Green Cryptanalysis: Meet-in-the-Middle Key-Recovery for the Full KASUMI Cipher [⋆]

Keting Jia[1], Christian Rechberger[2], and Xiaoyun Wang[1,3⋆⋆]

[1] Institute for Advanced Study, Tsinghua University, China
{ktjia,xiaoyunwang}@mail.tsinghua.edu.cn
[2] Department of Mathematics, Technical University of Denmark, Denmark
c.rechberger@mat.dtu.dk
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, China

Abstract. KASUMI is a block cipher with eight Feistel rounds and a key of up to 128 bits. Proposed more than 10 years ago, the confidentiality and integrity of 3G mobile communications systems depend on the security of KASUMI. In the practically interesting single key setting that we are aiming for in this work, no attack is known.

For the full 8-round KASUMI we show for the first time a wide variety of results with data complexities between 2^{32} chosen plaintexts and as few as 2 texts, while the speed-ups over brute force are between a factor 4 and 6. For use-cases of KASUMI in 2G networks, relying on a 64-bit master key, we describe key recovery methods with extremely low data complexity and speed-ups between a factor 2 and 3 for essentially any desired success probability. The latter results are the first of this type of cryptanalysis that could result in practically realizable cost and energy savings for key recovery efforts.

By also analyzing an earlier version of the KASUMI-64 design that had a different mapping from the 64-bit master key to the 128-bit cipher key, we shed some light on a high-level key schedule design issue that may be of independent interest.

Keywords: KASUMI, KASUMI-64, Meet-in-the-Middle Attack, Cryptanalysis

1 Introduction

The block cipher KASUMI, designed by ETSI SAGE [29], is the foundation of the 3GPP encryption and integrity algorithms. Nowadays, it is also recommended as the standard algorithms A5/3 and GEA3 in GSM and GPRS mobile communications systems, respectively [31]. By now there are over 1-billion subscribers for 3GPP and more than 4-billion GSM devices. Although A5/3 is not yet widely deployed in GSM networks, it is widely considered to be imperative to switch to A5/3 from A5/1, because it is trivial to eavesdrop GSM conversations [27,30], even to emulate a mobile phone to make phone calls and send text messages [28] using a time-memory trade-off attack on A5/1 [4]. In response to these attacks, the GSM Association has vowed to switch to the much more modern A5/3 cipher since 2010. Once adopted, KASUMI will become one of the most widely used cryptographic algorithms in the world. Hence, it is very important to understand the security offered by KASUMI, especially with the particular applications of KASUMI in mind.

Earlier cryptanalysis of KASUMI. KASUMI is a slightly modified version of the block cipher MISTY1 [26], which is optimized for hardware performance and has a 128-bit key and 8 Feistel rounds. Owing to the importance of KASUMI, it has received a lot of interest from the cryptographic research community. In the single-key setting, Kühn introduced the impossible differential attack on 6-round KASUMI [24], which was very recently extended to 7-round KASUMI by Jia et al. [21]. Sugio et al. gave a higher order differential attack on 5-round KASUMI with practical complexity [33]. Since the key schedule of KASUMI is linear and simple, many attacks

[⋆] Supported by 973 Program(2013CB834205) the National Natural Science Foundation of China (61133013 and 60931160442)

[⋆⋆] Corresponding author

were presented in the related-key setting. Blunden et al. gave a related-key differential attack on 6-round KASUMI [8]. The first related-key attack on the full 8-round KASUMI was proposed by Biham et al. with $2^{76.1}$ encryptions [6], which was improved to a practical related-key attack on the full KASUMI by Dunkelman et al.[12]. However, these attacks assume control over the differences of two or more related keys in all the 128 key bits. This renders the resulting attack inapplicable in most real-world usage scenarios. To our knowledge there is no attack on the full KASUMI without related keys.

MITM attacks. In this paper, we explore the meet-in-the-middle(MITM) attack vector and apply it to the full 8-round KASUMI. The meet-in-the-middle attack was first introduced by Diffie and Hellman to analyze the block cipher DES [14]. The method was used to analyze KTANTAN, GOST and IDEA, reduced-round DES and reduced-round AES [7,10,16,19,22]. Furthermore, the meet-in-the-middle framework found applications in integral attacks [13,15,17]. Recently, started by pioneering work of Aoki and Sasaki, this type of attack was extended to hash functions, and has been improved with many techniques to carry out preimage attacks on the hash functions MD5, Tiger, and reduced versions of SHA-1 and SHA-2 etc.[2,3,18,32]. These techniques include the splice-and-cut framework, initial structure and partial matching etc.

Bruteforce-like cryptanalysis. Khovratovich et al. introduced the biclique cryptanalysis originating from the initial structure technique in hash function cryptanalysis [23]. Later, the biclique technique was utilized to analyze block ciphers, and combined with a *partial brute-force* resulting in a meet-in-the-middle framework to give the first attacks on the full AES-128/-192/-256 with 3-5 times improved speed over a brute force attack [9]. Bruteforce-like cryptanalysis is not able to conclude that a particular target has a cryptanalytic weakness, as in principle any number of rounds can be “attacked”. However it can help better understand the real security offered against attacks in the absence of other shortcuts. Most recently reported applications of bruteforce-like biclique cryptanalysis have an advantage that is much smaller than a factor of 2 (e.g. [1,11,20,23]). For ciphers like AES with key sizes of 128 bits or more this is merely of academic interest, we argue however that for ciphers with key sizes of 80 bits or less, this is very useful to know, especially when cost saving is a factor 2 or more.

New results. In this paper, we capture some observations on KASUMI, which are used to separate some key words to speed up the key search. We firstly develop a MITM attack on full KASUMI with only a single chosen plaintext and a known plaintext that is about 4 times faster than brute-force search. Then we give an attack with lower time complexity and a higher data complexity. KASUMI has a 128-bit key in the specification, but the key length needs to be reduced in some cases, e.g., for downwards compatibility to 2G networks, which only allow the key length to be 64 [31] bits. Subsequently we call the block cipher KASUMI used in GSM and GPRS “KASUMI-64”. As a result of the widespread use of GSM mobiles, we also focus on KASUMI-64. We obtain a noticeable improvement over the brute-force attack for the full 8 rounds by the using the MITM attacks. Firstly, we start by giving a MITM attack on KASUMI-64, which only needs a single known plaintext/ciphertext pairs and still results in a factor 2 speed-up over brute-force. Secondly, this is improved to a computational complexity of about $2^{62.63}$ encryptions and a data complexity of 2^{20} chosen plaintexts. Furthermore, a trade-off between the data complexity and time complexity is given, which needs 1152 chosen plaintexts, and a time complexity of $2^{62.75}$ encryptions. All time complexities are worst-case complexities, and the amount of known or chosen plaintext is small enough to be relevant for the actual use-case of KASUMI, i.e. in specific communication protocols. As exhaustive search of a key space of size 2^{64} is feasible for well-funded adversaries, even such relatively small improvements using new cryptanalytic results still have the *potential for a practical impact* in terms of computational complexity and also energy consumption that is very rare for modern designs that withstood cryptanalytic scrutiny for a long time. Hence the tongue-in-cheek “Green Cryptanalysis” in the title.

Paper organization. We give a brief description of the block cipher KASUMI in Sect. 2. Section 3 presents meet-in-the-middle attacks on the full KASUMI. We introduce several MITM attacks on the full KASUMI-64 and draw a lesson on key-schedule design in Sect. 4, before we summarize the findings and conclude in Sect. 5.

2 Description of KASUMI

KASUMI uses a key of up to 128-bit to encrypt a 64-bit block. A brief description of KASUMI is given in this section.

Key schedule. The key schedule of KASUMI is much simpler than the original key schedule of MISTY1, which makes the hardware significantly smaller and reduces key set-up time. The 128-bit key K is divided into eight 16-bit words: k_1,k_2, ..., k_8, i.e., $K = (k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8)$, which are used to compute the round subkeys. The round subkeys consist of three parts KL_i, KO_i and KI_i, where $KL_i = (KL_{i,1}, KL_{i,2})$, $KO_i = (KO_{i,1}, KO_{i,2}, KO_{i,3})$ and $KI_i = (KI_{i,1}, KI_{i,2}, KI_{i,3})$. We summarize the details of the key schedule of KASUMI in Tab. 3 in App. B.

KASUMI-64 and variants. The key length of KASUMI is allowed to 128 bits at most. For KASUMI used in GSM and GPRS, the key length is set to 64, and the most significant bits of key carry the effective 64-bit key information, whereas the remaining, least significant bits duplicate the key, i.e., $k_5 = k_1$, $k_6 = k_2$, $k_7 = k_3$, $k_8 = k_4$. We call it KASUMI-64. The variant of KASUMI-64 is to simply set $k_5, \ldots, k_8$ to a constant zero. In fact this option was chosen in an earlier version of the KASUMI-64 design [34], and we refer to as KASUMI-64-0.

Encryption. KASUMI has a 8-round Feistel structure, and each round consists of an FL function and an FO function. The FL function precedes the FO function in odd numbered rounds, whereas the FO function precedes the FL function in even numbered rounds (see Fig. 7 (a) in App. B).

Let $L_{i-1}||R_{i-1}$ be the input of the i-th round, and then the round function is defined as

$$\begin{aligned} L_i &= FO(FL(L_{i-1}, KL_i), KO_i, KI_i) \oplus R_{i-1}, \\ R_i &= L_{i-1}, \end{aligned}$$

where $i = 1, 3, 5, 7$.‘ $\oplus$ ‘ denotes the bitwise exclusive-or (XOR), and ‘$\|$‘ represents the concatenation. When $i = 2, 4, 6, 8$,

$$\begin{aligned} L_i &= FL(FO(L_{i-1}, KO_i, KI_i), KL_i) \oplus R_{i-1}, \\ R_i &= L_{i-1}. \end{aligned}$$

Here, L_{i-1}, R_{i-1} denote the left and right 32-bit halves of the i-th round input, and $L_0\|R_0$, $L_8\|R_8$ are the plaintext and ciphertext respectively.

The FL function is a simple key-dependent boolean function. Let $XL_i = XL_{i,l}\|XL_{i,r}$, $KL_i = (KL_{i,1}, KL_{i,2})$ be the input of the FL function of the i-th round, and $YL_i = YL_{i,l}\|YL_{i,r}$ be the corresponding output, where $XL_{i,l}, XL_{i,r}, YL_{i,l}$ and $YL_{i,r}$ are 16-bit variables. The FL function has the form:

$$YL_{i,r} = ((XL_{i,l} \wedge KL_{i,1}) \lll 1) \oplus XL_{i,r}, \tag{1}$$
$$YL_{i,l} = ((YL_{i,r} \vee KL_{i,2}) \lll 1) \oplus XL_{i,l}, \tag{2}$$

where ‘ $\wedge$ ‘ and ‘ $\vee$ ‘ denote bitwise AND and OR respectively, and ‘$x \lll i$‘ implies that x rotates left by i bits. We call FL_i the FL function of i-th round with subkey KL_i.

The FO function is another three-round Feistel structure made up of three FI functions and key mixing stages, which provides the non-linear property in each round (see Fig. 7 (b) in App. B). Let $XO_i = XO_{i,l}\|XO_{i,r}$, $KO_i = (KO_{i,1}, KO_{i,2}, KO_{i,3})$, $KI_i = (KI_{i,1}, KI_{i,2}, KI_{i,3})$ be the input of the FO function of i-th round, and $YO_i = YO_{i,l}\|YO_{i,r}$ be the corresponding output, where

$XO_{i,l}, XO_{i,r}, YO_{i,l}, YO_{i,r}$ and $XO_{i,3}$ are 16-bit variables. Then the FO function is defined as follows:

$$\begin{aligned} XO_{i,3} &= FI((XO_{i,l} \oplus KO_{i,1}), KI_{i,1}) \oplus XO_{i,r}, \\ YO_{i,l} &= FI((XO_{i,r} \oplus KO_{i,2}), KI_{i,2}) \oplus XO_{i,3}, \\ YO_{i,r} &= FI((XO_{i,3} \oplus KO_{i,3}), KI_{i,3}) \oplus YO_{i,l}. \end{aligned}$$

For simplicity, we call FO_i the FO function of i-th round with subkeys KO_i and KI_i.

The FI function includes two Sboxes S7 and S9 which are permutations of 7-bit to 7-bit and 9-bit to 9-bit respectively. Suppose the input of the j-th FI function of the i-th round is ($XI_{i,j}$, $KI_{i,j}$) and the output is $YI_{i,j}$, where $XI_{i,j}$ and $YI_{i,j}$ are 16-bit variables. In order to abbreviate the FI function, we define half of FI function as $\overline{FI}$, which is a 16-bit to 16-bit permutation. See Fig. 7 (d) in App. B for the illustrations of FI and $\overline{FI}$. $\overline{YI}_{i,j} = \overline{FI}(XI_{i,j})$ is written as

$$\begin{aligned} \overline{YI}_{i,j}[0-8] &= S9(XI_{i,j}[7-15]) \oplus XI_{i,j}[0-6], \\ \overline{YI}_{i,j}[9-15] &= S7(XI_{i,j}[0-6]) \oplus \overline{YI}_{i,j}[0-6], \end{aligned}$$

where $z[i_1 - i_2]$ denotes the $(i_2 - i_1 + 1)$ bits from the i_1-th bit to i_2-th bit of z, and '0' is the least significant bit. The FI function is simplified as

$$YI_{i,j} = FI(XI_{i,j}, KI_{i,j}) = \overline{FI}((\overline{FI}(XI_{i,j}) \oplus KI_{i,j}) \lll 7).$$

We call $FI_{i,j}$ the j-th FI function of the i-th round with subkey $KI_{i,j}$, and define four variables $XI_{i,j,1}$, $\overline{XI}_{i,j}$, $YI_{i,j,1}$ and $YI_{i,j,2}$ for simplification (see the illustration of FI function in Fig. 7 (d) in App. B).

$$\begin{aligned} XI_{i,j,1} &= S9(XI_{i,j}[7-15]), & \overline{XI}_{i,j} &= \overline{YI}_{i,j} \oplus KI_{i,j}, \\ YI_{i,j,1} &= S9(\overline{XI}_{i,j}[0-8]), & YI_{i,j,2} &= S7(\overline{XI}_{i,j}[9-15]). \end{aligned}$$

3 Meet in the Middle Attacks on KASUMI

In contrast to SPN networks like the AES, where the number of subkey-bits used per round exactly corresponds to the block size of the cipher, KASUMI uses the 1.5 fold amount. This makes, in principle, an application of MITM attacks based on separating key spaces more difficult. Nevertheless we will design a key recovery method with this high level approach in mind.

In particular, we capture some observations on KASUMI, which are used to separate some key words to speed up the key search. Combining the observation of FL function, we explore a meet-in-the-middle attack on KASUMI, which only needs a single chosen plaintext and a known plaintext with a factor 4 speed-up. This attack is converted to a known plaintext attack, which needs about 128 known plaintexts with the same time complexity. Besides, we find that when the left 32-bit inputs of the first and third round are assigned to the value $0x0000ffff$, there exists a 3.3-round subcipher without the 16-bit key word k_3. Based on the subcipher and several optimizations, we give an improved attack, which succeeds with a data complexity of 2^{32} chosen plaintexts and a time complexity of $2^{125.5}$ encryptions.

3.1 Some Observations in KASUMI

Let the plaintext $P = P_l \| P_r$, the ciphertext $C = C_l \| C_r$, and $L_i = L_{i,l} \| L_{i,r}$ be the left 32-bit input of $(i+1)$-th round, $i = 0, 1, ..., 7$ (see Fig. 7 (a) in App. B).

Observation 1 *Let XL_i, YL_i be the input and output of FL_i, then*

$$\begin{aligned} YL_{i,r}[j+1] &= XL_{i,r}[j+1] \oplus (XL_{i,l}[j] \wedge KL_{i,1}[j]) = XL_{i,r}[j+1] \ \textit{when}\ XL_{i,l}[j] = 0, \\ YL_{i,l}[j+1] &= XL_{i,l}[j+1] \oplus (YL_{i,r}[j] \vee KL_{i,2}[j]) = XL_{i,l}[j+1] \oplus 1 \ \textit{when}\ YL_{i,r}[j] = 1, \end{aligned}$$

where $j = 0, 1, \ldots, 15$, and $j+1$ is an integer modulo 16.

This observation is straightforward by the bit equation presentation of the FL function, and it implies that $KL_{i,1}[j]$ does not affect the output $YL_{i,r}$ when $XL_{i,l}[j] = 0$, and $KL_{i,2}[j]$ does not affect the output $YL_{i,l}$ when $YL_{i,r}[j] = 1$.

Observation 2 *The intermediate value* $\overline{XI}_{i,1}[7-8]$ *is independent of* $KO_{i,1}[0-6]$ *in the function* FO_i*, and* $XI_{i,j}[0-6]$ *is independent of* $KI_{i,j}[7-8]$ *in the inversion of the* $FI_{i,j}$ *function.*

Proof. By the definition of FO function, we know

$$\overline{XI}_{i,1}[7-8] = S9(XO_{i,1}[7-15] \oplus KO_{i,1}[7-15])[7-8] \oplus KI_{i,j}[7-8].$$

Hence, $\overline{XI}_{i,1}[7-8]$ is independent of $KO_{i,1}[0-6]$.

We know $KI_{i,j}$ is not involved in the computation of the intermediate value $\overline{XI}_{i,j}$ by inverting $FI_{i,j}$, and $XI_{i,j}[0-6]$ is computed as follows,

$$XI_{i,j}[0-6] = S7^{-1}(KI_{i,j}[0-6] \oplus \overline{XI}_{i,j}[0-6] \oplus KI_{i,j}[9-15] \oplus \overline{XI}_{i,j}[9-15]).$$

It is obvious that $XI_{i,j}[0-6]$ is independent of $KI_{i,j}[7-8]$. □

This observation is applied to reduce the data complexity in our analysis of KASUMI-64.

Observation 3 *Let* $L_0 = L_2 = 0x0000ffff$*. Then there exists a subcipher covering 3.3 rounds of KASUMI, which does not include the key word* k_3*.*

Proof. We compute the plaintext variant $\overline{P}$ and intermediate value S from L_0 and L_2 by partial encryption and decryption, which is depicted in Fig. 1.

From equations (1) and (2), it is easy to know that when the input of FL is $0x0000ffff$, the output of FL is always $0xffffffff$, for all the values of KL_1 and KL_3.

Because the key words in rounds 1 and 3 do not contain k_3 except KL_1 and KL_3. So, the output values YO_1 and YO_3 of the first and third round functions have no relation with the key word k_3, where

$$YO_1 = FO(0xffffffff, KO_1, KI_1), YO_3 = FO(0xffffffff, KO_3, KI_3).$$

According to the details of KASUMI, we deduce the output YL_2 of FL_2 function,

$$YL_2 = L_0 \oplus L_2 = 0x0000ffff \oplus 0x0000ffff = 0.$$

Inverting the second round function, the input $XI_{2,1}$ of $FI_{2,1}$ and the right 16-bit $L_{1,r}$ of L_1 are obtained, and $L_{1,l} = XI_{2,1} \oplus (k_3 \lll 5)$.

By forward encryption from L_0 and L_2, we get the internal state $S = (YI_{4,2}, F)$ satisfying the following equations:

$$\begin{aligned} YI_{4,2} &= FI(YO_{3,r} \oplus L_{1,r} \oplus KO_{4,2}, KI_{4,2}), \\ F &= YO_{3,l} \oplus XI_{2,1} \oplus KO_{4,1}. \end{aligned}$$

We obtain a plaintext variant $\overline{P} = (\overline{P}_l \| \overline{P}_r)$ by backward decryption from L_2.

$$\begin{aligned} \overline{P}_l &= L_0 = 0x0000ffff, \\ \overline{P}_r &= YO_1 \oplus (XI_{2,1} \| L_{1,r}). \end{aligned}$$

It is easy to verify that $P_{r,l} = \overline{P}_{r,l} \oplus (k_3 \lll 5)$ and $P_{r,r} = \overline{P}_{r,r}$.

Because k_3 is not intervened in the computations of YO_1, YO_3, $XI_{2,1}$ and $L_{1,r}$, the pair ($\overline{P}$, S) is independent of the key word k_3. There are about 3.3-round computations from $\overline{P}$ to S in total, so we conclude that there exists a 3.3-round subcipher from $\overline{P}$ to S without k_3 when $L_0 = L_2 = 0x0000ffff$. □

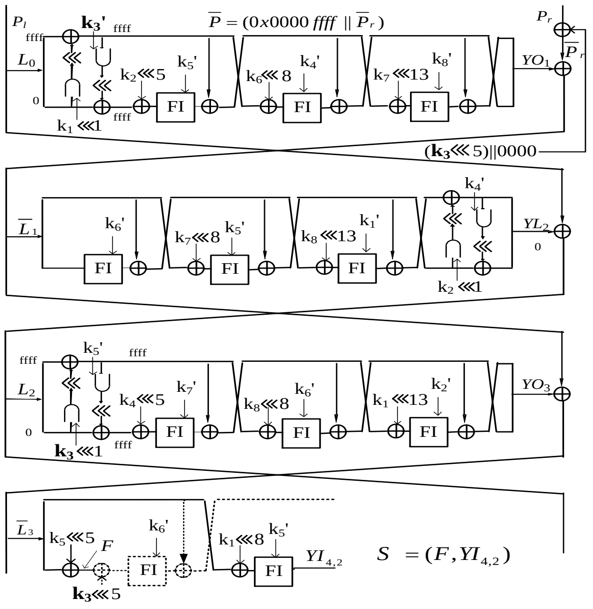


Fig. 1. The subcipher covering 3.3 rounds of KASUMI without key word k_3

This interesting phenomenon motivates us to explore an attack on KASUMI by separating the 16-bit key word k_3 from the other key words to speed-up the search of the total key space.

Observation 4 *For KASUMI, the key words k_2, k_3 and k_8 are not involved in the computations of the functions $FI_{8,1}$, $FI_{8,3}$ and $\overline{YI}_{8,2} = \overline{FI}(KO_{8,2} \oplus C_{r,r})$ in the process of decrypting a ciphertext C, where $\overline{YI}_{8,2}$ is the output of the first $\overline{FI}$ function of $FI_{8,2}$.*

This observation follows straightforwardly from the definition of KASUMI, and is used to separate the key words k_2, k_3 and k_8 from the rest of the key space to skip some computations in the last round.

Distinguisher. For a known pair (P, C) of KASUMI, we get $L_5 = YO_5 \oplus L_3$ by encrypting the plaintext, and $L_5 = YO_7 \oplus L_7$ by decrypting the ciphertext. Then we know the equation

$$YO_5 \oplus L_3 = YO_7 \oplus L_7 \tag{3}$$

holds with probability 1 for the right key K. But for a wrong key, it holds only with probability 2^{-32}. We also get another equation

$$YO_3 \oplus L_1 = YO_5 \oplus L_5. \tag{4}$$

Equation (4) holds with probability 1 for the right key K, and holds with probability 2^{-32} for a wrong key too. The two equations are used as distinguishers in our attacks.

Observation 5 *Given the input value (L_4, L_3) of the 5-th round and the output value (L_7, L_6) of the 7-th round, it takes about 9 Sbox calls to detect the equation $YO_5 \oplus L_3 = YO_7 \oplus L_7$ hold or not for a guessed key, where YO_5 and YO_7 are the outputs of FO_5 and FO_7 respectively.*

The equation $YO_5 \oplus L_3 = YO_7 \oplus L_7$ is able to be expressed as 32 bit-equations in parallel. Hence, we apply the partial matching technique and early abort technique to test it. The complexity proof refers to App. A.

Cost models. In the evaluation of time complexity of our attacks, we take a hardware-centric view: the number of Sbox calls will dominate the cost, as all other operations are comparatively cheap to implement, both in terms of gate-count or energy consumption. Hence, for simplicity, the time complexity only considers the number of Sbox calls.

3.2 The MITM Attack on KASUMI with a Single Chosen Plaintext

In the cryptanalysis of a block cipher, some attacks usually take high data complexities, such that it is very hard to carry out in the practical attack environments owing to prohibitive high data complexities. Here we propose an attack on KASUMI with only a single chosen plaintext and an additional known plaintext, using Observation 1 and the key schedule algorithm.

We consider a special plaintext P for KASUMI. The left 16 bits of the plaintext $P_{l,l} = 0x0000$, which is used to absorb the impact of key word k_1 on the FL_1 function. The remaining 48-bit part of the plaintext is allowed to any 48-bit value. On the basis of the definition of round functions, we know that $k_1[9-15]$ is independent of the computation of the intermediate value $S_0 = (YI_{2,2}, \overline{YI}_{2,3}[9-15], YI_{2,3,1})$ by partially encrypting the plaintext P. Then we decrypt the ciphertext C corresponding to P, and get the intermediate value $YL_{8,r}$. There are about 8 bits of $YL_{8,r}$ which equal to 1. By Observation 1, 8 bits of k_2 have no impact on the computations of $YL_{8,l}$ by partial decrypting C, which hence avoid some computations from YL_8 to the intermediate value $(\overline{YI}_{7,2}, YI_{7,3})$. Furthermore, there are about 4.5 bits [4] of $k_2[0-8]$ independent of the computations of the intermediate values $S_1 = (\overline{YI}_{7,2}[0-8], YI_{7,2,2}, YI_{7,3})$. Hence, we get two trunks: one is partial encryption from P to S_0, in which 7-bit $k_1[9-15]$ is not involved, and the other is partial decryption from C to S_1, in which there are about 4.5 bits of $k_2[0-8]$ is not used (see Fig. 2).

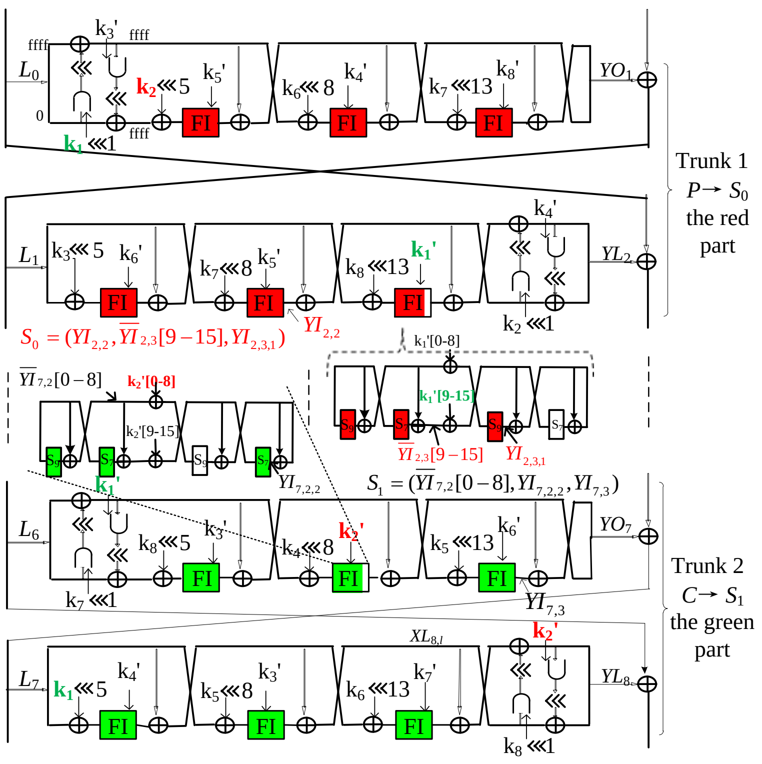


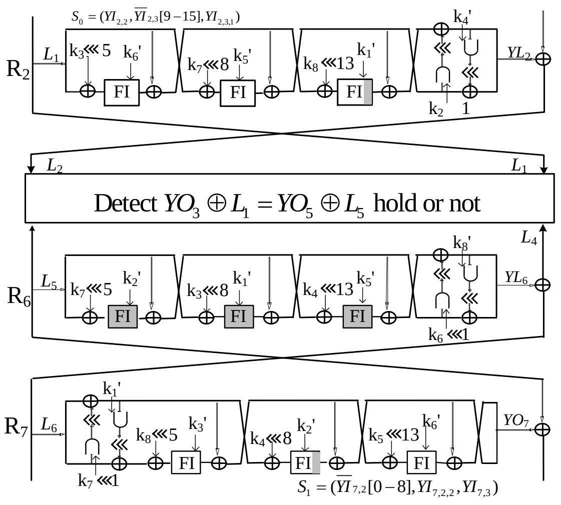
Fig. 2. The two trunks of KASUMI in the MITM attack with a single chosen plaintext

Let $P = 0x0000 \| P_{l,r} \| P_r$, where $P_{l,r} \| P_r$ is a fixed 48-bit random value, and then query the corresponding ciphertext C. We divide the key space into 2^{112} small groups corresponding to $(k_1[0-8], k_2[9-15], k_3, k_4, k_5, k_6, k_7, k_8)$, and each group traverses $(k_1[9-15], k_2[0-8])$. For each group, we execute the following process to distinguish the right key (see Fig. 3).

[4] There are about 4.5 js, such that $YL_{8,r}[j] = 1$, which absorb the impact of $k_2[j]$ on $YL_{8,l}$, where $j = 0, ..., 8$.

1. Traverse $k_2[0-8]$, encrypt the plaintext P to get the intermediate values L_1, $S_0 = (YI_{2,2}, \overline{YI}_{2,3}[9-15], YI_{2,3,1})$, and store them in a hash table indexed with $k_2[0-8]$. It takes $2^9 \times 23$ Sbox calls.
2. For each $k_1[9-15]$, compute the intermediate values $(XL_{8,l}, YL_{8,r})$ by partial decryption. By Observation 1, for j=0,...,8, when $YL_{8,r}[j] = 0$, guess $k_2[j]$ which affects the output $YL_{8,l}$, and compute the intermediate values L_6 and $S_1 = (\overline{YI}_{7,2}[0-8], YI_{7,2,2}, YI_{7,3})$. Then guess the remaining bits of $k_2[0-8]$, and compute the output values of the 5-th round, which needs $2^7 \times 12 + 2^7 \times 2^{4.5} \times 11 + 2^7 \times 2^9 \times 13$ [5]Sbox calls.
3. Partially encrypt S_0 corresponding to $k_2[0-8]$ with the same subkey $k_1[9-15]$ as Step 2 to get the input values of the 3-rd round. We spend 2^{16} Sbox calls in this step.
4. For the input values of the 3-rd round and the output values of the 5-th round corresponding to $(k_1[9-15], k_2[0-8])$, detect the equation $YO_3 \oplus L_1 = YO_5 \oplus L_5$ hold or not by the similar computations as Observation 5, which takes $2^{16} \times 9$ Sbox calls.
5. For the subkeys which make Equation (4) hold, we search the right key by trial encryptions with 2 known plaintext/ciphertext pairs. There are about $2^{-32} \times 2^{128} = 2^{96}$ keys to keep Equation (4) hold, resulting in the time complexity of 2^{96} encryptions in this step.

Therefore, we need $2^{112} \times 2^{16} \times (13.69 + 1 + 9) = 2^{128} \times 23.69$ Sbox calls, which are equivalent to $2^{125.98}$ encryptions. The data complexity is a chosen plaintext and a known plaintext. And the memory complexity is 2^7 blocks to store the hash table in Step 1.



$S_0 = (YI_{2,2}, \overline{YI}_{2,3}[9-15], YI_{2,3,1})$
R2
L1
k3⋘5
k6'
k7⋘8
k5'
k8⋘13
k1'
k4'
FI
k2 1
YL2
L2
L1
Detect $YO_3 \oplus L_1 = YO_5 \oplus L_5$ hold or not
L4
R6
L5
k7⋘5
k2'
k3⋘8
k1'
k4⋘13
k5'
k8'
YL6
k6⋘1
R7
L6
k1'
k8⋘5
k3'
k4⋘8
k2'
k5⋘13
k6'
YO7
k7⋘1
$S_1 = (\overline{YI}_{7,2}[0-8], YI_{7,2,2}, YI_{7,3})$

Fig. 3. Detect whether $YO_3 \oplus L_1 = YO_5 \oplus L_5$ holds

It is remarked that our attack only needs a single chosen plaintext of this special form, and a subsequent known plaintext/ciphertext pair to filter the remaining key candidates. For each key group taking 2^{16} values of $(k_1[9-15], k_2[0-8])$, 7 bit-conditions $P_{l,l}[1, 10-15] = 0$ are necessary to make the computations from P to S_0 independent of $k_1[9-15]$. Since there are about one plaintext such that $P_{l,l}[1, 10-15] = 0$ among 128 random plaintexts, it is feasible to convert this attack to a known-plaintext attack. The known-plaintext attack needs 128 plaintext/ciphertext pairs and the time complexity is also $2^{125.98}$ encryptions. The success rate is dominated by the probability to get a plaintext P with $P_{l,l}[1, 10-15] = 0$ from 128 random plaintexts, i.e., $1 - (1 - 1/128)^{128} = 0.63$.

[5] There are 4.5 bits of the 9-bit $k_2[0-8]$ which are absorbed by $YL_{8,r}[0-8]$ on average, because there are about 4.5 js, such that $YL_{8,r}[j] = 1$, where $j = 0, ..., 8$.

3.3 The MITM Attack on KASUMI with Improved Time Complexity

In this section, combining Observation 3 and Observation 4, we explore a meet-in-the-middle attack on KASUMI with a little improvement of the time complexity.

Data collection. We know that the left 32-bit parts of plaintexts are assigned to a constant, i.e., $P_l = 0x0000ffff$ according to Observation 3. For the 2^{32} plaintexts $P = 0x0000ffff\|P_r$, collect their corresponding ciphertexts C, and store the pairs (P_r, C) in a hash table T_1 with the 16-bit index $P_{r,r}$. The total data complexity is hence 2^{32} chosen plaintexts.

Key recovery. Observation 3 shows that k_3 has no relation with the subcipher $\overline{P} \xrightarrow{3.3\ rounds} S$ under the given condition $L_0 = L_2 = 0x0000ffff$, so it is feasible to separate k_3 from other key words to speed-up exhaustive search. We also notice that the key words (k_2, k_3, k_8) have no effect on the partial decryption from C to S_1, where $S_1 = (\overline{YI}_{8,2}, YI_{8,3})$ by Observation 4. Consequently, we start our attack by partitioning the whole key space into small key groups. Each group corresponds to a value of key words $\overline{K} = \{k_1, k_4, k_5, k_6, k_7\}$, and traverses 2^{48} values of key words (k_2, k_3, k_8). There are $2^{128-48} = 2^{80}$ such groups.

For each key group, set the intermediate value $L_0 = L_2 = 0x0000ffff$, and traverse (k_2, k_8) to compute the input and output values $(\overline{P}, S)$ of the 3.3-round subcipher. The pair (P, C) and its corresponding $(\overline{P}, S)$ satisfy the following equations except $P_l = \overline{P}_l = 0x0000ffff$.

$$P_{r,l} = \overline{P}_{r,l} \oplus (k_3 \lll 5), \tag{5}$$

$$P_{r,r} = \overline{P}_{r,r}. \tag{6}$$

Therefore, given the matched (P, C) and $(\overline{P}, S)$, the corresponding key word k_3 is computed by equation (5).

We demonstrate our attack as follows, which is depicted in Fig. 8 in App. B.

1. For each key group, traverse k_2 and k_8 to compute the values of $(\overline{P},\ S)$ from the values $L_0 = L_2 = 0x0000ffff$ by partial encryption and decryption. We obtain 2^{32} pairs $(\overline{P}(0x0000ffff\|\overline{P}_r), S)$ corresponding to 2^{32} key word pairs (k_2, k_8). Store $(\overline{P}_r, S, k_2, k_8)$ in a hash table T_2 indexed by $\overline{P}_{r,r}$. There are about 2^{16} entries in T_2 with each index. This step needs $2^{80} \times 2^{32} \times 10 \times 4 = 2^{112} \times 40$ Sbox calls.
2. For each $P_{r,r}$, there are 2^{16} (P, C) pairs on average in table T_1. For each pair, compute the intermediate values S_1 by partially decrypting C. There are about $2^{80} \times 2^{16} \times 2^{16} \times 10 = 2^{112} \times 10$ Sbox calls and 2^{96} table T_1 look-ups in this step.
3. For each pair (P, C), search 2^{16} elements $(\overline{P}_r,\ S,\ k_2,\ k_8)$ with $\overline{P}_{r,r} = P_{r,r}$ in table T_2. For every $(\overline{P}_r, S, k_2, k_8)$, calculate $k_3 = (\overline{P}_{r,l} \oplus P_{r,l}) \ggg 5$. Here we know all the 128-bit key. Then we compute the output of the 7-th round by partially decrypting S_1 with 2 Sbox calls, and the input of the 5-th round by partially encrypting S with 8 Sbox calls for each key. It takes $2^{80} \times 2^{16}$ table T_2 look-ups and $2^{128} \times 10$ Sbox calls.
4. Then check whether $(YO_5 \oplus L_3) = (YO_7 \oplus L_7)$ holds by Observation 5. If they are equal, go to the next step. Otherwise we conclude the key is wrong and discard it. We spend $2^{128} \times 9$ Sbox calls in this step.
5. There are about 2^{16} candidates of key words (k_2, k_3, k_8) such that Equation (3) holds. Detect whether the key words $(\overline{K}, k_2, k_3, k_8)$ are right by trail encryptions of another two known plaintext/ciphertext pairs. Go to Step 2. If all the $P_{r,r}$s are traversed, go to Step 1, and test for a new key group. This steps takes $2^{80} \times 2^{16}$ encryptions.

Altogether we spend 19 Sboxes instead of 96 for every key and 2^{97} table lookups, resulting in a time complexity with about $2^{125.67}$ KASUMI computations.

Remark. Furthermore, we extend one more Sbox in $(\overline{P}, S)$, i.e., $S = (F[7{-}15], XI_{4,1,2}, YI_{4,2})$, where $XI_{4,1,2} = S7(F[0-6] \oplus k_3[0-1, 11-15])$. It is obvious that $k_3[2-10]$ does not occur in the computation of $(\overline{P}, S)$ from $L_0 = L_2 = 0x0000ffff$. We also append one more Sbox in (C, S_1), i.e., $S_1 = (\overline{YI}_{8,2}[0-8], YI_{8,2,2}, YI_{8,3})$. $k_3[0-8]$ is independent of the partial decryption from C to S_1. Here, we partition the key space into key groups of $(k_2, k_3[2-8], k_8)$ instead of (k_2, k_3, k_8), and perform the above attack process. Hence it needs to compute in total 17 Sboxes for each key in turn resulting in $2^{125.5}$ encryptions.

4 MITM Attacks on KASUMI with a 64-bit Key

For backwards compatibility, e.g. with 2G-telephony, KASUMI is used with a 64-bit key. We focus on KASUMI with a key of 64 bits in this section. In order to be faster than brute force, a key recovery technique has to require the computations of less than 2^{64} KASUMI encryptions in the worst case. This, together with the fact that the key words are repeated twice as often makes the design of such key recovery algorithms more difficult. Nevertheless, for this situation we give several MITM attacks on KASUMI-64. This is probably the first instance of this line of cryptanalytic work, where the speed-up over brute force is not merely an academic curiosity but can lead to cost-savings in practice. We first give several time/data trade-offs for the attacks. Then we introduce an attack on KASUMI-64-0 with a different key schedule. Finally, we draw some conclusions on key-schedule design in Sect. 4.5.

4.1 The MITM Attack on KASUMI-64 with a Known Plaintext

It is obvious that the subkey $k_4[9-15]$ is not used in the partial encryption from P to S_0, where $S_0 = (\overline{YI}_{1,2}[9-15],\ YI_{1,2,1},\ \overline{YI}_{1,3}[9-15],\ YI_{1,3,1})$, and the subkey $k_3[9-15]$ is not included in the partial decryption from C to S_1, where $S_1 = (\overline{YI}_{8,2}[9-15],\ YI_{8,2,1},\ \overline{YI}_{8,3}[9-15],\ YI_{8,3,1})$ for a known pair (P, C) (see Fig. 4). Then we get two trunks, one is from P to S_0, in which the subkey $k_4[9-15]$ is not involved, the other is from C to S_1, in which the subkey $k_3[9-15]$ does not occur. Then we partition the key space into 2^{50} small groups, and each takes all values of $(k_3[9-15], k_4[9-15])$. Given a known pair (P, C), for each $(k_1, k_2, k_3[0-8], k_4[0-8])$, we do the following steps to find the right key.

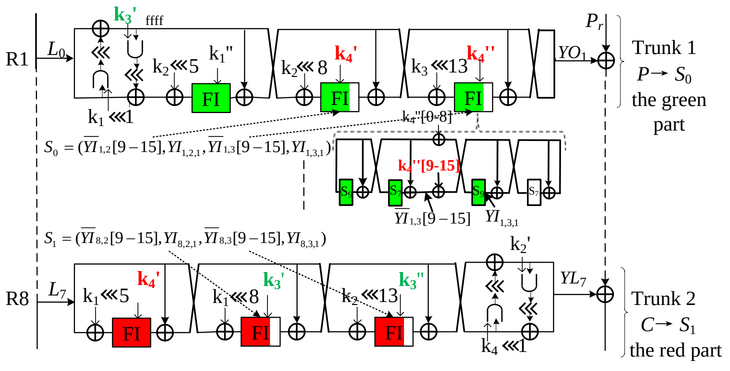


Fig. 4. The two trunks of KASUMI-64 in the known-plaintext attack

1. Traverse $k_3[9-15]$, partially encrypt P to get the intermediate values S_0, and store 2^7 values of S_0 in a hash table indexed with $k_3[9-15]$. It takes $2^7 \times 10$ Sbox calls.
2. Guess $k_4[9-15]$, partially decrypt C to get the intermediate values S_1, which needs $2^7 \times 10$ Sbox calls.

3. Guess $k_3[9-15]$, partially decrypt S_1 to get the output values of the 7-th round, and partially encrypt S_0 corresponding to $k_3[9-15]$ to get the input values of the 5-th round. This step takes $2^{14} \times 10 \times 4$ Sbox calls.
4. Detect whether the equation $YO_5 \oplus L_3 = YO_7 \oplus L_7$ holds by the method in Observation 5 for the input values of the 5-th round and the output values of the 7-th round, which needs $2^{14} \times 9$ Sbox calls.
5. For the subkeys which make the Equation (3) hold, we search the right key by trial encryptions with a known plaintext/ciphertext pairs. Since there are about $2^{-32} \times 2^{64} = 2^{32}$ keys to keep Equation (3) hold, it takes 2^{32} encryptions in this step.

This attack needs $2^{50} \times (2^7 \times (10 + 10) + 2^{14} \times 49)$ Sbox calls in the worst case, which is about $2^{63.03}$ encryptions in total. The ciphertexts of only two known plaintexts are enough, and the memory complexity is 2^7 32-bit words.

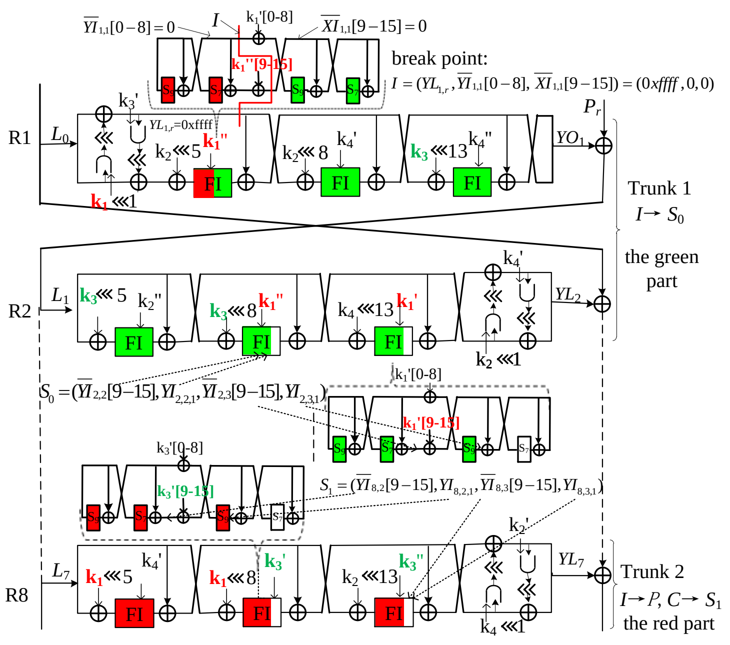
4.2 The MITM Attack on KASUMI-64 with Lower Time Complexity

Key partitioning. For KASUMI-64, select $YL_{1,r} = 0xffff$ to cancel the differences of k_3 in the FL_1 function, and set $I = (YL_{1,r}, \overline{YI}_{1,1}[0-8], \overline{XI}_{1,1}[9-15])$ to obtain two trunks. The first trunk is $I \xrightarrow{\text{partial Enc}} S_0$, where $S_0 = (\overline{YI}_{2,2}[9-15], YI_{2,2,1}, \overline{YI}_{2,3}[9-15], YI_{2,3,1})$. The second trunk includes two parts, $I \xrightarrow{\text{partial Dec}} P$ and $C \xrightarrow{\text{partial Dec}} S_1$, where $S_1 =(\overline{YI}_{8,2}[9-15]$, $YI_{8,2,1}$, $\overline{YI}_{8,3}[9-15]$, $YI_{8,3,1})$. This is depicted in Fig. 5. We notice that the subkey $k_3[9-15]$ is not used in the second trunk when $YL_{1,r} = 0xffff$, and the subkey $k_1[9-15]$ is not intervened in the first trunk, because $\overline{XI}_{9,15} = 0$ holds always for different $k_1[9-15]$. Then we partition the KASUMI-64 key space into 2^{50} groups, and each takes all values of $(k_1[9-15], k_3[9-15])$, and carry out the meet-in-the-middle attack on KASUMI-64.

Data collection. Let $YL_{1,r} = 0xffff$, $\overline{YI}_{1,1}[0-8] = 0$, $\overline{XI}_{1,1}[9-15] = 0$ and the right 32-bit parts P_r of the plaintexts be fixed to a constant, compute all the left 32-bit parts P_l of the plaintexts by partial decrypting the intermediate values by searching all 2^{48} subkeys (k_1, k_2, k_3). By the computer search, there are exact $1119744 \approx 2^{20}$ P_ls in total. For all the plaintexts $P = P_l \| P_r$ found above, query their corresponding ciphertexts and store these plaintext/ciphertext pairs.

Key recovery. For each key group corresponding to $(k_1[0-8], k_2, k_3[0-8], k_4)$, we execute the following attack process to distinguish the right key.

1. Traverse $k_1[9-15]$, compute plaintexts P by partially decrypting the intermediate values $YL_{1,r} = 0xffff$, $\overline{YI}_{1,1}[0-8] = 0$, $\overline{XI}_{1,1}[9-15] = 0$, query the corresponding ciphertexts C, compute the intermediate values S_1 by partial decryption, and store (C_r, S_1) in a hash table indexed with $k_1[9-15]$, which takes $2^7 \times 12$ Sbox calls.
2. Guess $k_3[9-15]$, partially encrypt the intermediate values $YL_{1,r} = 0xffff$, $\overline{YI}_{1,1}[0-8] = 0$, $\overline{XI}_{1,1}[9-15] = 0$ and P_r to get the intermediate values L_1 and S_0. This step needs $6 + 2^7 \times 14$ Sbox calls.
3. Guess $k_1[9-15]$, partially encrypt L_1 and S_0 to get the input values of the 5-th round, and partially decrypt S_1 corresponding to $k_1[9-15]$ to get the output values of the 7-th round. We spend $2^{14} \times 28$ Sbox calls in this step.
4. Detect the equation $YO_5 \oplus L_3 = YO_7 \oplus L_7$ hold or not by the method in Observation 5 for the input values of the 5-th round and the output values of the 7-th round, which needs $2^{14} \times 9$ Sbox calls.
5. For the subkeys such that the Equation (3) holds, we search the right key by trial encryptions with 2 known plaintext/ciphertext pairs. There are about $2^{-32} \times 2^{64}$ keys to make Equation (3) hold, so we speed 2^{32} encryptions in this step.



$k_i'' = k_i + c_{i+4}, i = 0, 1, 2, 3.$

Fig. 5. The two trunks for the MITM attack on KASUMI-64 with lower time complexity

Here we need about 2^7 blocks memory to store the intermediate values in Step 1. We need about $2^{50} \times (6 + 2^7 \times (14 + 12) + 2^{14} \times 37)$ Sbox calls equivalent to $2^{62.63}$ encryptions in total. The data complexity is about 2^{20} chosen plaintexts.

4.3 Trading off Time with Data for MITM Attacks on KASUMI-64

In this section, we give two new trunks to reduce the data complexity by relaxing a little of the time complexity. We choose $(k_1[7-8], k_3[9-15])$ as a key group and the break point $I = (YL_{1,r}, YL_{1,l}[0-6], A)$, where $YL_{1,r} = 0xffff$, $YL_{1,l}[0-6] = 0$, $A = (k_1''[7-8] \| 0^7) \oplus XI_{1,1,1}$, and $A[7-8] = 0$ (see Fig. 6). The first trunk is from I to S_2 by partial encryptions, in which $k_1[7-8]$ is not involved due to $\overline{XI}_{1,1}[7-8] = A[7-8] = 0$, and the second trunk is from I to S_1 with partial decryptions, in which $k_3[9-15]$ is not used as a result of $YL_{1,r} = 0xffff$. Here $S_1 = (\overline{YI}_{8,2}[9-15], YI_{8,2,1}, \overline{YI}_{8,3}[9-15], YI_{8,3,1})$, is the same as that in the Sect. 4.2, and $S_2 = (\overline{YI}_{2,2}[0-8], YI_{2,2,2}, \overline{YI}_{2,3}[0-8], YI_{2,3,2})$[6].

On basis of Observation 2, we know $KO_{1,1}[0-6](k_2[0-1, 11-15])$ is independent of the computations of the intermediate value $\overline{XI}_{1,1}[7-8]$ in the $FI_{1,1}$ function, and $k_1[7-8]$ has no relation with the intermediate value $XI_{1,1}[0-6]$ in the computations of plaintexts from the break point by partial decryption. Furthermore $k_1[7-8]$ is not involved in the computations of the intermediate value $YL_{1,l}[0-6]$, so we assign $YL_{1,l}[0-6] = 0$ to avoid the impacts of $k_1[0-5, 15]$ and $k_2[0-1, 11-15]$ in the computations of plaintexts from the break point I by partial decryption.

Data collection. Let $YL_{1,r} = 0xffff$, $YL_{1,l}[0-6] = 0$, $A = 0$ and the right 32-bit parts P_r of plaintexts are assigned to a constant, compute all the left 32-bit parts of plaintexts by partial decrypting the intermediate values and searching all the subkeys $(k_1[6-14], k_2[2-10], k_3)$. By the computer search, there are 1152 plaintexts in total. For all the plaintexts computed above, query their corresponding ciphertexts and store the 1152 plaintext/ciphertext pairs.

[6] Since we substitute $k_1[7-8]$ for $k_1[9-15]$ in the key group, which makes the end of the first trunk S_2 a little different from S_0 in Sect. 4.2.

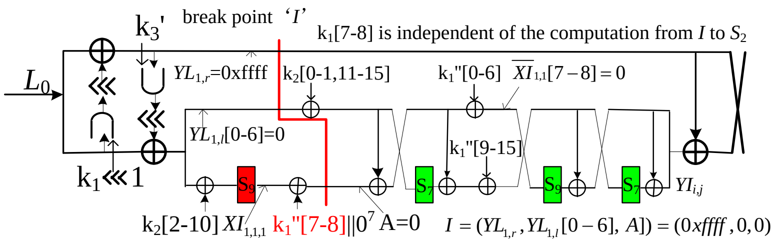


Fig. 6. The new break point of the MITM attack on KASUMI-64

Key recovery. For each key group with 2^9 values of $(k_1[7-8], k_3[9-15])$, we use the similar key recovery process including 5 steps as that in Sect. 4.2 with $k_1[7-8]$ instead of $k_1[9-15]$ and S_2 instead of S_0. Therefore the time complexity in Step 1 is $2^2 \times 12$ Sbox calls as a result of traversing $k_1[7-8]$. Step 2 still needs $6 + 2^7 \times 14$ Sbox calls. We spend $2^9 \times 28$ and $2^9 \times 9$ Sbox calls in Step 3 and Step 4, respectively. Totally, we need about $2^{55} \times (6 + 2^7 \times 14 + 2^9 \times 37)$ Sbox calls, which is equivalent to $2^{62.75}$ encryptions. The data complexity is 1152 chosen plaintexts and 2^2 blocks memory to store intermediate values in Step 1.

4.4 The MITM Attack on KASUMI-64-0

In this section we give a MITM attack on KASUMI-64-0, the least 64 bits of which are set to zero. Our cryptanalysis method of KASUMI in Sect. 3.2 is applicable to attack KASUMI-64-0. Because in our cryptanalysis we use the key groups of $(k_1[9-15], k_2[0-8])$. The key words $k_4 = 0, k_5 = 0, k_6 = 0, k_7 = 0$ have no impact on the attack process. If the method in Sect. 3.2 is applied to attack the KASUMI-64-0, the time complexity is about $2^{61.98}$ encryptions and the data complexity is a chosen plaintext. This attack is also converted to a known-plaintext attack, which succeeds with 128 known plaintexts and $2^{61.98}$ encryptions.

4.5 Lessons on Key-Schedule Design

By analyzing two versions of the KASUMI-64 we shed some light on key schedule design aiming to support multiple key sizes, a to-date extremely ad-hoc part of cipher design. Should key material that is smaller than the master key be repeated to fill up the master key, or should it rather be padded with a constant? Both natural options can also be found in other cipher designs. E.g. Serpent is designed with a key schedule supporting 256-bit master keys, and other versions of the Serpent key schedule are then specified with parts of them set to zero. Also for Hierocrypt-192, a 64-bit constant is used to make it fit into the 256-bit key schedule. For Hierocrypt-128, both options are combined. In Threefish, three distinct key schedules for 128, 192, or 256-bit are designed, but also intermediate key sizes are supported, and specified via padding with a constant rather than repeating key material.

For the first time we can give concrete evidence that the former, i.e., repeating key material, is preferable, by giving an improved attack on the later for the case of KASUMI-64. Even with respect to a single attack vector like the MITM attack considered in this paper this is not obvious, as repeating key material also makes it more likely to have high probably related-key differentials via local collisions, which can also be used in speed-up MITM key search. We give an MITM attack on the padding variant of KASUMI-64 in Sect. 4.4, which is a factor 4 rather than a factor 2 faster than brute-force. The reason is that re-introducing master-key material in subkeys increases the diffusion of the master key bits, which in turn affects the performance of our new MITM attacks.

5 Conclusion

In this paper, we present new key recovery methods for the full KASUMI block cipher. These are the first results on the full KASUMI with a single key. Tab. 1 and Tab. 2 summarize our results along with the best previous known results for KASUMI in single key setting.

Bruteforce-like cryptanalysis gets renewed attention since their application to full AES. It is a new way to better understand the real security offered by a cipher, and deserves attention. Common criticisms of recent bruteforce-like cryptanalytic results are (1) that they achieve only a small improvement over brute-force search, often much less than a factor of 2, (2) those small speed-ups are irrelevant with key sizes of 128 bits or more, and (3) in addition to that this often comes with impractically high data complexities. Our work on KASUMI is unique in the sense that it can address and refute all of those criticisms at the same time:

- We only give key recovery methods with speed-ups of a factor 2 or better. These speed-ups also remain speed-ups, when compared to clever brute-force optimizations that have recently been described by Biham et al. [7].
- The master key can be only 64 bits in real-world use, and achieved speed-ups apply to this case as well.
- The resulting methods have small enough data complexity (down to unicity distance) to allow for a practical collection of ciphertexts in actual use of communication networks.

Of independent interest, we give evidence that repeating key material to fill a larger master key as input to a key schedule is preferable over padding with a constant. The two attacks on KASUMI-64 and the earlier variant of it, together with some data/time trade-offs, give a set of data points in support of this conclusion. We hope that this discussion inspires more work on key-schedule design issues, which is compared to other aspects of cipher design notoriously under-researched.

Table 1. Summary of the new key recovery methods on full KASUMI. CP refers to the number of chosen plaintexts, KP refers to the number of known plaintexts, and Enc refers to the number of encryptions for the corresponding KASUMI version.

Attack Type	Rounds	Version	Data	Time	Source
Meet-in-the-Middle Attack	8	128-bit	1 CP + 1 KP	$2^{125.98}$ Enc	Sect. 3.2
Meet-in-the-Middle Attack	8	128-bit	128 KP	$2^{125.98}$ Enc	Sect. 3.2
Meet-in-the-Middle Attack	8	128-bit	2^{32} CP	$2^{125.67}$ Enc	Sect. 3.3
Meet-in-the-Middle Attack	8	128-bit	2^{32} CP	$2^{125.5}$ Enc	Sect. 3.3
Meet-in-the-Middle Attack	8	64-bit	1 KP	$2^{63.03}$ Enc	Sect. 4.1
Meet-in-the-Middle Attack	8	64-bit	2^{20} CP	$2^{62.63}$ Enc	Sect. 4.2
Meet-in-the-Middle Attack	8	64-bit	1152 CP	$2^{62.75}$ Enc	Sect. 4.3
Meet-in-the-Middle Attack	8	64-bit variant	1 CP	$2^{61.98}$ Enc	App. 4.4
Meet-in-the-Middle Attack	8	64-bit variant	128 KP	$2^{61.98}$ Enc	App. 4.4

Table 2. Summary of earlier key recovery methods on reduced KASUMI.

Attack Type	Rounds	Version	Data	Time	Source
Higher-Order Differential	5	128-bit	$2^{28.9}$ CP	$2^{31.2}$ Enc	[33]
Impossible Differential	6	128-bit	2^{55} CP	2^{100} Enc	[24]
Impossible Differential	7(2-8)	128-bit	$2^{52.5}$ CP	$2^{114.3}$ Enc	[21]
Impossible Differential	7(1-7)	128-bit	2^{62} KP	$2^{115.8}$ Enc	[21]

References

1. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Biclique Cryptanalysis of the PRESENT and LED Lightweight Ciphers, Cryptology ePrint Archive, Report 2012/591.
2. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 82–98. Springer, Heidelberg (2008)
3. Aoki, K., Sasaki, Y.:Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
4. Barkan, E., Biham, E., Keller, N.: Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 600–616. Springer, Heidelberg (2003)
5. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124-138. Springer, Heidelberg (1999)
6. Biham, E., Dunkelman, O., Keller, N.: A Related-Key Rectangle Attack on the Full KASUMI. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 443–461. Springer, Heidelberg (2005)
7. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New Data-Efficient Attacks on Reduced-Round IDEA. Available at http://eprint.iacr.org/2011/417, 2011.
8. Blunden, M., Escott, A.: Related Key Attacks on Reduced Round KASUMI. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 277–285. Springer, Heidelberg (2002)
9. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011, LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
10. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010, LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2010)
11. Çoban, M., Karakoç, F., Boztaş, Ö.: Biclique Cryptanalysis of TWINE, CANS 2012, to appear.
12. Dunkelman, O., Keller, N., Shamir, A.: A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In: Rabin, T. (ed.) CRYPTO 2010. LNCS 6223, pp. 393–410. Springer, Heidelberg (2010)
13. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Springer, Heidelberg (2010)
14. Diffie, W., Hellman, M. E.: Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10 (6): 74–84.
15. Demirci, H., Selçuk. A.: A Meet-in-the-Middle Attack on 8-round AES. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 116–126. Springer, Heidelberg (2008)
16. Dunkelman, O., Sekar, G., Preneel, B.: Improved Meet-in-the-Middle Attacks on Reduced Round DES. In: Srinathan, K., Ranga, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 86–100. Springer, Heidelberg (2007)
17. Demirci, H., Taskin, I., Coban, M., Baysal, A.: Improved Meet-in-the-Middle Attacks on AES. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp.144–156. Springer, Heidelberg (2009)
18. Guo,J., Ling,S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
19. Isobe, T.: A Single-Key Attack on the Full GOST Block Cipher. In A. Joux (ed.) FSE 2011. LNCS, vol. 6733, pp. 290–305. Springer, Heidelberg (2011)
20. Jeong, K., Kang, H., Lee, C., Sung, J., Hong, S.: Biclique Cryptanalysis of Lightweight Block Ciphers PRESENT, Piccolo and LED, Cryptology ePrint Archive, Report 2012/621.
21. Jia, K., Li,L., Rechberger, C., Chen, C., Wang, X.: Improved Cryptanalysis of the Block Cipher KASUMI. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 222–233. Springer, Heidelberg (2012)
22. Khovratovich, D., Leurent, G., Rechberger, C.: Narrow Bicliques: Cryptanalysis of Full IDEA. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 392–410, Springer, Heidelberg (2012)
23. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol, 7549, pp. 244–263, Springer, Heidelberg (2012)
24. Kühn, U.: Cryptanalysis of Reduced-Round MISTY. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 325–339. Springer, Heidelberg (2001)
25. Kühn, U.: Improved Cryptanalysis of MISTY1. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 61-75. Springer, Heidelberg (2002)
26. Matsui, M.: Block Encryption Algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 64–74. Springer, Heidelberg (1997)
27. Nohl, K.: Attacking Phone Privacy, Black Hat, Las Vegas, 2010.
28. Nohl, K., Munaut, S.: Wideband GSM Sniffing, 27th Chaos Communication Congress, Berlin, 2010.
29. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, V3.1.1 (2001)

30. Paget, C., Nohl, K.: GSM: SRSLY?, 26th Chaos Communication Congress, Berlin, 2009.
31. 3rd Generation Partnership Project, Technical Specification Group Services and ystem Aspects, 3G Security, Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 1: A5/3 and GEA3 Specifications, V6.2.0 (2003)
32. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster than Exhaustive Search. In: Cramer, R. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
33. Sugio, N., Aono, H., Hongo, S., Kaneko, T.: A Study on Higher Order Differential Attack of KASUMI. IEICE Transactions 90-A(1), pp. 14-21 (2007)
34. Universal Mobile Telecommunications System (UMTS), 3G Security; Cryptographic Algorithm Requirements, 3G TS 33.105 version 3.2.0 Release 1999 (2000)
35. Universal Mobile Telecommunications System (UMTS), LTE; Cryptographic Algorithm Requirements, 3GPP TS 33.105 version 10.0.0 Release 10 (2011)

A Proof of Observation 5

This section describes the proof of Observation 5. We use the partial matching technique and early abort technique to check whether the equation $YO_5 \oplus L_3 = YO_7 \oplus L_7$ holds for a guess key. Because the equation can be represented as 32 parallel bit equations, and some bit equations are much easier to be detected owing to the Feistel structure with two Sboxes of different length for FI functions.

Proof. Concerning the FI function, it takes 2 Sbox calls to compute $YI_{i,j}[7-8]$ by the given input $XI_{i,j}$ and $KI_{i,j}$ [7]. Computing $YO_{i,l}[7-8]$ takes 4 Sbox computations for given XO_i, KO_i and KI_i.

Consequently, we spend 8 Sbox calls to test whether the equation $YO_{5,l}[7-8] \oplus L_{3,l}[7-8] = YO_{7,l}[7-8] \oplus L_{7,l}[7-8]$ holds, the detail computations of which are given in the following.

1. The inputs of 4 FI functions $XI_{5,1}$, $XI_{5,2}$, $XI_{7,1}$ and $XI_{7,2}$ are computed by XOR the inputs of the FO functions and the corresponding subkeys.
2. Compute $YO_{5,l}[7-8]$ on the basis of $XI_{5,1}$ and $XI_{5,2}$ by partial encryption, which takes 4 Sbox calls.

$$\begin{aligned}
\overline{YI}_{5,1}[0-8] &= S9(XI_{5,1}[7-15]) \oplus XI_{5,1}[0-6],\\
YI_{5,1,1} &= S9(\overline{YI}_{5,1}[0-8] \oplus KI_{5,1}[0-8]),\\
\overline{YI}_{5,2}[0-8] &= S9(XI_{5,2}[7-15]) \oplus XI_{5,2}[0-6],\\
YI_{5,2,1} &= S9(\overline{YI}_{5,2}[0-8] \oplus KI_{5,2}[0-8]),\\
YO_{5,l}[7-8] &= (YI_{5,1,1} \oplus YI_{5,2,1} \oplus YL_{5,l})[7-8].
\end{aligned}$$

3. Compute $YO_{7,l}[7-8]$ with $XI_{7,1}$ and $XI_{7,2}$ by partial decryption, with a cost of 4 Sbox calls.

$$\begin{aligned}
\overline{YI}_{7,1}[0-8] &= S9(XI_{7,1}[7-15]) \oplus XI_{7,1}[0-6],\\
YI_{7,1,1} &= S9(\overline{YI}_{7,1}[0-8] \oplus KI_{7,1}[0-8]),\\
\overline{YI}_{7,2}[0-8] &= S9(XI_{7,2}[7-15]) \oplus XI_{7,2}[0-6],\\
YI_{7,2,1} &= S9(\overline{YI}_{7,2}[0-8] \oplus KI_{7,2}[0-8]),\\
YO_{7,l}[7-8] &= (YI_{7,1,1} \oplus YI_{7,2,1} \oplus YL_{7,l})[7-8].
\end{aligned}$$

4. Then compare $(YO_{5,l}[7-8] \oplus YO_{7,l}[7-8]$ with $(L_{3,l} \oplus L_{7,l})[7-8]$. If they are equal, we check other bit equations. Otherwise we conclude $YO_5 \oplus L_3 \neq YO_7 \oplus L_7$.

[7] Given an input $XI_{i,j}$ of the $FI_{i,j}$ function, $YI_{i,j}[7-8] = S9(S9(XI_{i,j}[7-15]) \oplus (00\|XI_{i,j}[0-6]) \oplus KI_{i,j}[0-8])[7-8]$, which costs 2 Sbox calls.

Compute $YO_{5,l}[0-6]$ and $YO_{7,l}[0-6]$ as follows, which takes $2^{-2} \times 4 = 1$ Sbox calls.

$$\begin{aligned}
YI_{5,1}[0-6] &= S7(XI_{5,1}[0-6]) \oplus \overline{YI}_{5,1}[0-6] \oplus KI_{5,1}[9-15] \oplus YI_{5,1,1},\\
YI_{5,2}[0-6] &= S7(XI_{5,2}[0-6]) \oplus \overline{YI}_{5,2}[0-6] \oplus KI_{5,2}[9-15] \oplus YI_{5,2,1},\\
YO_{5,l}[0-6] &= YI_{5,1}[0-6] \oplus YI_{5,2}[0-6],\\
YI_{7,1}[0-6] &= S7(XI_{7,1}[0-6]) \oplus \overline{YI}_{7,1}[0-6] \oplus KI_{7,1}[9-15] \oplus YI_{7,1,1},\\
YI_{7,2}[0-6] &= S7(XI_{7,2}[0-6]) \oplus \overline{YI}_{7,2}[0-6] \oplus KI_{7,2}[9-15] \oplus YI_{7,2,1},\\
YO_{7,l}[0-6] &= YI_{7,1}[0-6] \oplus YI_{7,2}[0-6].
\end{aligned}$$

Then compare $(YO_{5,l}[0-8] \oplus YO_{7,l}[0-8])$ with $(L_{3,l} \oplus L_{7,l})[0-8]$. If they are equal, we check the other bit equations. Otherwise we conclude $YO_5 \oplus L_3 \neq YO_7 \oplus L_7$.

Compute $YO_{5,l}$ and $YO_{7,l}$ using the values obtained in the above steps, which needs extra $2^{-9} \times 4$ Sbox calls, and then detect the equation $YO_{5,l} \oplus L_{3,l} = YO_{7,l} \oplus L_{7,l}$ hold or not.

If the above equation does not hold, we conclude $YO_5 \oplus L_3 \neq YO_7 \oplus L_7$. Otherwise we compute $YO_{5,r}$ and $YO_{7,r}$, which costs extra $2^{-16} \times 8$ Sbox calls. If the equation $YO_{5,r} \oplus L_{3,r} = YO_{7,r} \oplus L_{7,r}$ holds, then $YO_{5,l} \oplus L_{3,l} = YO_{7,l} \oplus L_{7,l}$. Otherwise $YO_{5,l} \oplus L_{3,l} \neq YO_{7,l} \oplus L_{7,l}$

Altogether we need 9 Sbox calls to detect the equation $YO_{5,l} \oplus L_{3,l} = YO_{7,l} \oplus L_{7,l}$ hold or not. □

B Some Tables and Figures for the Cryptanalysis of KASUMI

We list some tables and figures used in this paper.

Table 3. The key schedule of KASUMI

Round	$KL_{i,1}$	$KL_{i,2}$	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$
1	$k_1 \lll 1$	k'_3	$k_2 \lll 5$	$k_6 \lll 8$	$k_7 \lll 13$	k'_5	k'_4	k'_8
2	$k_2 \lll 1$	k'_4	$k_3 \lll 5$	$k_7 \lll 8$	$k_8 \lll 13$	k'_6	k'_5	k'_1
3	$k_3 \lll 1$	k'_5	$k_4 \lll 5$	$k_8 \lll 8$	$k_1 \lll 13$	k'_7	k'_6	k'_2
4	$k_4 \lll 1$	k'_6	$k_5 \lll 5$	$k_1 \lll 8$	$k_2 \lll 13$	k'_8	k'_7	k'_3
5	$k_5 \lll 1$	k'_7	$k_6 \lll 5$	$k_2 \lll 8$	$k_3 \lll 13$	k'_1	k'_8	k'_4
6	$k_6 \lll 1$	k'_8	$k_7 \lll 5$	$k_3 \lll 8$	$k_4 \lll 13$	k'_2	k'_1	k'_5
7	$k_7 \lll 1$	k'_1	$k_8 \lll 5$	$k_4 \lll 8$	$k_5 \lll 13$	k'_3	k'_2	k'_6
8	$k_8 \lll 1$	k'_2	$k_1 \lll 5$	$k_5 \lll 8$	$k_6 \lll 13$	k'_4	k'_3	k'_7

$x \lll i$: x rotates left by i bits.

$k'_i = k_i \oplus c_i$, where the c_is are fixed constants.

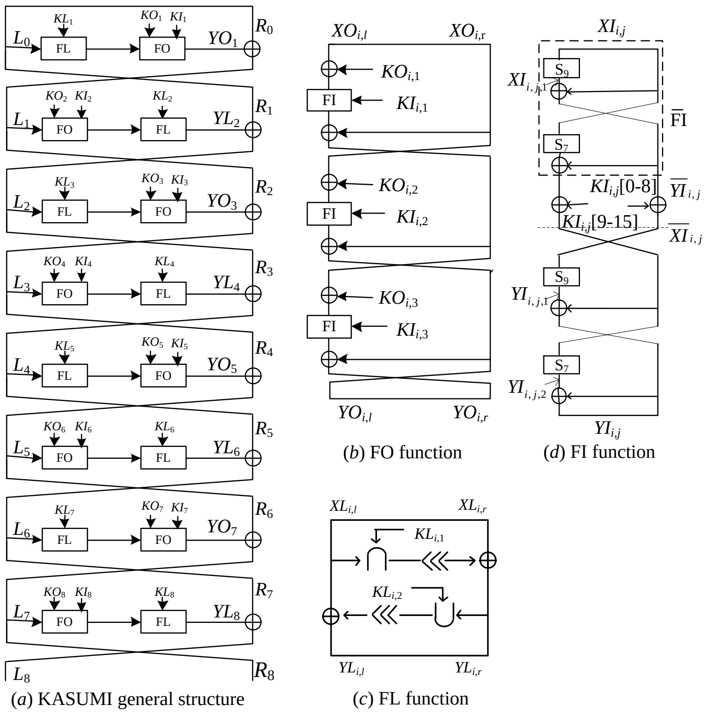


Fig. 7. The structure and building blocks of the block cipher KASUMI

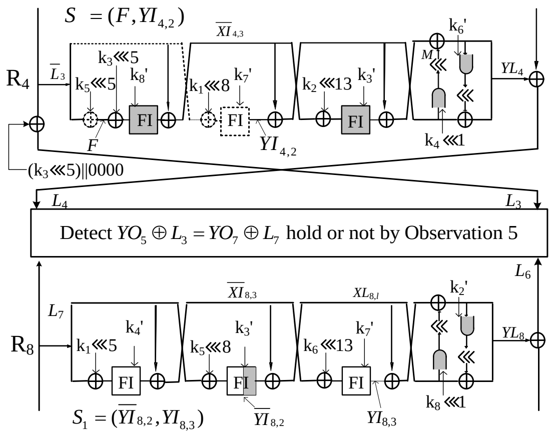


Fig. 8. Detect whether $YO_5 \oplus L_3 = YO_7 \oplus L_7$ holds using Observation 5