A Meet-in-the-Middle Attack on the Full KASUMI*

Keting Jia[1], Honbo Yu[2], and Xiaoyun Wang[1,3]**

[1] Institute for Advanced Study, Tsinghua University, Beijing 100084, China
ktjia, xiaoyunwang@mail.tsinghua.edu.cn
[2] Department of Computer Science, Tsinghua University, Beijing 100084, China
yuhongbo@mail.tsinghua.edu.cn
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China

Abstract. KASUMI is a block cipher which consists eight Feistel rounds with a 128-bit key. The confidentiality and integrity of UMTS, GSM and GPRS mobile communications systems depend heavily on the security of the block cipher KASUMI. We find that the 16-bit subkey k_3 has a linear relationship with some special plaintexts and the corresponding outputs of the 3-rd round function respectively. Considering this property, we explore a meet-in-the-middle attack on the full KASUMI by seperating the subkey k_3 in the process of searching the secret key. At the same time, some collision properties and table-lookups are applied to reduce the time complexity. With 2^{32} choose plaintexts, our attack needs $2^{125.8}$ encryptions with 2^{48} memory to recover all the key. The time complexity can be improved to 2^{125} encryptions with optimal implementation. Although the attack is a slight advantage over the exhaustive search, it is the first attack on the full version of KASUMI with a single key.

Key words: KASUMI, Meet-in-the-Middle Attack, Block Cipher, Cryptanalysis

1 Introduction

KASUMI which is a 64-bit block cipher with a 128-bit key, was designed for 3GPP (3rd Generation Partnership Project, which is the body standardizing the next generation of mobile telephony) as the basis of confidentiality and integrity algorithms by ETSI SAGE [19]. Now KASUMI is used in UMTS, GSM and GPRS mobile communications systems, in which the security of the confidentiality (f8) and integrity algorithms (f9) relies on the security of KASUMI [20].

KASUMI consists of eight Feistel rounds with 128-bit keys, which is slightly modified for optimal hardware performance from the block cipher MISTY1(which was presented by Matsui at FSE 1997 [18]) by simplifying the key schedule algorithm and doing a little changes in the round function. MISTY1 is an eight-round Feistel type block cipher with a 64-bit block and a 128-bit key. Its recursive structure, as well as

* Supported by 973 Project (No.2007CB807902), the National Natural Science Foundation of China (Grant No.60931160442), Tsinghua University Initiative Scientific Research Program (2009THZ01002) and China Postdoctoral Science Foundation(20110490442).

** Corresponding author

the added FL layers makes the algorithm resist against some cryptanalytic techniques. MISTY1 has provable security properties against various types of attacks, and no attack is known on its full version. Dunkelman and Keller presented an improved impossible differential attack on 6-round reduced variant of the 8-round MISTY1, and the time complexity is more than 2^{123} encryptions [7]. Tsunoo et al. introduced a higher order differential attack on 7-round MISTY1, which can recover a partial subkey with a data complexity $2^{54.1}$ and a computational complexity $2^{120.7}$ [22].

Because of the importantance of KASUMI, there are some cryptanalytic results deserved to be paied attention to. In the single-key setting the best previous result is an impossible differential attack on a 6-round version of the cipher presented by Kühn [17]. In the related-key setting, Blunden and Escott introduced a related-key differential attack of KASUMI reduced to 6 rounds [4]. Later, Biham et al introduced related-key boomerang and rectangle attacks to the full 8-round KASUMI, which need $2^{78.7}$ and $2^{76.1}$ encryptions, respectively [3]. At Crypto 2010, Dunkelman et al. proposed a practical related-key attack on the full KASUMI by using a new strategy named sandwich attack [8], which is a formal extension of boomerang attack. As far as we know there is no attack on the full 8-round KASUMI without related-key setting.

In this paper we apply the meet-in-the-middle attack on the full 8-round KASUMI with a slight advantage than exhaustive search. With 2^{32} choose plaintexts, our attack needs $2^{125.8}$ encryptions and 2^{48} memory to recover the secret key.

Meet-in-the-Middle Attack was first introduced by Diffie and Hellman to attack double encryption of the block cipher DES [10]. Variants have also been developed, for example unbalanced meet-in-the-middle attacks. In the cryptanalysis of block ciphers, the meet-in-the-middle attack was used to attack the KTANTAN, reduced round DES and reduced round AES [6, 12, 15, 23]. In some cryptanalysis, the meet-in-the-middle framework was used in the integral attacks [9, 11, 13]. Although the meet-in-the-middle attack has a history of more than 30 years, it is a method that gets less attention than differential, linear, impossible differential cryptanalysis. However, since the meet-in-the-middle attack was introduced to attack hash functions in 2008, which has been improved with many techniques to carry out the preimage attack on the hash function MD5, reduced version of SHA-1 and SHA-2 etc. [1,2,14,21]. These techniques include splice-and-cut framework, initial structure, partial match etc. Khovratovich et al. introduced biclique cryptanalysis originating from the initial structure in the hash function cryptanalysis, which led to the preimage attack on 50 rounds of SHA-512, and the first attack on round-reduced Skein hash function [16]. Later, the biclique technique was introduced to cryptanalyze block ciphers by Bogdanov et al.. They gave the first attack on the full AES-128/-192/-256 [5].

In this paper, We find that when the inputs of the 1-st and 3-rd round function are assigned 0x0000ffff, 16-bit subkey k_3 is linear with the corresponding plaintexts and outputs of the 3-rd round function respectively. Considering this property, we use some collision properties to decrease the time complexity. Besides, some table-lookups are used to improve the efficiency of our attack. The key recovery attack needs 2^{48} memory, while the time complexity is $2^{125.8}$ encryptions, and the data complexity is 2^{32} chosen plaintexts. Some optimal implementation of our attack can reduce the time complexity

to 2^{125} encryptions. Although the attack is a little faster than exhaustive search, it is the first attack on the full version of KASUMI with single key.

The paper is organized as follows: Section 2 describes the block cipher KASUMI in details. Section 3 presents meet-in-the-middle attack on the full KASUMI. Some optimal implementation is shown in Section 4 to speed up the efficiency of our attack. We summarize the paper in Section 5.

2 Description of KASUMI

KASUMI is a block cipher with 64-bit block and 128-bit key. It was a slightly modification of MISTY1 for optimized hardware implementations. Therefore, most of the components of KASUMI are similar to the corresponding components of MISTY1. We give a brief description of KASUMI in the following.

Key Schedule. In order to make the hardware significantly smaller and to reduce key set-up time, the key schedule of KASUMI is much simpler than the original key schedule of MISTY1. The 128-bit key K is divided into eight 16-bit words: k_1,k_2, ..., k_8, i.e., $K = (k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8)$. Each k_i is used to compute $k'_i = k_i \oplus C_i$, where the C_is are fixed constants (we omit these from the paper, and refer the reader to [19]). In each round, eight words are used as the round subkey, which is made up of three parts KL_i, KO_i and KI_i. Here, $KL_i = (KL_{i,1}, KL_{i,2})$, $KO_i = (KO_{i,1}, KO_{i,2}, KO_{i,3})$ and $KI_i = (KI_{i,1}, KI_{i,2}, KI_{i,3})$, which are linearly derived from the secret key K. We summarize the details of the key schedule of KASUMI in Table 1.

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 rotate left by i bites

Table 1. The Key Schedule of KASUMI

Encryption. The KASUMI is an eight round Feistel type cipher. Each round is made up of an FL function and an FO function. In odd numbered rounds the FL function precedes the FO function, whereas in even numbered rounds the FO function precedes the FL function. See Fig.1.

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

$$L_i = FO_(FL(L_{i-1}, KL_i), KO_i, KI_i) \oplus R_{i-1},$$
$$R_i = L_{i-1},$$

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

$$L_i = FL_(FO(L_{i-1}, KO_i, KI_i), KL_i) \oplus R_{i-1},$$
$$R_i = L_{i-1}.$$

Here, $L_0||R_0$ is the plaintext, and the $L_8||R_8$ is the ciphertext.

The FL function is linear for a fixed key. Let the input of the FL function be $X = X_l||X_r$ and $KL_i = (KL_{i,1}, KL_{i,2})$ and the output be $Y = Y_l||Y_r$, then the FL function is defined as

$$Y_r = ((X_l \wedge KL_{i,1}) \lll 1) \oplus X_r, \tag{1}$$
$$Y_l = ((Y_r \vee KL_{i,2}) \lll 1) \oplus X_l, \tag{2}$$

where X and Y are 32-bit integers, ' $\wedge$ ' denotes bitwise AND, ' $\vee$ ' denotes bitwise OR, and '$x \lll i$' denotes x rotate left by i bits.

The FO function provides the non-linear property in each round, which is another three-round Feistel structure made up from three FI functions and key mixing stages. The FO function is depicted in Fig.1. There is a 96-bit subkey in FO function of each round(48 subkey bits used in the FI functions and 48 subkey bits in the key mixing stages). Let the input of the FO function be $X = X_l||X_r$, $KO_i = (KO_{i,1}, KO_{i,2}, KO_{i,3})$, $KI_i = (KI_{i,1}, KI_{i,2}, KI_{i,3})$ and the output be $Y = Y_l||Y_r$, then the FO function is defined as

$$X_r^1 = FI((X_l \oplus KO_{i,1}) \lll 1, KI_{i,1}) \oplus X_r, \;\; X_l^1 = X_r,$$
$$X_r^2 = FI((X_l^1 \oplus KO_{i,2}) \lll 1, KI_{i,2}) \oplus X_r^1, \; X_l^2 = X_r^1,$$
$$Y_r = FI((X_l^2 \oplus KO_{i,3}) \lll 1, KI_{i,3}) \oplus X_r^2, \; Y_l = X_r^2.$$

The FI functions use two S-boxes S7 and S9, where S7 is a 7-bit to 7-bit permutation and S9 is a 9-bit to 9-bit permutation, the input is 16-bit data x and an additional 16-bit subkey $KI_{i,j}$. In order to abbreviate the FI function, we define $\overline{FI}$ with 16-bit input x and 16-bit output y in the following, which is half of FI function. The structure of FI and $\overline{FI}$ is depicted in Fig.1. $y = \overline{FI}(x)$ is defined as

$$y[0-8] = S9(x[7-15]) \oplus x[0-6],$$
$$y[9-15] = S7(x[0-6]) \oplus y[0-6],$$

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

$$FI(x, KI_{i,j}) = \overline{FI}(\overline{FI}(x) \oplus (KI_{i,j} \lll 7)) \lll 9.$$

3 The Meet-in-the-Middle Attack on KASUMI

Although the meet-in-the-middle attack has been introduced more than 30 years in the cryptanalysis of block cipher, the approach caught less attention than the differential,

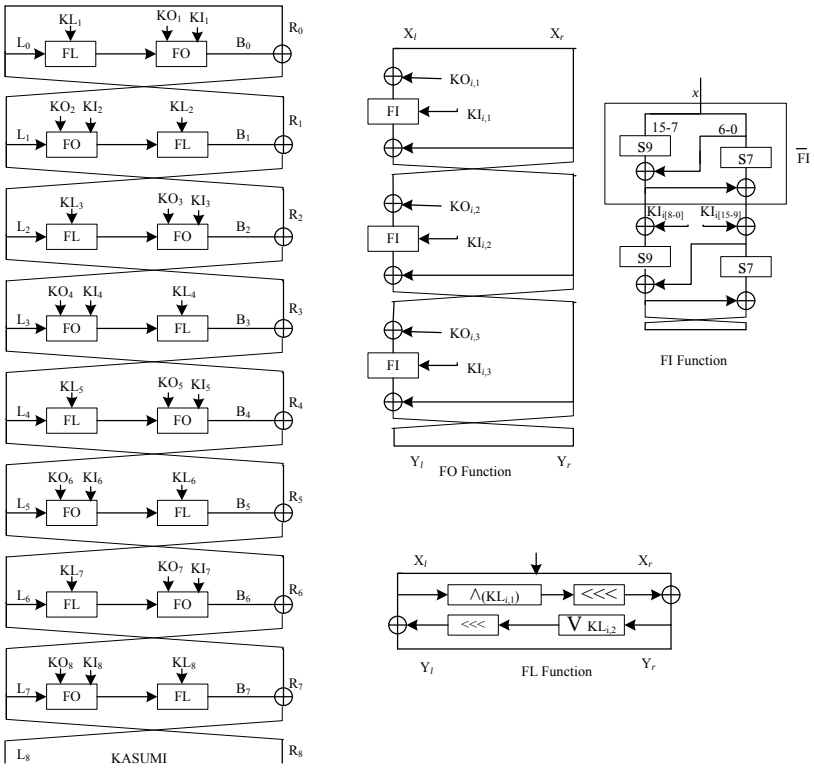

Fig. 1. The Structure of the Block Cipher KASUMI

linear and impossible differential method etc. in the cryptanalysis of block ciphers. However, the method may threaten the security of the full AES algorithm by combining the biclique method.

Inspired by the full-round cryptanalysis of AES, we propose a meet-in-the-middle attack on the full KASUMI. Although biclique structure is not used in our attack, similar idea, i.e., reducing the time complexity by dividing the algorithm into two parts including the initialization (bicique) part and the matching part, leads to a key recovery attack which is 4 times faster than exhaustive search. Our attack also follows the strategy of initial structure and partial matching, which is widely used in the preimage attacks of hash functions.

Let $K = \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8\}$. The plaintext $P = P_l||P_r$, and the ciphertext $C = C_l||C_r$. Let $L_i = L_{i,l}||L_{i,r}$ be the input of $(i+1)$-th round function. Denote $B_i = R_i \oplus L_{i+1} = B_{i,l}||B_{i,r}$, as the output of $(i+1)$-th round function, where $i = 0, 1, ..., 7$ (See Fig.1).

Property 1. Let the inputs of the 1-st round function and 3-rd round function L_0 and L_2 be $0x0000ffff$, compute the plaintext P backward and compute the internal state after the first FI function of the 4-th round function $S = (L_2, E, F, \overline{L}_3)$, where E is the output of the first FI function of the 4-th round function, $F = \overline{L}_{3,l} \oplus KO_{4,1}$, $\overline{L}_3 = L_3 \oplus (KO_{2,1}||0000)$, and L_3 is the input of the 4-th round function. The subkey k_3 is not involved in the state S, and the subkey k_3 has a linear relationship with the plaintext P and L_3 respectively.

Proof. By the equations (1) and (2), it is easy to know that when the input of FL is $0x0000ffff$, the output of FL must be $0xffffffffff$, no matter what values the key KL_i are assigned.

Because k_3 is in the subkey KL_1 and KL_3. So, the output values B_0 and B_2 of the 1-st and 3-rd round functions have no relation with the subkey k_3, where

$$B_0 = FO(0xffffffffff, KO_1, KI_1), B_2 = FO(0xffffffffff, KO_3, KI_3).$$

By the definition of the KASUMI, we know the output the 2-nd round function

$$B_1 = L_0 \oplus L_2 = 0x0000ffff \oplus 0x0000ffff = 0.$$

Then inverting the input of the 2-nd round function, L_1 can be obtained,

$$L_1 = FO^{-1}(FL^{-1}(B_1, KL_2), KO_2, KI_2).$$

L_1 is linear with the subkey k_3. Then, we let $\overline{L}_1 = L_1 \oplus (KO_{2,1}||0000)$. Thus, $\overline{L}_1$ is independent of the subkey k_3.

The elements E, F and $\overline{L}_3$ of the state S are described as

$$\begin{aligned} E &= FI(B_{2,r} \oplus \overline{L}_{1,r}, KI_{4,2}) \oplus KO_{4,2}, \\ \overline{L}_3 &= L_3 \oplus (KO_{2,1}||0000) = B_2 \oplus \overline{L}_1, \\ F &= B_{2,l} \oplus \overline{L}_{1,l} \oplus KO_{4,1}. \end{aligned}$$

Therefore, the state S does not include the subkey k_3. However, $L_3 = B_2 \oplus L_1$ has a linear relationship with the subkey k_3.

The plaintext is computed as

$$P_l = L_0 = 0x0000ffff,$$
$$P_r = B_0 \oplus L_1 = B_0 \oplus \overline{L}_1 \oplus (KO_{2,1}||0000).$$

Here, $B_0 \oplus \overline{L}_1$ consists of all the subkey except k_3, and $KO_{2,1} = k_3 \lll 5$. Obviously , the subkey k_3 has a linear relationship with the plaintext P. □

This interesting phenomenon motivates us to explore an attack on the KASUMI by separating k_3 in the process of searching the total key.

We construct some special plaintexts P and corresponding internal state S to recovery the secret key. Let $\overline{K} = \{k_1, k_2, k_4, k_6, k_7\}$, and $L_0 = L_2 = 0x0000ffff$. For each $\overline{K}$, traverse k_3, k_5, k_8, and computate the plaintext P backward and the internal state S forward.

Property 2. Given the subkeys k_1, k_2, k_4, k_6, k_7, traverse k_3, k_5, k_8. There are at most 2^{32} plaintexts and 2^{32} internal states S from the above construction.

The proof of this property is straightforward.

Proof. The plaintex $P = P_l||P_r$, where $P_l = 0x0000ffff$, $P_r = B_0 \oplus \overline{L}_1 \oplus (KO_{2,1}||0000)$. Because P_l is a constant, thus there are 2^{32} plaintexts at the most.

Since the state S does not involve the subkey k_3, the number of states corresponding to 2^{48} (k_3, k_5, k_8) is less than 2^{32} for given $\overline{K}$. □

By property 2, it is easy to know that for fixed $\overline{K}$, there are at most 2^{32} plaintexs P and 2^{32} internal states S for 2^{48} (k_3, k_5, k_8) corresponding to L_0=L_2=$0x0000ffff$. This property is applied to reduce the complexity, which is described in the following Step 2.

With these two properties, we explore a meet-in-the-middle attack on KASUMI. In the meet-in-the-middle framework, the collison property is used to avoid the repeated computation. Besides, we use partial match to filter the wrong key as early as possible. The attack is demonstrated as follows.

Step 1. Collecting the plaintext and ciphertext pairs

For the 2^{32} plaintexts $P = 0x0000ffff||P_r$, query their corresponding ciphertexts C, and store the pairs (P_r, C) in table T_3. Therefore the data complexity is 2^{32} choosen plaintexts, for the left part of the plaintext is always $0x0000ffff$. Sort the table T_3 with the right 16 bits as the index.

For $i = 0, 1, ..., 0xffff$, construct a table of the inverse of $\overline{FI}$, i.e., table T_0, where $T_0[\overline{FI}(i)] = i$, which can be used in the Step 3.

Step 2. Computing the plaintexts P and corresponding internal states $S = (L_2, E, F, \overline{L}_3)$

For each fixed $\overline{K}$, traverse k_5 and k_8 to calculate the plaintexts P and corresponding internal states $S = (L_2, E, F, \overline{L}_3)$ by the following steps. See Fig.2.

1.1 For each k_8, calculate the output of 3-rd round function B_2 with input $L_2 = 0x0000ffff$.

1.2 For each k_8 and k_5, compute the output of 1-st round function B_0 and the input of 2-nd round function $\overline{L}_1$. There are 2^{32} values $\overline{P}_r^{i,j} = B_0^{i,j} \oplus \overline{L}_1^{i,j}$ corresponding to k_5^i, and k_8^j. Store $\overline{P}_r^{i,j}$, k_5^i, k_8^j in table T_1, and sort T_1 by the right 16 bits of $\overline{P}_r^{i,j}$.

On average there are 2^{16} values $(\overline{P}_r^{i,j}, k_5^i, k_8^j)$ with the same right 16 bits of $\overline{P}_r^{i,j}$. Compute $\overline{L}_3^{i,j} = B_2^j \oplus \overline{L}_1^{i,j}$, $E^{i,j} = FI(\overline{L}_{3,r}^{i,j}, KI_{4,2}) \oplus KO_{4,2}$, $F^{i,j} = \overline{L}_{3,l}^{i,j} \oplus KO_{4,1}^i$, store $E^{i,j}$, $F^{i,j}$,$\overline{L}_3^{i,j}$, k_5^i, k_8^j in table T_2, and sort T_2 by the index $E^{i,j}$. On average there are 2^{16} values $(F^{i,j}, k_5^i, k_8^j)$ with the same $E^{i,j}$. By this construction, for a given plaintext $P_r = P_{r,l}||P_{r,r}$, the subkey k_3 can be computed as:

$$k_3 = (P_{r,l} \oplus \overline{P}_{r,l}^{i,j}) \ggg 5,$$

where $\overline{P}_{r,r}^{i,j} = P_{r,r}$, and '$x \ggg i$' denotes that x rotate right by i bits.

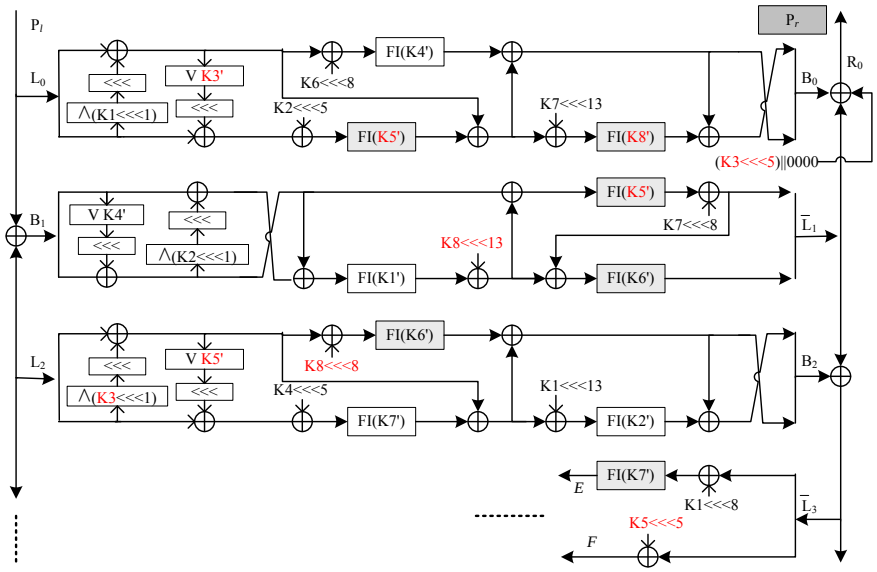

Fig. 2. The Initial Structure of the Meet-in-the-Middle Attack on KASUMI

Step 3. Recovering the right key by connecting the internal state S and C

For given $\overline{K}$, there are at most 2^{32} plaintext P corresponding to 2^{32} internal states S, which are consistent with $L_0 = L_2 = 0x0000ffff$. This property can be applied to decrease the computation complexity by avoiding some repeated computations.

For each plaintext P and corresponding internal state S, traverse k_3. For each key k_3, k_5, k_8 and the corresponding C, S, perform the following steps to connect the internal state S and C to distinguish the right key. See Fig.3 and Fig.4.

3.1 Compare whether $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$ is equal to $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$.

Here, a collision property is used to reduced the time complexity. There are 2^{32} states S, 2^{16} Fs corresponding to each E on average. Let G_0 be the value after XORing the subkey $KI_{4,1}$ in the 1-st FI of the 4-th round function, i.e., $G_0 = \overline{FI}(F \oplus KO_{2,1}) \oplus (KI_{4,1} \lll 7)$. Because the subkey k_3 is independent of E, there are 2^{16} Gs at most, which means there are some collisions in G_0 for different subkey (k_3, k_5, k_8). Then for each E, traverse G_0 to avoid the computation from G_0 to G_2 for some different (k_3, k_5, k_8), where G_2 is the value before XORing the subkey k'_3 in 3-rd FI of the 4-th round function. Let G_1 be the output of the 1-st FI of the 4-th round function, i.e., $G_1 = (\overline{FI}(G_0) \lll 9) \oplus E$, $G_2 = \overline{FI}(G_1 \oplus KO_{4,3})$. That is corresponding to the computation of the step 3 of Algorithm 1.

Besides, we apply the partial match technique to filter the wrong key as early as possible. From Fig.4, we know the values of $B_{4,l}[7-8]$ and $B_{6,l}[7-8]$ is independent from the computation of S7 S-box and the 3-rd FI function of the FO function. Therefore, filter the wrong key if $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$ is not equal to $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$. $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$ and $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$ are computed by the Algorithm 1 and Algorithm 2.

Algorithm 1. Compute $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$

INPUT : Table T_2.
OUTPUT : Table T_4.

1. For each different E of T_2 do
2. For $G \leftarrow 0$ to $0xffff$ do
3. Compute $G_1 = \overline{FI}(G_0) \oplus E$, $G_2 = \overline{FI}(G_1 \oplus KO_{4,3})$, $M = ((G_1 \oplus E) \wedge KL_{4,1})) \lll 1$;
4. For each (k'_8, F, k_5, E) in T_2 do
5. Calculate $KO_{2,1} = T_0[G_0 \oplus (k'_8 \lll 7)] \oplus F$, then $k'_3 = (KO_{2,1} \ggg 5) \oplus 0x89AB$;
6. With k'_3, M, G_2, L_2, compute $B_{3,r} = M \oplus E \oplus G_1 \oplus \overline{FI}(G_2, k'_3) \lll 9$, $B_{3,l} = G_1 \oplus E \oplus ((B_{3,r} \vee FL_{4,2}) \lll 1)$, $L_4 = B_3 \oplus L_2$, $\overline{L}_4 = FL(L_4, KL_5)$, and $\tilde{L}_4 = \overline{L}_4 \oplus (KO_{5,1} \| KO_{5,2})$;
7. Compute $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$: $t_0 = S9(S9(\tilde{L}_{4,l}[7-15] \oplus (00 \| \tilde{L}_{4,l}[0-6])) \oplus KI_{5,1}[0-8])$, $t_1 = S9(S9(\tilde{L}_{4,r}[7-15] \oplus (00 \| \tilde{L}_{4,r}[0-6])) \oplus KI_{5,2}[0-8])$, $t_2 = t_0 \oplus t_1 \oplus (\overline{L}_{4,r} \oplus \overline{L}_{3,l} \oplus KO_{2,1})[0-8]$, $B_{4,l}[7-8] \oplus L_{3,l}[7-8] = t_2[7-8]$;
8. Store $t_2, \tilde{L}_4$ index by k'_3, k_5, k'_8 in table T_4.
9. End For
10. End For
11. End For

Compare whether $B_{4,l}[7-8] \oplus L_{3,l}[7-8]$ is equal to $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$ with the same index k'_3, k_5, k'_8 in table T_4 and T_5 respectively. If they are not equal, delete the values from the two tables. There are $2^{48} \times 2^{-2} = 2^{46}$ values which are equal. Goto next step.

3.2 With the values in tables T_4 and T_5, compute $B_{4,l}[0-6]$ forward and $B_{6,l}[0-6]$ backward respectively, and compare whether $B_{4,l}[0-6] \oplus L_{3,l}[0-6]$ is equal to the

Algorithm 2. Compute $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$

INPUT : Table T_1, T_3 and the right part of plaintext P_r.
OUTPUT : Table T_5.
1. For $P_r \leftarrow 0$ to $0xffffffff$ do
2. Search C by lookup table T_3.
3. Then $H_0 = FI(L_{7,l} \oplus KO_{8,1}), KI_{8,1}) \oplus L_{7,r}$ and $H_1 = FI(H_0 \oplus (KO_{8,3}), KI_{8,3})$ can be obtained, where $L_7 = C_r$.
4. For each $k_5, k'_8, \overline{P}_r$ in T_2, where the right 16 bits of $\overline{P}_r$ has the same value with that of P_r do
5. Compute $KO_{2,1} = P_{r,l} \oplus \overline{P}^{i,j}_{r,l}$, then $k'_3 = (KO_{2,1} \ggg 5) \oplus 0x89AB$, hence $H_2 = FI(L_{7,r} \oplus k'_3), KI_{8,2}), L_6 = FL(H_1 \| H_2, KL_8) \oplus C_l$, $\overline{L}_6 = FL(L_6, KL_7)$, and $\tilde{L}_6 = \overline{L}_6 \oplus (KO_{7,1} \| KO_{7,2})$;
6. Calculate $B_{6,l}[7-8] \oplus L_{7,l}[7-8]$:
$t_3 = S9(S9(\tilde{L}_{6,l}[7-15] \oplus (00 \| \tilde{L}_{6,l}[0-6])) \oplus KI_{7,1}[0-8])$,
$t_4 = S9(S9(\tilde{L}_{6,r}[7-15] \oplus (00 \| \tilde{L}_{6,r}[0-6])) \oplus KI_{7,2}[0-8])$,
$t_5 = t_3 \oplus t_4 \oplus (\overline{L}_{6,r} \oplus L_{7,l})[0-8]$,
$B_{6,l}[7-8] \oplus L_{7,l}[7-8] = t_5[7-8]$;
7. Store $t_5, \tilde{L}_6$ index by k'_3, k_5, k'_8 in table T_5.
8. End For
9. End For

corresponding $B_{6,l}[0-6] \oplus L_{7,l}[0-6]$. If not, delete the values from the two tables. There are $2^{46} \times 2^{-7} = 2^{39}$ values left, and goto next step.

3.3 Calculate $B_{4,l}[9-15]$ forward and $B_{4,l}[9-15]$ backward according to the values in tables T_4 and T_5, and compare whether $B_{4,l}[9-15] \oplus L_{3,l}[9-15]$ is equal to $B_{6,l}[9-15] \oplus L_{7,l}[9-15]$ with the same index k'_3, k_5, k'_8. Delete the values from the two tables if they are not equal. There are $2^{39} \times 2^{-7} = 2^{32}$ values left in the two tables, and goto next step.

3.4 By the remaining values in tables T_4 and T_5, compute $B_{4,r}$ forward and $B_{6,r}$ backward, and check if $B_{4,r} \oplus L_{3,r}$ is equal to the corresponding $B_{6,r} \oplus L_{7,r}$. If they are not equal, delete the values from the two tables. There are $2^{32} \times 2^{-16} = 2^{16}$ values which are equal. Goto next step.

3.5 Calculate B_5 from L_5 by the 6-th round according to the values in tables T_4 and T_5, and check whether $L_4 \oplus L_6$ is equal to L_5. If they are not equal, delete the values from the two tables. There are $2^{16} \times 2^{-32} = 2^{-16}$ values which are equal. If they are no values left, choose another $\overline{K}$, and goto Step 1.

3.6 Test the key by trial encryption. Execute the cipher with the key for a given plaintext, and compare the result with the given ciphertext. If there is equality, return the full key.

Complexity Computation.

At step 2, there are 2^{96} one round computations and 2^{112} 2 rounds computations, which equal to 2^{110} encryptions. The table T_1 is 2^{36} bytes memory.

For step 3.1, let us consider the complexity of Algorithm 1 and algorithm 2 respectively. There are 2^{16} different E at most and 2^{16} Gs, so the complexity of step 3 in Algorithm 1 is $2^{80} \times 2^{32} \times 1/16 = 2^{108}$ encryptions. There are about 2 FI functions and 1.5 FL

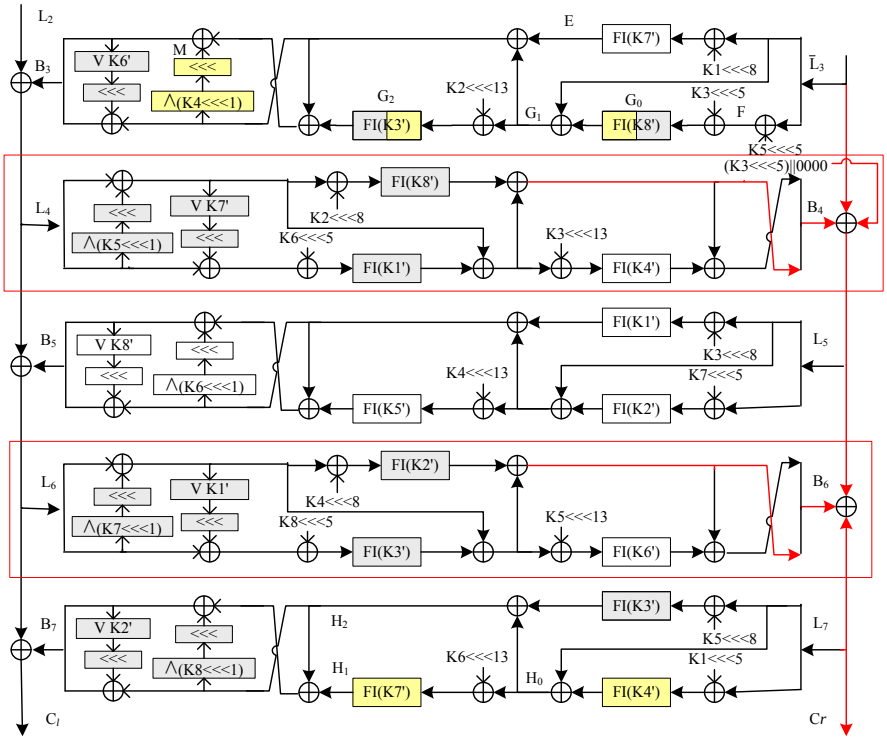

Fig. 3. The Match of the Meet-in-the-Middle Attack on KASUMI

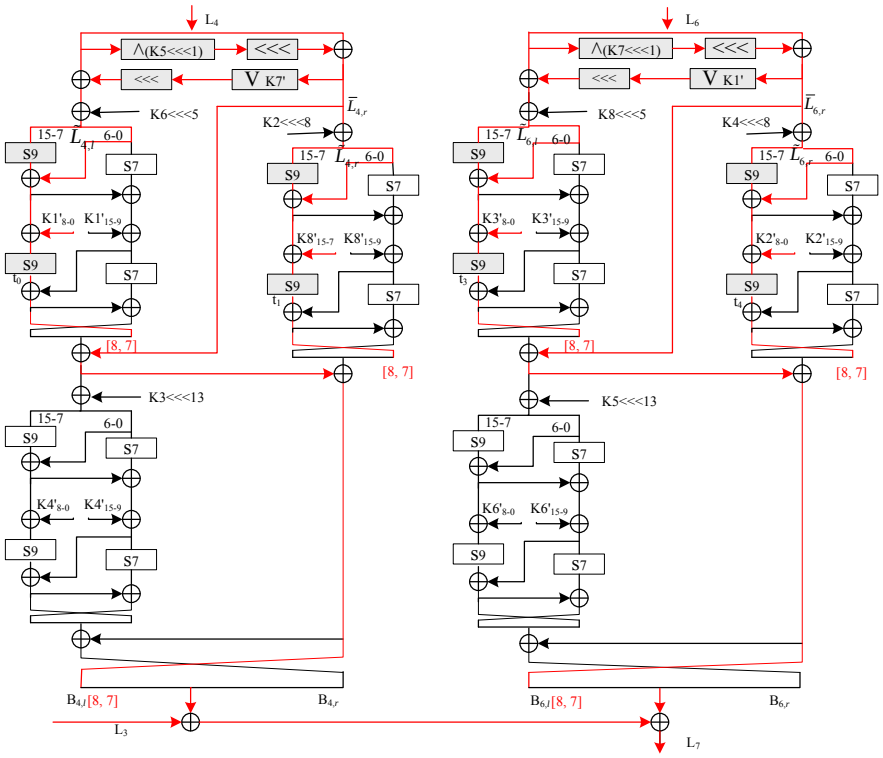

Fig. 4. The Match Detail of 4-th Round and 6-th Round in the Meet-in-the-Middle Attack on KASUMI

functions computations in the third loop, which is equivalent to $2^{80} \times 2^{48} \times 1/10 = 2^{124.7}$ encryptions. For the Algorithm 2, the computation complexity of H is $2^{80} \times 2^{32} \times 1/8 = 2^{109}$. There are about 2 FI and 2 FL functions form step 5 to step 7 in Algorithm 2, hence the computation complexity is about $2^{80} \times 2^{48} \times 1/9 = 2^{124.9}$ encryptions. As a result, the complexity of step 3.1 is $2^{125.8}$ encryptions.

The time complexity of step 3.2 is about $2^{128} \times 2^{-2} \times 2^{-6} = 2^{120}$ encryptions. Step 3.3 needs about $2^{128} \times 2^{-9} \times 2^{-6} = 2^{113}$ encryptions. There is about $2^{128} \times 2^{-16} \times 2^{-6} = 2^{106}$ encryptions in step 3.4. Step 3.5 costs about $2^{128} \times 2^{-32} \times 2^{-3} = 2^{93}$ encryptions. The compuattion complexity of step 3.6 is about $2^{128} \times 2^{-64} = 2^{64}$ encryptions. Therefore, it is about $2^{125.8}$ encryptions to recover the right key in total.

4 The Improved Meet-in-the-Middle Attack on KASUMI

In the meet-in-the-middle attack on the block cipher KASUMI, we can use precomputation and table-lookups to make the attack faster. Because there are three rotation constants $1, 5, 8$, which are involved in the computation of Step 3.1 in the match process for KASUMI. The rotation costs 3 operations, which can be replaced by 1 table-lookup operation. So construct 3 tables T_6, T_7, T_8, $T_6[i] = i \lll 1$, $T_7[i] = i \lll 5$, $T_8[i] = i \lll 18$, $i = 0, ..., 0xffff$. To get higher implementation performance of FI, construct another two tables $T_9[i] = \overline{FI}(i) \lll 7$, $i = 0, 1, ..., 0xffff$, and $T_{10}[i] = \overline{FI}(i \ggg 7) \lll 9$, $i = 0, 1, ..., 0xffff$. Therefore $FI(x, KI_{i,j}) = T_{10}[T_9[x] \oplus KI_{i,j}]$, which needs two table-lookups and 1 exclusive-or operation. In order to speedup the comparison of step 2.2, build table $T_6[i||j] = S_9(S_9(i \gg 7) \oplus (i \wedge 0x7f) \oplus j) \gg 7$, where $i = 0, 1, ..., 0x1ff$, $j = 0, 1, ..., 0x1ff$. So, some computations of the round function can be replaced by some table-lookups. Sum up, the attack needs $2^{128} \times 3.5$ FL functions computations and $2^{128} \times 10$ table-lookups. The complexity is equivalent to 2^{125} encryptions.

5 Conclusion

In this paper, we give a middle-in-the-middle attack on KASUMI, which is the first attack on the full KASUMI with single key. We notice that the key impact on the block cipher is nonrandom in 3-round KASUMI. There is a linear equation in the plaintexts P and the subkey k_3, and the subkey k_3 is not involved in the internal state S. P and S are computed backward and forward by the values of the inputs of the 1-st round function and 3-rd round function, where the input values are 0x0000ffff. Take advantage of this property, we use some collision properties to decrease the time complexity. This attack needs 2^{32} choose plaintexts with a computation complexity of $2^{125.8}$ encryptions and 2^{48} block memory. Besides, some optimal implementation is shown to give a faster attack by table-lookup techniques. The computation complexity can be decrease to 2^{125} encryptions. Our attack can also be extended to attack the block cipher MISTY1.

Acknowledgments.

We would like to thank Jiazhe Chen for the helpful comments on this paper.

References

1. 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)
2. Aoki, K., Sasaki, Y.:Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi(ed.) Advances in Cryptology-CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
3. 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)
4. 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)
5. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. available at http://eprint.iacr.org/2011/449.pdf, 2011.
6. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In SAC 2010, LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2010)
7. Dunkelman, O., Keller, N.:An Improved Impossible Differential Attack on MISTY1. In: Pieprzyk, J. (Ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 441–454. Springer, Heidelberg (2008)
8. 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)
9. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Springer, Heidelberg (2010)
10. Diffie, W., Hellman, M. E.: Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10 (6): 74–84.
11. Demirci, H., Selçuk. A.: A Meet-in-the-Middle Attack on 8-round AES. In FSE 2008. LNCS, vol. 5086, pp. 116–126. Springer, Heidelberg (2008)
12. Dunkelman, O., Sekar, G., Preneel, B.: Improved Meet-in-the-Middle Attacks on Reduced Round DES. In INDOCRYPT 2007. LNCS, vol. 4859, pp. 86–100. Springer, Heidelberg (2007)
13. Demirci, H., Taskin, I., Coban, M., Baysal, A.: Improved Meet-in-the-Middle Attacks on AES. In INDOCRYPT 2009. LNCS, vol. 5922, pp.144–156. Springer, Heidelberg (2009)
14. 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)
15. 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)
16. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. available at http://eprint.iacr.org/2011/286.pdf, 2011.
17. Kühn, U.: Cryptanalysis of Reduced-Round MISTY. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 325–339. Springer, Heidelberg (2001)
18. Matsui, M.: Block encryption algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 64–74. Springer, Heidelberg (1997)
19. 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)

20. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 4: Design and evaluation report, V6.1.0 (2002)
21. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Cramer, R. (ed.) Advances in Cryptology-EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
22. Tsunoo1,Y., Saito, T., Shigeri,M., Kawabata, T.:Higher Order Differential Attacks on Reduced-Round MISTY1, In Lee, P.J., Cheon, J.H.(Eds.) ICISC 2008. LNCS, vol. 5461, pp. 415–431. Springer, Heidelberg (2009)
23. Wei, L., Rechberger, C., Guo, J., Wu, H., Wang, H., Ling, S.: Improved Meet-in-the-Middle Cryptanalysis of KTANTAN. Cryptology ePrint Archive, Report 2011/201, 2011. http://eprint.iacr.org/.