reduced Versions of the Camellia Block Cipher. IWSEC 2012:197-215
58. Stefan Lucks. Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In *AES Candidate Conference*, pages 215–229, 2000.
59. Hamid Mala. Bilcique Cryptanalysis of the Block Cipher Square. IACR Eprint Archive Report 2011/500, 2011.
60. Hamid Mala, Mohammad Dakhilalian, Mohsen Shakiba: Impossible Differential Attacks on 13-Round CLEFIA-128. J. Comput. Sci. Technol. (JCST) 26(4):744-750 (2011)
61. Hamid Mala, Mohammad Dakhilalian, Vincent Rijmen, and Mahmoud Modarres-Hashemi. Improved Impossible Differential Cryptanalysis of 7-Round AES-128. In *INDOCRYPT'10*, volume 6498 of *Lecture Notes in Computer Science*, pages 282–291. Springer, 2010.
62. Hamid Mala, Mohsen Shakiba, Mohammad Dakhilalian, Ghadamali Bagherikaram: New Results on Impossible Differential Cryptanalysis of Reduced-Round Camellia-128. Selected Areas in Cryptography 2009:281-294
63. Raphael Chung-Wei Phan. Impossible differential cryptanalysis of 7-round advanced encryption standard (AES). *Inf. Process. Lett.*, 91(1):33–38, 2004.
64. Havard Raddum, Lars R. Knudsen: A Differential Attack on Reduced-Round SC2000. Selected Areas in Cryptography 2001:190-198
65. Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In *EUROCRYPT'09*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2009.
66. Yu Sasaki. Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool. In *FSE'11 Preproceedings*, 2011.
67. Yu Sasaki, Sareh Emami, Deukjo Hong, Ashish Kumar: Improved Known-Key Distinguishers on Feistel-SP Ciphers and Application to Camellia. ACISP 2012:87-100
68. Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, Hidema Tanaka: The Block Cipher SC2000. FSE 2001:312-327
69. Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, Tetsu Iwata: The 128-Bit Blockcipher CLEFIA (Extended Abstract). FSE 2007:181-195

70. Xuehai Tang, Bing Sun, Ruilin Li, Chao Li: Impossible differential cryptanalysis of 13-round CLEFIA-128. Journal of Systems and Software (JSS) 84(7):1191-1196 (2011)
71. Michael Tunstall: Improved "Partial Sums"-based Square Attack on AES. SECRYPT 2012:25-34
72. Cihangir Tezcan: The Improbable Differential Attack: Cryptanalysis of Reduced Round CLEFIA. INDOCRYPT 2010:197-209
73. Yukiyasu Tsunoo, Etsuko Tsujihara, Maki Shigeri, Teruo Saito, Tomoyasu Suzaki, Hiroyasu Kubo: Impossible Differential Cryptanalysis of CLEFIA. FSE 2008:398-411
74. Qingju Wang, Andrey Bogdanov: The provable constructive effect of diffusion switching mechanism in CLEFIA-type block ciphers. Inf. Process. Lett. (IPL) 112(11):427-432 (2012)
75. Yanfeng Wang, Wenling Wo, Xiaoli Yu. Biclique Cryptanalysis of Reduced-Round Piccolo Block Cipher. IPSEC 2012, LNCS, vol. 7232, pp.337-352, Springer-Verlag, 2012.
76. Yongzhuang Wei, Jiqiang Lu, Yupu Hu: Meet-in-the-Middle Attack on 8 Rounds of the AES Block Cipher under 192 Key Bits. ISPEC 2011:222-232
77. Wenling Wu, Lei Zhang, Wentao Zhang: Improved Impossible Differential Cryptanalysis of Reduced-Round Camellia. Selected Areas in Cryptography 2008:442-456
78. Wenling Wu, Wentao Zhang, Dengguo Feng: Impossible Differential Cryptanalysis of Reduced-Round ARIA and Camellia. J. Comput. Sci. Technol. (JCST) 22(3):449-456 (2007)
79. Hitoshi Yanami, Takeshi Shimoyama, Orr Dunkelman: Differential and Linear Cryptanalysis of a Reduced-Round SC2000. FSE 2002:34-48
80. Wenying Zhang, Jing Han: Impossible Differential Analysis of Reduced Round CLEFIA. Inscrypt 2008:181-191
81. Wentao Zhang, Wenling Wu, and Dengguo Feng. New results on impossible differential cryptanalysis of reduced AES. In *ICISC'07*, volume 4817 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2007.