Practical-Time Attacks Against Reduced Variants of MISTY1 ⋆

Orr Dunkelman[1,3] and Nathan Keller[2,3]

[1] Computer Science Department
University of Haifa
Haifa 31905, Israel
`orrd@cs.haifa.ac.il`

[2] Department of Mathematics
Bar Ilan University
Ramat Gan, 52900, Israel
`nathan.keller@math.biu.ac.il`

[3] Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26, Rehovot 76100, Israel

Abstract. MISTY1 is a block cipher designed by Matsui in 1997. It is widely deployed in Japan where it is an e-government standard, and is recognized internationally as a NESSIE-recommended cipher as well as an ISO standard and an RFC. Moreover, MISTY1 was selected to be the blueprint on top of which KASUMI, the GSM/3G block cipher, was based. Since its introduction, and especially in recent years, MISTY1 was subjected to extensive cryptanalytic efforts, which resulted in numerous attacks on its reduced variants. Most of these attacks aimed at maximizing the number of attacked rounds, and as a result, their complexities are highly impractical.

In this paper we pursue another direction, by focusing on attacks with a *practical* time complexity. The best previously-known attacks with practical complexity against MISTY1 could break either 4 rounds (out of 8), or 5 rounds in a modified variant in which some of the FL functions are removed. We present an attack on 5-round MISTY1 with all the FL functions present whose time complexity is 2^{38} encryptions. When the FL functions are removed, we present a devastating (and experimentally verified) related-key attack on the full 8-round variant, requiring only 2^{18} data and time.

While our attacks clearly do not compromise the security of the full MISTY1, they expose several weaknesses in MISTY1's components, and improve our understanding of its security. Moreover, future designs which rely on MISTY1 as their base, should take these issues into close consideration.

⋆ This paper is based on the paper "*An Improved Impossible Differential Attack on MISTY1*" [8] presented at ASIACRYPT 2008. The results presented in Sections 4 and 5 are entirely new and were not published before.

1 Introduction

MISTY1 is a 64-bit block cipher with 128-bit keys designed in 1997 by Matsui [20]. In 2002, MISTY1 was selected by the Japanese government to be one of the CRYPTREC e-government ciphers, and since then, it became widely deployed in Japan. MISTY1 also gained recognition outside Japan, when it was selected to the portfolio of the NESSIE-recommended ciphers [22], and approved as an RFC in 2000 [21] and as an ISO standard in 2005 [12]. Furthermore, the block cipher KASUMI [27] designed as a slight modification of MISTY1 is used in the GSM/3G networks, which makes it one of the most widely used block ciphers today. This makes examination of the security of MISTY1 and its variants one of the central practical questions in block cipher cryptanalysis.

MISTY1 has an 8-round recursive Feistel structure, where the round function FO is in itself a 3-round Feistel construction, whose F-function FI is in turn a 3-round Feistel construction using 7-bit and 9-bit invertible S-boxes. The specific choice of S-boxes and the recursive structure suggest security against differential and linear cryptanalysis. In addition, to further thwart attacks, after every two rounds an FL function is applied to each of the two halves independently. The FL functions are key-dependent linear functions which play the role of whitening layers (even in the middle of the encryption).

Since its introduction, and especially in recent years, MISTY1 was subjected to extensive cryptanalytic efforts, which resulted in numerous attacks on its reduced variants. The best currently known attacks (in terms of number of rounds) are an impossible differential attack on 6-round MISTY1 [13] requiring $2^{112.4}$ encryptions, an impossible differential attack on a 7-round variant with some of the FL functions removed [13] requiring $2^{124.8}$ encryptions, and a meet-in-the-middle attack which allows to speed up exhaustive key search on the full MISTY1 by a small factor [14].

Most of the previous attacks on MISTY1 aimed at maximizing the number of attacked rounds, and as a result, their complexities were highly impractical. In this paper, we pursue a different research direction, by focusing on attacks with a *practical* time complexity. Only several such attacks were presented before, the best of those on 4-round MISTY1 requiring 2^{45} encryptions [18] and on 5-round MISTY1 with the last two FL functions removed, requiring $2^{27.32}$ encryptions [25].

In this paper we present three practical-time attacks on reduced variants of MISTY1. The first two attacks target 5-round MISTY1 with all FL functions present, and the faster among them requires only 2^{38} encryptions. The third attack is a related-key attack on 8-round MISTY1 without the FL functions, requiring only 2^{18} plaintexts and time. We fully verified the related-key attack experimentally, finding the full last round subkey in a negligible time on a PC. A comparison of our attacks with previous practical-time attacks on reduced-round MISTY1 is presented in Table 1.

While our three attacks use very different techniques (impossible differentials [3], Square attack [7], and related-key attack [2]), their high efficiency results from exploiting the same weakness, which appears in all MISTY1 components.

Attack	Rounds	FL functions	Complexity Data	Complexity Time
Slicing Attack [18]	4	All	$2^{22.25}$ CP	2^{45}
Higher-Order Differential [1]	5	None	$2^{10.5}$ CP	2^{17}
Integral [16]	5	Most	2^{34} CP	2^{48}
Integral [25]	5	Most	2^{34} CP	$2^{27.32}$
Impossible Differential (Section 3)	5	All	$2^{38.6}$ CP	2^{46}
Square (Section 4)	5	All	$2^{35.6}$ CP	2^{38}
Related-Key Slide (Section 5)	8	None	2^{18} CP	2^{18}

CP – Chosen plaintext

Table 1. Summary of Practical-Time Attacks on Reduced Variants of MISTY1

This weakness is the ability to *divide* the states into several sub-states whose influence on each other is limited. Hence, despite the recursive structure, and the large number of S-boxes (9 S-boxes) in each round, the adversary is still able to divide the 32-bit state of the FO function to four chunks of 7,9,7, and 9 bits, and divide the 32-bit state of the FL function to even smaller chunks of 2 bits each. This division allows significantly speeding up cryptanalytic attacks which now have to deal with a smaller state (and thus, less keying material). We note that variants of both divisions were used in previous works: A division of FO to two 16-bit chunks in order to attack 5-round MISTY1 without the last two FL-functions [25] and a division of FL to 2-bit chunks in order to attack 4-round MISTY1 [18].

Our attacks highlight several weaknesses in the design of MISTY1 components, which lead to the "division" properties described above. The first is the fact that both the FO function and the FI function have only 3 rounds, which allows for dividing the state of FO into four chunks. The second is the bit-sliced nature of the FL function, which allows to divide its state to sixteen 2-bit chunks. Interestingly, both weaknesses were partially addressed in the design of KASUMI: a fourth round was added to the FI function (but not to the FO function), and a rotation by a single bit was added into the FL function. However, these changes address the weaknesses only partially, as a variant of our second attack applies to a 5-round version of KASUMI (though with a higher complexity of 2^{64}). Therefore, while our attacks clearly do not compromise the security of the full MISTY1, they point out weaknesses in its components that should be avoided in design of future MISTY1-based ciphers.

This paper is organized as follows: In Section 2 we give a brief description of the structure of MISTY1. In Section 3 we present an impossible differential attack on 5-round MISTY1 based on dividing the FL function state. In Section 4 we present a Square attack on 5-round MISTY1 based on dividing both the FO function and the FL function states simultaneously. In Section 5 we present a highly efficient related-key attack on 8-round MISTY1 without FL functions, and we conclude with a summary and discussion in Section 6.

$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KO_{i,4}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$	$KL_{i,1}$	$KL_{i,2}$
K_i	K_{i+2}	K_{i+7}	K_{i+4}	K'_{i+5}	K'_{i+1}	K'_{i+3}	$K_{\frac{i+1}{2}}$ (odd i) $K'_{\frac{i}{2}+2}$ (even i)	$K'_{\frac{i+1}{2}+6}$ (odd i) $K_{\frac{i}{2}+4}$ (even i)

Table 2. The Key Schedule Algorithm of MISTY1

2 Brief Description of MISTY1

MISTY1 is an 8-round Feistel construction, where the round function, FO, is in itself a variant of a 3-round Feistel construction, defined as follows. The input to FO is divided into two halves. The left one is XORed with a subkey, enters a keyed permutation FI, and the output is XORed with the right half. After the XOR, the two halves are swapped, and the same process (including the swap) is repeated two more times. After that, an additional swap and an XOR of the left half with a subkey is performed (see Figure 1).

The FI function in itself also has a Feistel-like structure. Its 16-bit input is divided into two unequal parts — one of 9 bits, and the second of 7 bits. The left part (which contains 9 bits) enters an S-box, $S9$, and the output is XORed with the right 7-bit part (after padding the 7-bit value with two zeroes as the most significant bits). The two parts are swapped, the 7-bit part enters a different S-box, $S7$, and the output is XORed with 7 bits out of the 9 of the right part. The two parts are then XORed with a subkey, and swapped again. The 9-bit value again enters $S9$, and the output is XORed with the 7-bit part (after padding). The two parts are then swapped for the last time.

Every two rounds, starting before the first one, each of the two 32-bit halves enters an FL layer. The FL layer is a simple linear transformation. Its input is divided into two halves of 16 bits each, the AND of the left half with a subkey is XORed to the right half, and the OR of the updated right half with another subkey is XORed to the left half. We outline the structure of MISTY1 and its parts in Figure 1.

The key schedule of MISTY1 takes the 128-bit key, and treats it as eight 16-bit words $K_1, K_2, \ldots, K_8$. From this sequence of words, another sequence of eight 16-bit words is generated, according to the rule $K'_i = FI_{K_{i+1}}(K_i)$.[1]

In each round, seven words are used as the round subkey, and each of the FL functions accepts two subkey words. We give the exact key schedule of MISTY1 in Table 2.

In [17], Kühn observed that the FO function has an equivalent description which uses only 107 equivalent subkey bits (rather than 112). We present this description and use it in Section 4.

[1] In case the index of the key j is greater than 8, the used key word is $j - 8$.

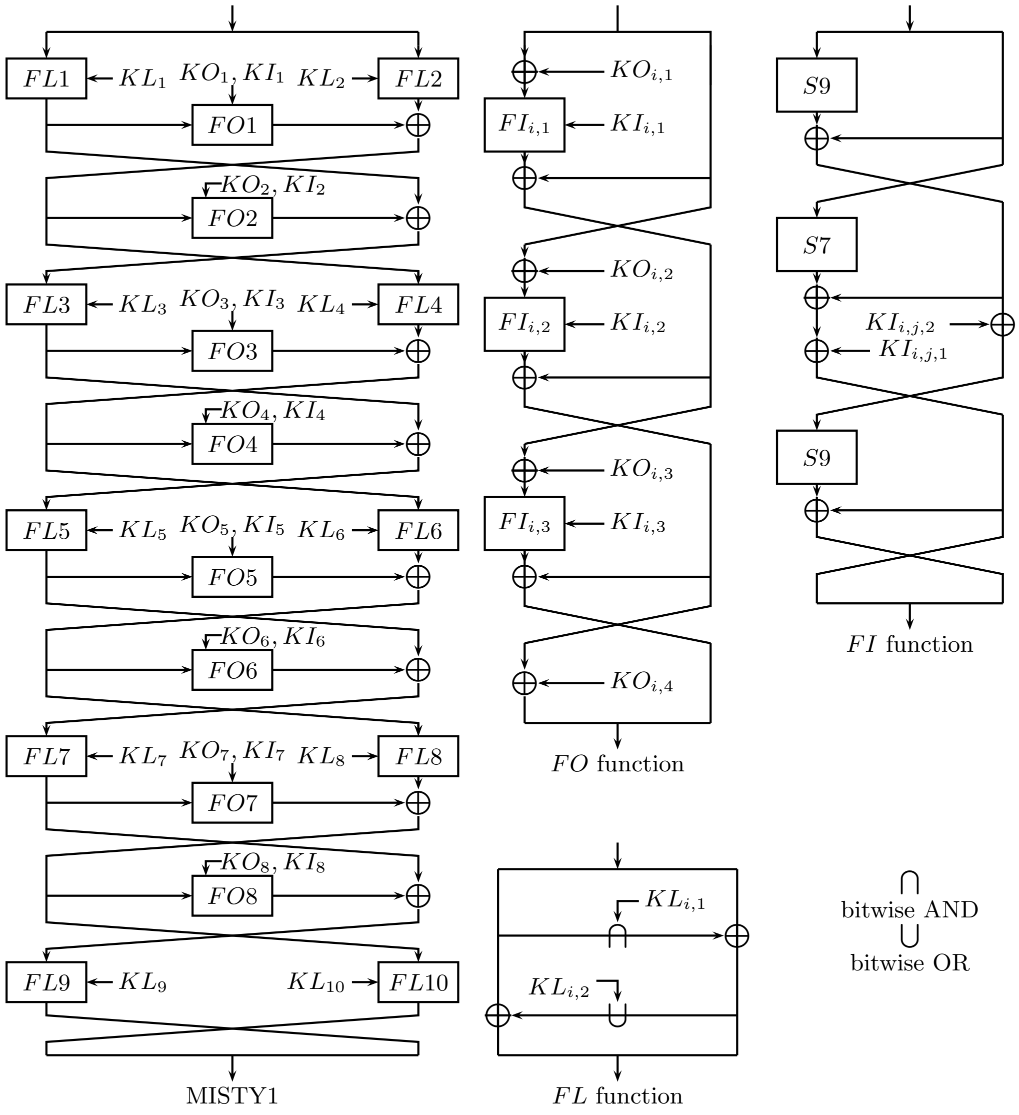


Fig. 1. Outline of MISTY1

3 Impossible Differential Attack on 5-Round MISTY1 – Dividing the *FL* Function

Due to its general structure, 5-round MISTY1 without the FL functions is susceptible to the generic impossible differential attack against 5-round Feistel constructions with a bijective round function [3, 15]. In [18] it was claimed that the existence of the FL functions makes such an attack impossible against 5-round MISTY1 with all the FL functions present. In this section we show that rather than destroying the attack, the FL functions allow to make it much more effective, due to the possibility to attack the FL functions themselves, rather than the more complex round functions. We end up with attacking a cascade of four sequential applications of FL, which can be performed very efficiently since FL can be treated as sixteen 2-bit functions applied in parallel. Using this division, we obtain an attack which requires 2^{38} chosen plaintexts and 2^{46} memory accesses and recovers slightly more than 41 bits of information on the 96 key bits used in the FL functions.[2] Although the attack complexity is higher than that of the Square attack presented in the next section, we chose to present this attack, since it demonstrates and exploits a weakness of the MISTY1 design which is not exploited in the Square attack. One may thus hope to find a more powerful combined attack which will make use of both weaknesses simultaneously.

We note that the idea of attacking the FL functions and dividing them was first presented by Kühn [18], and used to attack 4-round MISTY1. We extend the idea and combine it with the generic 5-round impossible differential, which allows us to attack 5-round MISTY1 with roughly the same time complexity as [18].

The remainder of this section is organized as follows: In Section 3.1 we present the 5-round impossible differential used in the attack. We then show the main approach in using this impossible differential in Section 3.2. The resulting attack which recovers about 41 bits of the key is described in Section 3.3, and we discuss how to recover the remainder of the key in Section 3.4.

3.1 The 5-Round Impossible Differential Used in the Attack

The generic attack on 5-round Feistel constructions is based on the following impossible differential:

Observation 1 ([3], page 136) *Let $E : \{0,1\}^{2n} \to \{0,1\}^{2n}$ be a 5-round Feistel construction with a bijective round function $f : \{0,1\}^n \to \{0,1\}^n$. Then for all $\alpha \in \{0,1\}^n$, the differential $(0,\alpha) \to (0,\alpha)$ through E is impossible.*

We observe that a similar impossible differential exists even if FL layers are added to the construction, as in MISTY1. Note that since for a given key, the FL layers are linear, we can define $FL(\alpha)$ for a difference α as the unique difference β such that $(x \oplus y = \alpha) \Rightarrow (FL(x) \oplus FL(y) = \beta)$.

Proposition 1. *Let E denote a 5-round variant of MISTY1, with all the FL functions present (including an FL layer after round 5). If for the given secret key we have $FL7(FL6(FL4(FL2(\alpha)))) = \beta$, then the differential $(0,\alpha) \to (0,\beta)$ through E is impossible.*

[2] Throughout this paper, we discuss attacks which reveal a significant portion of the secret key. Once this part is revealed we consider the scheme to be broken due to the large reduction in the security of the cipher. For all attacks we briefly discuss the issue of retrieving the full key (after the attacks' description).

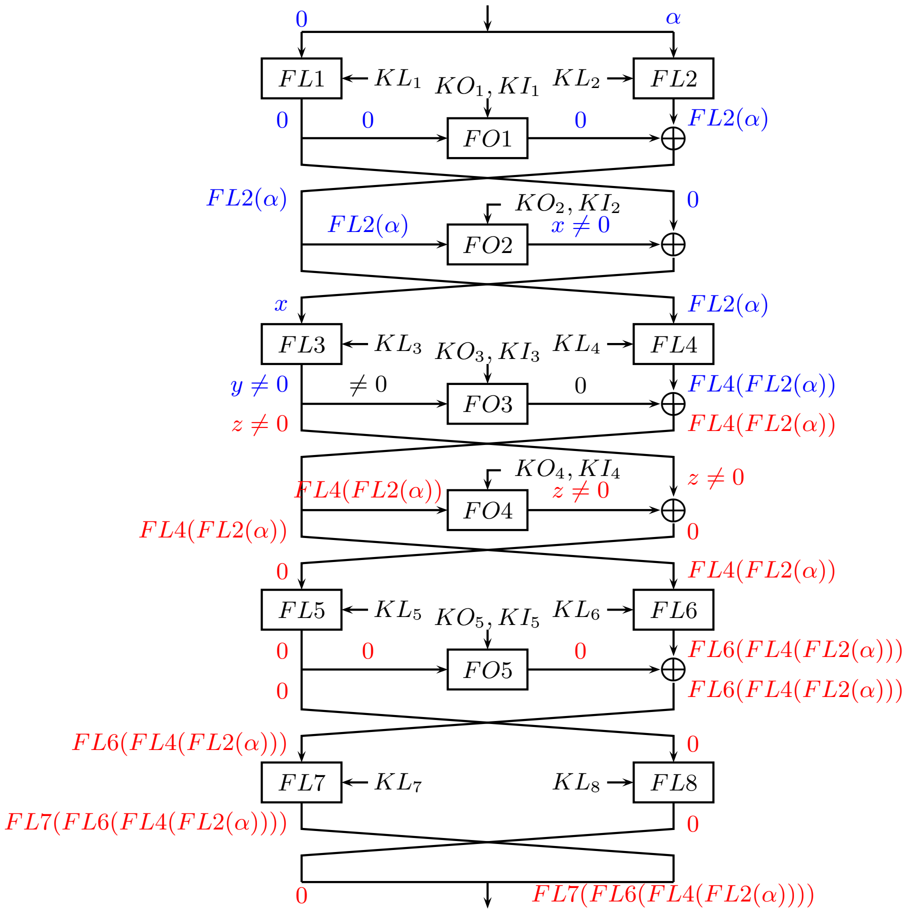


Fig. 2. A 5-Round Impossible Differential of MISTY1

Proof. The impossible transition is demonstrated in Figure 2. If the plaintext difference is $(0, \alpha)$, then after the first FL layer, the difference becomes $(0, FL2(\alpha))$. This difference evolves after two rounds (including the second FL layer) to $(y, FL4(FL2(\alpha)))$, where $y \neq 0$ due to the bijectiveness of the round function of MISTY1.

On the other hand, if the output difference is $(0, \beta)$ such that $\beta = FL7(FL6(FL4(FL2(\alpha))))$, then before the last FL layer, the difference is $(FL6(FL4(FL2(\alpha))), 0)$, and thus the input difference to round 5 (before the swap) is $(0, FL6(FL4(FL2(\alpha))))$. Thus, the difference before the third FL layer is $(0, FL4(FL2(\alpha)))$.

However, if the input difference to round 3 is $(z \neq 0, FL4(FL2(\alpha)))$ and the output difference of round 4 (before the FL layer, including the swap) is $(0, FL4(FL2(\alpha)))$, then the output difference of the FO function in round 3 must be zero. This is impossible since the input difference to this FO function is $y \neq 0$, and the FO function is bijective.

Hence, the differential $(0, \alpha) \rightarrow (0, \beta)$ is indeed impossible.

We note that a similar approach is used in [18], using the generic 3-round impossible differential $(0, \alpha) \rightarrow (0, \beta)$ for all non-zero α, β which holds for every 3-round Feistel construction with a bijective round function.

3.2 Dividing the *FL* Functions

A straightforward way to use the impossible differential described above to attack 5-round MISTY1 is to encrypt many pairs with difference $(0, \alpha)$ for non-zero α's, consider the pairs whose ciphertext difference is of the form $(0, \beta)$, and discard subkeys of the FL layers for which $FL7(FL6(FL4(FL2(\alpha)))) = \beta$. However, since the subkeys used in $FL2, FL4, FL6$, and $FL7$ are determined by 96 subkey bits, this approach is very time consuming.[3] Instead, we show that the division of the FL function to sixteen 2-bit functions applied in parallel, allows to detect efficiently all the instances for which $FL7(FL6(FL4(FL2(\alpha)))) = \beta$, for any given pair (α, β).

We use a series of observations, most of which were first presented in [18]. In the followings, the function $FL7 \circ FL6 \circ FL4 \circ FL2$ is denoted by G.

1. For each $0 \leq i \leq 15$, the i-th bits of both halves of the output of an FL function depend only on the i-th bits of both halves of the input and the i-th bits of both halves of the corresponding subkey KL_i. As a result, each FL function can be represented as a parallel application of 16 functions $f_i : \{0,1\}^2 \rightarrow \{0,1\}^2$ keyed by two different subkey bits each.
2. Each such f_i is linear for a fixed key and invertible.
3. The two observations above hold also for a series of FL functions applied sequentially. In particular, the function $G = FL7 \circ FL6 \circ FL4 \circ FL2$ can be represented as a parallel application of 16 functions $g_i : \{0,1\}^2 \rightarrow \{0,1\}^2$ keyed by eight subkey bits each. Each g_i is linear and invertible, and hence, can realize only one of six possible functions.[4] Thus, there are only $6^{16} = 2^{41.36}$ possible G functions.
4. Since each g_i is invertible, the differentials $0 \rightarrow a$ and $a \rightarrow 0$ through g_i are impossible, for each non-zero $a \in \{0,1\}^2$. As a result, most of the differentials of the form $\alpha \rightarrow \beta$ through G are impossible, regardless of the subkeys used in the FL functions. In each of the g_i-s, only 10 out of the 16 possible input/output pairs are possible. Hence, only $(10/16)^{16} = 2^{-10.85}$ of the input/output pairs for G are possible.

[3] A reader interested in a variant of MISTY1 with no swap after the fifth round will need to replace $FL7$ with $FL8$ in all places. The effect on the attack is just in the used key words, but this has no effect on the data, nor the time complexities of the attack.

[4] Since we are interested only in differences, we treat two functions that differ by an additive constant as the same function. The total number of functions for each g_i is actually 24.

5. The impossible pairs can be detected efficiently by checking a simple Boolean expression. Formally, let the input difference to G be (x_1, x_2) and the output difference be (y_1, y_2). Also, let $\overline{t}$ denote the bitwise not of t, let & denote bitwise AND, and | denote bitwise OR. Then the transition is impossible for any key if

$$\overline{x_1}\&\overline{x_2}\&(y_1|y_2) \neq 0 \qquad \text{or} \qquad \overline{y_1}\&\overline{y_2}\&(x_1|x_2) \neq 0,$$

since this corresponds to a differential of the form $0 \to a$ or $a \to 0$ for $a \neq 0$ in at least one of the sixteen g_i functions.
6. Let (α, β) be a pair such that the differential $\alpha \to \beta$ through G is possible and was not discarded at the previous step. We want to find how many of the possible $2^{41.36}$ G functions satisfy $G(\alpha) = \beta$. For each g_i, there are 10 possible input/output pairs (the other six pairs are impossible for any subkey). For the $0 \to 0$ pair, all the six possible g_i functions satisfy this condition. For each of the other 9 input/output pairs, two of the six functions satisfy the condition. Since the g_i functions are independent, the expected number of functions satisfying $G(\alpha) = \beta$ is:

$$\sum_{j=0}^{16} \binom{16}{j} \cdot \left(\frac{9}{10}\right)^j \cdot \left(\frac{1}{10}\right)^{16-j} \cdot 2^j \cdot 6^{16-j} = 2^{20.3}.$$

7. Finally, since each g_i function can be treated independently, one can enumerate the $2^{41.36}$ possible G functions in such a way that for each pair (α, β), the functions G satisfying $G(\alpha) = \beta$ can be found efficiently.

Using these observations on the structure of the FL functions, we are ready to present our attack.

3.3 Attack Algorithm and Analysis

The attack algorithm is the following:

1. Ask for the encryption of 64 structures of 2^{32} plaintexts each, such that in each structure, the left half of all the plaintexts is equal to some random value A, while the right half obtains all possible values. (As a result, the difference between two plaintexts in the same structure is of the form $(0, \alpha)$).
2. In each structure, find the pairs whose output difference is of the form $(0, \beta)$.
3. For each pair with input difference $(0, \alpha)$ and output difference $(0, \beta)$ check whether $\alpha \to \beta$ is an impossible differential for the function G (as described in Section 3.2). Discard pairs which fail this test.
4. For each remaining pair, find all the functions G satisfying the condition $G(\alpha) = \beta$ and discard them from the list of all possible G functions.
5. After analyzing all the remaining pairs, output the list of remaining G functions.

Step 2 of the algorithm can be easily implemented by a hash table, resulting in about 2^{31} pairs from each structure. Step 3 is performed by evaluating a simple Boolean function on the input and the output. It follows from Observation 4 in Section 3.2 that in each structure, out of the 2^{31} pairs, about $2^{20.15}$ pairs remain at this point. Each of these pairs discards about $2^{20.3}$ possible values of G on average (as shown in Observation 6 in Section 3.2), and the identification of the discarded functions can be performed very efficiently. After analyzing about 64 structures, the expected number of remaining G functions (except for the right one) is

$$2^{41.36} \cdot (1 - 2^{20.3-41.36})^{2^{20.15+6}} = 2^{-7.78}$$

which means that we are left only with the right G function. The time complexity of the attack is about $64 \cdot 2^{20.15} \cdot 2^{20.3} = 2^{46.45}$ memory accesses, and the information retrieved by the adversary is equivalent to 41.36 key bits.

3.4 Retrieving the Rest of the Secret Key

The most naive approach toward retrieval of the rest of the key is to try all possible 2^{96} subkeys which affect the functions $FL2, FL4, FL6$, and $FL7$, and check (for each subkey) whether it yields the correct function G. A more efficient way is based on the fact that this G can be treated as an 8-round Feistel, where the following keys are used as subkeys: K_2', K_3', K_4', K_5' and K_4, K_5, K_6, K_7 (the order of the keys is such that K_2' and K_4 are used last in $FL7$). This allows performing a meet in the middle attack on G. Namely, by guessing K_3, K_4, K_5, and K_6, it is possible to compute half of the output of $FL6 \circ FL4 \circ FL2$, and by guessing K_2, K_3, and K_4, we can compute the full output of $FL7^{-1}$. By taking out the common part (K_3 and K_4) to the outer loop, one can easily find all consistent keys in time 2^{64} (using 2^{32} memory). Recovering K_7 is then a trivial operation. Hence, identifying the appropriate $2^{96} \cdot 2^{-41.36} = 2^{54.64}$ 96-bit subkeys takes about 2^{64} evaluations of $FL6 \circ FL4 \circ FL2$, which are significantly faster than 2^{64} evaluations of the 5-round MISTY1.

Retrieving the rest of the key by exhaustive search leads to a total time complexity of $2^{86.64}$ encryptions. We note that this part of the attack can probably be performed much more efficiently using some different attack techniques and exploiting the key information obtained so far. For example, several such techniques applicable in special cases (e.g., if the adversary can use both chosen plaintext and chosen ciphertext queries) are presented in [18].

4 Square Attack on 5-round MISTY1 — Dividing Both *FO* and *FL* Simultaneously

The Feistel structure of MISTY1 and its bijective round function allow to mount a generic Square attack [7] on 4-round MISTY1. The idea behind the generic attack is simple. Consider a set of 2^{32} chosen plaintexts in which the left half is constant and the right half assumes each of the 2^{32} possible values exactly once.

It is easy to see that the XOR of the right halves of the 2^{32} corresponding values after 4 Feistel rounds is equal to zero (see proof below). As was observed in [16], this property holds even if the FL layers are inserted every two rounds, like in MISTY1.

In order to use this property to attack 5-round MISTY1, one has to consider several such Square structures, and examine the 5th round in order to check whether the right halves after the 4th round sum up to zero. As the round function of MISTY1 is very complex and depends on 176 subkey bits (including the last FL layer), one would expect such an analysis to be very time consuming. However, we show that the FO and FL functions can be divided simultaneously in such a way that the Square structures can be checked and the subkey can be found very efficiently. We obtain an attack on 5-round MISTY1 with all FL functions present, requiring only $2^{35.6}$ data and 2^{38} time.

This section is organized as follows. In Section 4.1 we recall an equivalent description of the FO function and give the notations used in the attack. In Section 4.2 we present the basic Square attack, in Section 4.3 we present the division of FO and FL, and in Section 4.4 we present a meet-in-the-middle approach which allows to combine the division with the Square attack efficiently. The attack algorithm is presented in Section 4.5, and analyzed in Section 4.6. In Section 4.7 we show how the adversary can retrieve the rest of the secret key efficiently, and we consider application of the attack to other variants of MISTY1 in Section 4.8.

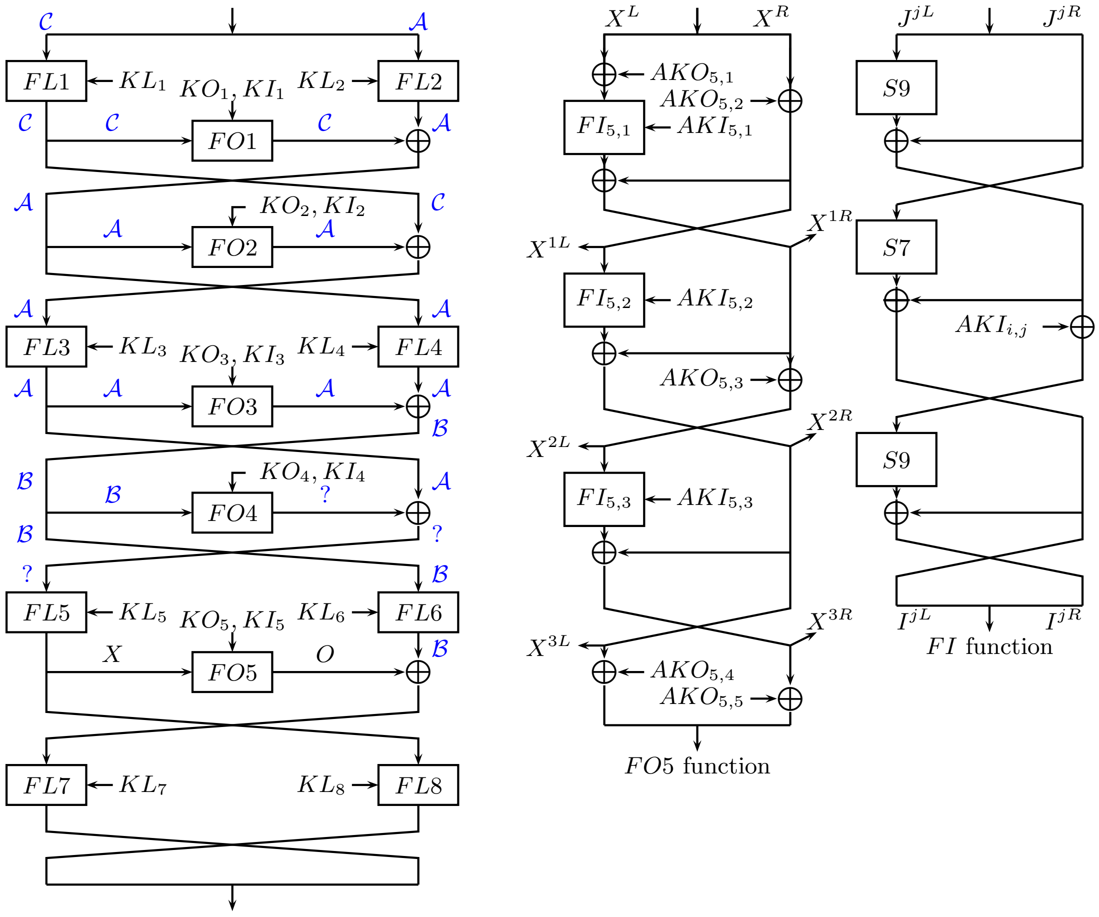
4.1 Notations

In our attack, we use an equivalent description of the FO function, presented by Kühn in [17]. Kühn observed that since the FI function has only 3 rounds, the 7-bit subkey $KI_{i,j,1}$ affects its output in a linear way, and thus it can be absorbed in the subsequent KO subkeys. This leads to an equivalent description of the FO function which uses only 107 equivalent subkey bits, instead of 112 subkey bits in the original description. This equivalent description is presented in the middle part of Figure 3. The equivalent subkeys are the following:

$$\begin{aligned}
&AKO_{i,1} = KO_{i,1}\\
&AKO_{i,2} = KO_{i,2}\\
&AKO_{i,3} = KO_{i,2} \oplus KO_{i,3} \oplus KI'_{i,1}\\
&AKO_{i,4} = KO_{i,2} \oplus KO_{i,4} \oplus KI'_{i,1} \oplus KI'_{i,2}\\
&AKO_{i,5} = KO_{i,2} \oplus KI'_{i,1} \oplus KI'_{i,2} \oplus KI'_{i,3}\\
&AKI_{i,j} = [KI_{i,j}]_{\{8,\ldots,0\}}
\end{aligned}$$

where we use $X_{\{a,a-1,a-2,\ldots,b\}}$ to denote bits a downto b of the word X, where $KI'_{i,j} = [KI_{i,j}]_{\{15,\ldots,9\}}||00||[KI_{i,j}]_{\{15,\ldots,9\}}$, $[KI_{i,j}]_{\{15,\ldots,9\}}$ are the 7 most significant bits of $KI_{i,j}$ which are XORed with the 7-bit half in the FI function, and where $[KI_{i,j}]_{\{8,\ldots,0\}}$ are the 9 least significant bits of $KI_{i,j}$.

We use the following notations, demonstrated in Figure 3, for intermediate states of the function $FO5$ and of the FI functions included in it: The input



$\mathcal{A}$ denotes an active word, $\mathcal{B}$ denotes a balanced word, $\mathcal{C}$ denotes a constant word, and ? denotes a word whose status is unknown.

Fig. 3. Notations used in our 5-Round Square Attack on MISTY1

to the function $FO5$ is denoted by X, and its left and right 16-bit halves are denoted by X^L and X^R, respectively. The two halves of the intermediate state after the j'th FI function are denoted by X^{jL} and X^{jR}. The 9-bit left part and the 7-bit right part of the input to the j'th FI function are denoted by J^{jL} and J^{jR}. The 7-bit left part and the 9-bit right part of the output of the j'th FI function are denoted by I^{jL} and I^{jR}, respectively. The output of the entire function $FO5$ is denoted by O.

For each state Z (either a 32-bit state, 16-bit state, etc.), the value of the state in the intermediate encryption of the i'th plaintext is denoted by Z_i, the XOR difference between its values in two encryptions is denoted by ΔZ, and the XOR of its values over the entire Square structure is denoted by $\sum Z$.

4.2 The Square Property

We now present in detail the Square property used in the attack: Consider a structure of 2^{32} plaintexts of the form $(\mathcal{C},\mathcal{A})$, where $\mathcal{C}$ denotes a constant word, and $\mathcal{A}$ denotes an active word.[5] In other words, the structure is composed of 2^{32} plaintexts of the form (α,β_i), where α is a constant 32-bit word, and β_i assumes all 2^{32} possible values. In the sequel, we denote such structures by "Square structures". The propagation of the values of a Square structure through rounds 1–4 of MISTY1 is shown in the left part of Figure 3.

As we can see in the figure, after the initial FL layer, the left half remains constant (though, possibly it may become a different constant), and the right half remains active (though, the order of the values may change). Since the input of the function $FO1$ is constant, at the beginning of round 2, the right half is constant and the left half is active. Given that for a fixed key the $FO2$ function is bijective, at the end of round 2, both halves are active. This situation remains also in the input of round 3, since the FL layer is also bijective.

Again, as the input $FO3$ is active, then so is its output. However, this property is not preserved by the XOR with the right half of the input to round 3 (which is active as well), but the word is still balanced. Hence, the left half of the input to round 4 is balanced, which following the Feistel structure becomes the right half after round 4. Finally, since the FL function is linear, it follows that the XOR of the 2^{32} outputs of $FL6$ is also equal to zero. We have thus proved:

Proposition 2. *[16] Consider a structure of 2^{32} plaintexts where the left 32 bits are held constant, and the right 32 bits take on all possible values. Then the XOR of the 2^{32} outputs of the function $FL6$ is zero.*

The standard way to exploit the 4-round Square property to attack 5-round MISTY1 is to consider several Square structures, guess some key material in round 5, partially decrypt the ciphertexts and check whether the 2^{32} corresponding outputs of $FL6$ sum up to zero. Formally, the equation we check is:

$$\sum_{i=1}^{2^{32}} O_i \oplus FL7^{-1}(C_i^R) \overset{?}{=} 0, \qquad (1)$$

since $O_i \oplus FL7^{-1}(C_i^R)$ equals the output of $FL6$ in the encryption of the i'th plaintext.

Due to the complex structure of the FO function and the final functions $FL7, FL8$, a direct check of Equation 1 is very time consuming. In the followings we show how a division of the FO and FL functions, along with a meet-in-the-middle approach, allows to check whether Equation (1) holds much more efficiently.

[5] We use the notation $\mathcal{B}$ to denote a word which is balanced (i.e., the XOR of the word in all the values of the structure is 0).

4.3 Dividing the *FO* and *FL* Functions

In [23], Sakurai and Zheng observed that given two inputs of the function $FO5$, 7 bits of the difference between the two corresponding outputs can be represented as the XOR of two values, where one of them depends only on the subkey $AKO_{5,1}$ and the other depends only on the subkey $AKO_{5,2}$, and all other subkeys are not involved. Indeed, we have

$$\begin{aligned} \Delta O^L_{\{15,14,\ldots,9\}} &= \Delta I^{2L} \oplus \Delta X^{1R}_{\{15,14,\ldots,9\}} \\ &= \Delta I^{2L} \oplus \Delta I^{1L} \oplus \Delta X^{R}_{\{15,14,\ldots,9\}}. \end{aligned} \tag{2}$$

By the structure of the function FI, the values I^{1L} and I^{2L} depend only on the subkeys $AKO_{5,1}$ and $AKO_{5,2}$, respectively.

If the final FL layer is removed from 5-round MISTY1, this observation allows to check the validity of the Square property efficiently. Indeed, it is sufficient to guess 32 key bits and check whether the XOR of the 2^{32} outputs of $FL6$ equals to zero in its 7 leftmost bits. Checking this condition for four Square structures is sufficient for discarding most of the possible values of the subkeys $KO_{5,1}$ and $KO_{5,2}$. Such an attack, along with some enhancements, was used in [16] to break 5-round MISTY1 without the last FL layer, with time complexity of 2^{48}. Later on, Sun and Lai [25] refined the Sakurai-Zheng relation and used it to improve the complexity of [16]'s attack to $2^{27.32}$ encryptions (without taking into account the time needed to encrypt 2^{34} plaintexts).

In order to handle the final FL layer, we further refine the Sakurai-Zheng property, and show that the FO function state can be divided not only into two 16-bit chunks, but actually into four chunks of sizes 7,9,7, and 9 bits, respectively.

We observe that due to the structure of FI, the value I^{1L} composed of bits $\{15, 14, \ldots, 9\}$ of I^1 can be further divided into the XOR of two values, such that one of them depends only on the 9 leftmost bits of $AKO_{5,1}$, and the second one depends only on the 7 rightmost bits of $AKO_{5,1}$. Indeed, we have

$$I^{1L} = S7(J^{1R}) \oplus \left(J^{1R} \oplus S9(J^{1L})\right),$$

where $S7(J^{1R}) \oplus J^{1r}$ depends only on the 7 least significant bits of $AKO_{5,1}$ and $S9(J^{1L})$ depends only on 9 most significant bits of $AKO_{5,1}$. The same holds also for $I^2_{\{15,14,\ldots,9\}}$. Substituting into Equation (2), we obtain:

$$\begin{aligned} \Delta O^L_{\{15,14,\ldots,9\}} &= \left(J^{2R} \oplus S7(J^{2R})\right) \oplus S9(J^{2L}) \oplus \left(J^{1R} \oplus S7(J^{1R})\right) \oplus S9(J^{1L}) \oplus \Delta X^R_{\{15,14,\ldots,9\}} \\ &= \underbrace{\left(S9(J^{2L}) \oplus S9(J^{1L})\right)}_{(\star)} \oplus \underbrace{\left((J^{2R} \oplus S7(J^{2R})) \oplus (J^{1R} \oplus S7(J^{1R}))\right)}_{(\star\star)} \oplus \Delta X^R_{\{15,14,\ldots,9\}}. \end{aligned} \tag{3}$$

In Equation (3), the first part $((\star))$ depends only on the 9 leftmost bits of both halves of the input to $FO5$ and on the 9 leftmost bits of the subkeys $AKO_{5,1}$ and $AKO_{5,2}$. Likewise, the second part $((\star\star))$ depends only on the 7 rightmost bits of both halves of the input to $FO5$ and on the 7 rightmost bits of the subkeys $AKO_{5,1}$ and $AKO_{5,2}$.

Now we observe that in 5-round MISTY1, the 9 leftmost bits and the 9 rightmost bits of the input to $FO5$, can be found given the ciphertext and 18 corresponding bits of the subkey KL_8. Likewise, the 14 bits of input to FO required in the second parentheses can be found given the 14 remaining bits of KL_8. Therefore, Equation (3), along with a meet-in-the-middle technique, allow us to break into parts both the subkeys $AKO_{5,1}$ and $AKO_{5,2}$ and the subkeys used in $FL8$.

4.4 Efficient Method to Check the Square Property Using Division and a Meet-in-the-Middle Approach

Recall that we consider Square structures, and want to check whether Equation (1) holds (in some of the bits). This will allow us to discard wrong suggestions of the subkey of round 5, and thus to retrieve subkey material. The condition we would like to check for a given key guess is whether

$$\left(\sum FL7^{-1}(C^R)\right)_{\{31,30,\ldots,25\}} = \left(\sum O^L\right)_{\{15,14,\ldots,9\}}.$$

Indeed, this is exactly Equation (1), restricted to the 7 leftmost bits. Using Equation (3), we can reformulate the condition we have to check as follows:

$$\begin{aligned}\left(\sum FL7^{-1}(C^R)\right)_{\{31,30,\ldots,25\}} =& \sum \left[\left(S9(J^{2L}) \oplus S9(J^{1L})\right) \oplus \left((J^{2R} \oplus S7(J^{2R})\right)\right] \oplus \\ & \sum \left[\left(J^{1R} \oplus S7(J^{1R}))\right) \oplus \sum X^R_{\{15,14,\ldots,9\}}\right],\end{aligned}$$

or equivalently (by rearranging the terms),

$$\begin{aligned}&\underbrace{\left(\sum FL7^{-1}(C^R)\right)_{\{31,30,\ldots,25\}} \oplus \sum \left((J^{2R} \oplus S7(J^{2R})) \oplus (J^{1R} \oplus S7(J^{1R}))\right) =}_{LHS} \\ &\qquad\qquad \underbrace{\sum \left[\left(S9(J^{2L}) \oplus S9(J^{1L})\right) \oplus \sum X^R_{\{15,14,\ldots,9\}}\right]}_{RHS}.\end{aligned} \tag{4}$$

Note that by the structure of the FL function, the value $\left(\sum FL7^{-1}(C^R)\right)_{\{31,30,\ldots,25\}}$ can be computed from the ciphertext given only the 7 leftmost bits of the subkey $KL_{7,2}$. Therefore, given the ciphertext, the left hand side of Equation (4) depends on 35 subkey bits (7 bits of KL_7, 14 bits of KL_8, 7 bits of $AKO_{5,1}$ and 7 bits of $AKO_{5,2}$) while the right hand side depends on 36 subkey bits (18 bits of KL_8, 9 bits of $AKO_{5,1}$ and 9 bits of $AKO_{5,2}$). We can thus check the validity of Equation (4) by a meet-in-the middle procedure between two evaluations that depend on 35 and 36 disjoint subkey bits, respectively.

4.5 Attack Algorithm

After discussing all the required components, the description of the attack is simple and straightforward. For the sake of clarity, we introduce a notation for the subkey bits used in the attack.

Let K_{LHS} denote the 36 subkey bits required for the computation of the left hand side (LHS) of Equation (4). These are bits $\{15, 14, \ldots, 7\}$ of the subkey words $KL_{8,1}, KL_{8,2}, AKO_{5,1}$ and $AKO_{5,2}$. Similarly, K_{RHS} (required for the RHS) is composed of bits $\{6, 5, \ldots, 0\}$ of $KL_{8,1}, KL_{8,2}, AKO_{5,1}$ and $AKO_{5,2}$ and of bits $\{15, 14, \ldots, 9\}$ of $KL_{7,2}$.

The attack algorithm is the following:

1. Ask for the encryption of 12 structures of 2^{32} chosen plaintexts, such that in each structure, the left 32 bits are held constant, and the right 32 bits take on all possible values.
2. For each guess of the 35 subkey bits denoted by K_{RHS}, compute the RHS of Equation (4) for each of the 12 structures, and store the 84-bit vector composed of the 12 values of the 7-bit RHS in a hash table.
3. For each guess of the 36 subkey bits denoted by K_{LHS}, compute the LHS of Equation (4) for each of the 12 structures, check whether the 84-bit vector composed of the 12 values of the LHS is in the hash table.

Since the table provides an 84-bit filtering and we examine only 2^{71} subkey suggestions, it is expected that only the right key guess suggests a collision in the table. Thus, the attack retrieves the value of 71 subkey bits (the full subkeys $KL_{8,1}, KL_{8,2}, AKO_{5,1}$, and $AKO_{5,2}$, and 7 bits of $KL_{7,2}$).

4.6 Efficient Implementation of the Attack and Its Analysis

In a naive implementation, the time complexity of the attack is about $12 \cdot 2^{36} \cdot 2^{32} = 2^{71.6}$ operations, since in Step 2, for each of the 12 structures, we have to compute the XOR of 2^{32} values, under each of 2^{36} subkey guesses. The time complexity can be dramatically reduced, using the *partial sum* technique, proposed by Ferguson et al. in their Square attack on reduced-round AES [10].

The partial sum technique suggests to use the fact that the Square structure can be replaced by a smaller structure which yields the same value of $\sum$, after only a part of the subkey bits are guessed. For example, due to the linearity of the FL function, we have

$$\sum FL7^{-1}(C^R) = FL7^{-1}\left(\sum C^R\right).$$

Hence, instead of computing $\sum FL7^{-1}(C^R)$ for each guess of the 7 bits of KL_7 separately (which would require $2^7 \cdot 2^{32} = 2^{39}$ operations), we can compute $\sum C^R$ (which requires 2^{32} XORs), and only then compute $FL7^{-1}(\sum C^R)$ for each guess of KL_7 (which requires only 2^7 operations).

As the application of the partial sum technique in computing the terms of Equation (4) is quite cumbersome, and all terms are computed roughly in the same way, we exemplify this application for one of the terms, and leave the rest of the terms to the reader.

Consider the term $\sum S9(J^{2L})$ which depends on 27 subkey bits. We perform its computation in several stages.

1. First, we proceed from C^L to $FL8^{-1}(C^L)$ which is the input to $FO5$. We observe that since the value of the 18 bits in $FL8^{-1}(C^L)$ we are interested in, depends on only 18 bits of C^L, then if two ciphertexts are equal in these 18 bits then their contributions to $\sum S9(J^{2L})$ cancel each other. Thus, before guessing KL_8, we can go over the ciphertexts and replace the sequence of 2^{32} ciphertexts with a sequence of length 2^{18} that indicates the *parity* of the number of appearances of each 18-bit value amongst these ciphertexts. This computation requires 2^{32} operations, but can be performed before any key material was guessed. The result of this step, is that the following operations are performed for "structures" of size 2^{18} rather than 2^{32}.
2. As a second step, we compute $\sum S9(J^{2L})$ for the reduced "structure". Since $S9(J^{2L})$ depends on only 9 of the 18 bits we computed in $FL8^{-1}(C^R)$, we can replace the sequence of 2^{18} parity bits with a shorter sequence of size 2^9 which corresponds to the 9 bits that influence $S9(J^{2L})$. This shrinking cannot be performed before the guess of the 18 bits of KL_8, but it can be performed before the guess of $AKO_{5,2}$. Therefore, under the "heavier" guess of $AKO_{5,2}$ we have to perform only 2^9 simple operations.
3. As a third step, we compute $\sum S9(J^{2L})$ independently of the guess of $AKO_{5,1}$ and store it in an auxiliary table corresponding to the guessed value of $AKO_{5,2}$. Then, the combination of the guesses of $AKO_{5,1}$ with the guesses of $AKO_{5,2}$ requires only a few operations for each pair of guesses.

Using this procedure, the computation of $\sum S9(J^{2L})$ can be performed in $2^{32} + 2^{18} \cdot 2^{18} + 2^{27} \cdot 2^9 + 2^{36} \cdot 2 = 2^{38}$ operations, instead of $2^{32} \cdot 2^{27} = 2^{59}$ operations.

All the other terms of Equation (4) can be also computed in time of at most 2^{38} simple operations for each Square structure. In total, the number of operations performed by the attack is approximately 2^{36} 1-round encryptions for each Square structure, which amount to less than 2^{38} encryptions. Hence, the data complexity of the attack is $2^{35.6}$ chosen plaintexts, and its time complexity is 2^{38} encryptions. The memory complexity is dominated by the storage of the plaintext/ciphertext pairs, which requires $2^{36.6}$ 64-bit blocks.

4.7 Retrieving the Rest of the Secret Key

The adversary can use similar techniques to retrieve the rest of the subkey used in AKO_5. First, she can perform a meet-in-the-middle procedure based on examining bits $\{24, 23, \ldots, 16\}$ of the output of $FL6$, and retrieve the subkeys $AKI_{5,1}, AKI_{5,2}$, and the 9 remaining bits of $KL_{7,2}$. Then, by checking the rest of the bits of $FL6$'s output, she can retrieve the subkeys $AKO_{5,3}, AKI_{5,3}$, and $KL_{7,1}$. By the key schedule, these subkeys supply the adversary with the key words $K2, K3, K4, K5, K6$, and $K7$. The two remaining key words can be found by exhaustive key search over 2^{32} possible values. Therefore, the entire secret key can be found in total time of 2^{38} encryptions.

4.8 Application to Other Variants of MISTY1

It is clear that the attack applies without change to a stronger variant of MISTY1 in which the FL layers are applied after every single round.

It is more interesting to note that a variant of the attack applies to 5-round KASUMI. Since in the design of KASUMI, the FI function was extended to 4 rounds, it becomes impossible to divide the state of FO to 4 chunks. However, since the FO function itself was not extended, division into two 16-bit chunks using the Sakurai-Zheng relation is still possible. This allows to check the Square condition by a meet-in-the-middle procedure, which uses an external guess of the 16-bit subkey $KL_{5,1}$, the subkeys $(KL_{5,2}, KO_{5,1}, KI_{5,1,2})$ from the one side and $(KO_{5,2}, KI_{5,2,2})$ from the other side. It appears that the partial sums technique also becomes less effective, and the total complexity of the attack becomes around 2^{68} operations. We do not present the details in full, since this attack is less efficient than the higher-order differential attack of Sugio et al. [24].

5 A Related-Key Slide Attack on 8-round MISTY1 without FL Layers

In a MISTY1 variant without the FL layers, each subsequent round subkey is a shift by one key word of the previous round subkey (see Table 2 and Figure 4). This allows mounting a related-key slide attack [2],[6] using the pair of related-keys $K = (K_1, K_2, \dots, K_8)$, $K^* = (K_2, K_3, \dots, K_8, K_1)$. As we show below, 2^{18} appropriately chosen plaintexts are expected to contain about four slid pairs, and these pairs can be efficiently identified. Once detected, they can be used to attack a single round of MISTY1 with several plaintext/ciphertext pairs. We follow and show that given these pairs, using the division of the FO function, we can efficiently recover the key. This yields a related-key attack on 8-round MISTY1 without the FL layers requiring only 2^{18} data and time. In the followings, we present the detailed description of the attack and report its experimental verification.

5.1 First Phase – Detecting the Related-Key Slid Pairs

Consider a pair of related-keys of the form $K = (K_1, K_2, \dots, K_8)$ and $K^* = (K_2, K_3, \dots, K_8, K_1)$ for 8-round MISTY1 without the FL layers. Note that due to the key schedule, for each $2 \leq i \leq 8$, the i-th round key corresponding to K^* is equal to the $i+1$-th round key corresponding to K. That is, $KO^*_{i,j} = KO_{i+1,j}$ and $KI^*_{i,j} = KI_{i+1,j}$ for all i, j. This implies that if for a pair of plaintexts (P, P^*), P^* is equal to the 1-round MISTY encryption of P under the subkey

[6] We note that in [2] the name of the attack is a related-key attack. However, due to the development of related-key attacks (and especially the introduction of related-key statistical attacks), we use the term related-key slide to refer to this sort of attack, which predates the slide attack.

$(KO_{1,1}, KO_{1,2}, KO_{1,3}, KO_{1,4}, KI_{1,1}, KI_{1,2}, KI_{1,3})$, then C^* is equal to the 1-round MISTY encryption of C under the 8th round subkey of K^* which happens to be:

$$(KO^*_{8,1}, KO^*_{8,2}, KO^*_{8,3}, KO^*_{8,4}, KI^*_{8,1}, KI^*_{8,2}, KI^*_{8,3}) = \\ (KO_{1,1}, KO_{1,2}, KO_{1,3}, KO_{1,4}, KI_{1,1}, KI_{1,2}, KI_{1,3})$$

(see Figure 4). We denote such pairs as *slid pairs*.[7]

Given the Feistel structure of MISTY1 (with no FL functions), it is easy to construct and detect slid pairs. This relies on the fact that if (P, C) and (P^*, C^*) compose a slid pair, then the following two relations must hold: $P^L = P^{*R}$ and $C^L = C^{*R}$. Since in the second phase of the attack we need at least three slid pairs, we construct them as follows:

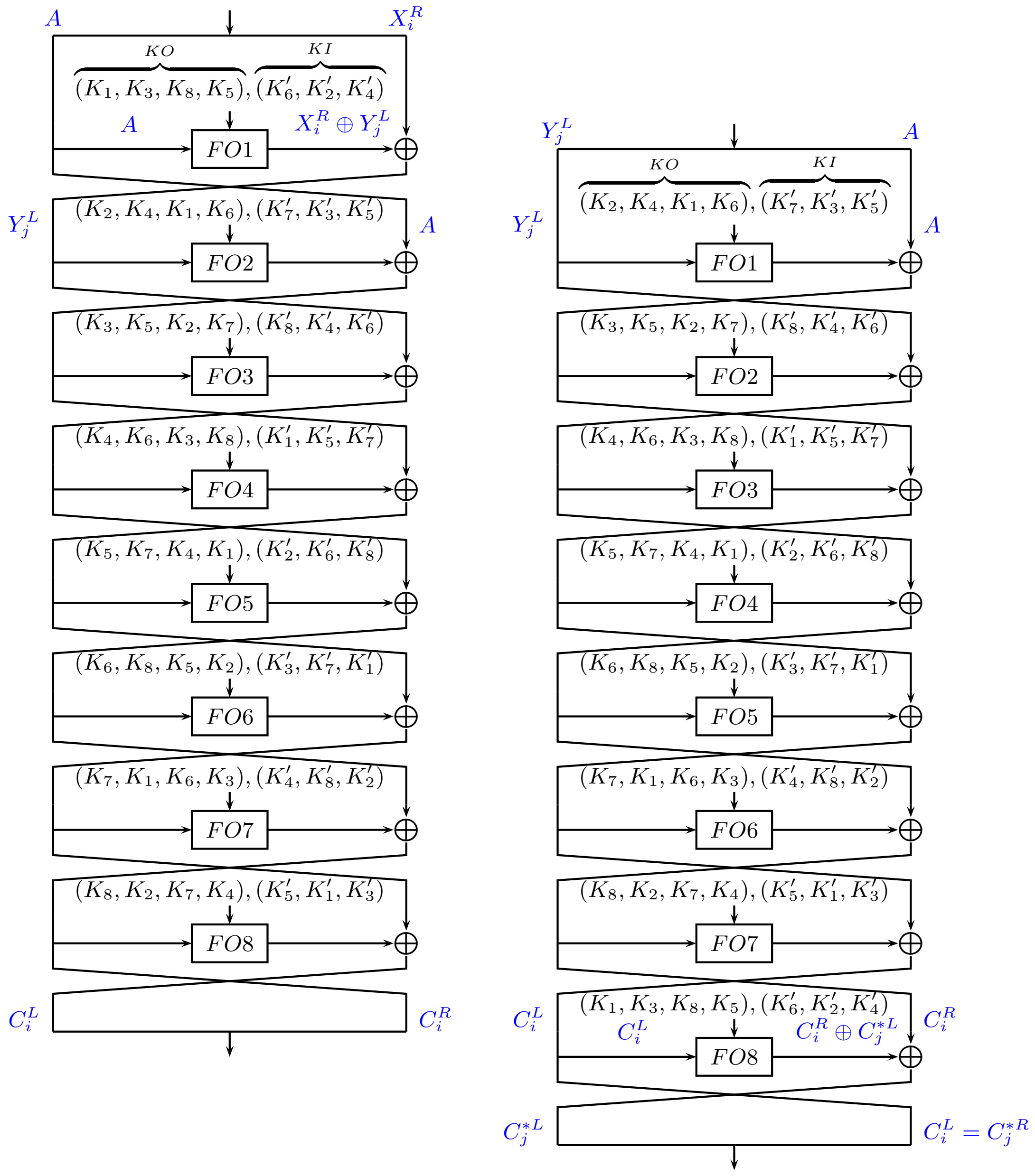
1. **Detecting Candidate Slid Pairs:**
   (a) Ask for the encryption, under the key K, of a structure of 2^{17} plaintexts of the form $P_i = (A, X_i)$, where A is a random 32-bit fixed value and X_i assumes 2^{17} different values. Denote the corresponding ciphertexts by C_i $(1 \le i \le 2^{17})$, and store the pairs (P_i, C_i) in a hash table sorted according to the value of C_i^L.
   (b) Ask for the encryption, under the key K^*, of a structure of 2^{17} plaintexts of the form $P_i^* = (Y_i, A)$, where A is the same 32-bit fixed value as in the previous step, and Y_i assumes 2^{17} different values. Denote the corresponding ciphertexts by C_j^* $(1 \le j \le 2^{17})$. For each pair (P_j^*, C_j^*), access the hash table in the cell corresponding to C^{*R} to find all pairs (P_i, C_i) such that $C_i^L = C_i^{*R}$.

This algorithm is an efficient way to check the equality in 32 bits among all the $2^{17} \cdot 2^{17} = 2^{34}$ pairs. Hence, we expect that about $2^{34} \cdot 2^{-32} = 4$ pairs pass this step randomly. Given the equality in the 32 bits which were set in the plaintext, the probability of a pair to indeed be a slid pair is 2^{-32}, which means we expect 4 slid pairs (in addition to the few random ones).

Fortunately, we can easily discard the false hits by examining the first round. For a slid pair $(P_i, C_i), (P_j^*, C_j^*)$, we know that P_j^* is equal to the 1-round encryption of P_i. In particular, this implies $FO1(A) \oplus X_i^R = Y_j^L$. Since A is constant for all plaintexts in the structures, we get that for all slid pairs, $X_i^R \oplus Y_j^L = B$, for some constant $B(= FO1(A))$. One can safely assume (as supported by our experiments), that the values of $X_i^R \oplus Y_j^L$ for wrong pairs which passed the filtering, are distributed randomly, thus we can identify the slid pairs as follows:

2. **Discarding the False Hits:** For each pair that passed the first step, compute the value of $X_i^R \oplus Y_j^L$. Discard all pairs whose result is unique.

[7] We again alert the reader to the abuse in notation, as slid pairs were defined for the slide attack, which is the original related-key attack in the cases that a single key is its own "related" one.



A
X_i^R
KO
KI
$(K_1, K_3, K_8, K_5), (K'_6, K'_2, K'_4)$
A
$X_i^R \oplus Y_j^L$
$FO1$
Y_j^L
$(K_2, K_4, K_1, K_6), (K'_7, K'_3, K'_5)$
A
$FO2$
$(K_3, K_5, K_2, K_7), (K'_8, K'_4, K'_6)$
$FO3$
$(K_4, K_6, K_3, K_8), (K'_1, K'_5, K'_7)$
$FO4$
$(K_5, K_7, K_4, K_1), (K'_2, K'_6, K'_8)$
$FO5$
$(K_6, K_8, K_5, K_2), (K'_3, K'_7, K'_1)$
$FO6$
$(K_7, K_1, K_6, K_3), (K'_4, K'_8, K'_2)$
$FO7$
$(K_8, K_2, K_7, K_4), (K'_5, K'_1, K'_3)$
$FO8$
C_i^L
C_i^R
Y_j^L
A
KO
KI
$(K_2, K_4, K_1, K_6), (K'_7, K'_3, K'_5)$
Y_j^L
A
$FO1$
$(K_3, K_5, K_2, K_7), (K'_8, K'_4, K'_6)$
$FO2$
$(K_4, K_6, K_3, K_8), (K'_1, K'_5, K'_7)$
$FO3$
$(K_5, K_7, K_4, K_1), (K'_2, K'_6, K'_8)$
$FO4$
$(K_6, K_8, K_5, K_2), (K'_3, K'_7, K'_1)$
$FO5$
$(K_7, K_1, K_6, K_3), (K'_4, K'_8, K'_2)$
$FO6$
$(K_8, K_2, K_7, K_4), (K'_5, K'_1, K'_3)$
$FO7$
$(K_1, K_3, K_8, K_5), (K'_6, K'_2, K'_4)$
C_i^L
C_i^L
$C_i^R \oplus C_j^{*L}$
C_i^R
$FO8$
C_j^{*L}
$C_i^L = C_j^{*R}$

Fig. 4. Two Related-Key MISTY1 Encryptions

With an overwhelming probability, only the slid pairs remain after this stage. Since the number of slid pairs follows the $Poisson(4)$ distribution, the probability that the data contains at least three slid pairs is $1 - e^{-4}(1 + 4 + 16/2) = 0.76$. We assume that this is the case, and denote these pairs (w.l.o.g.)

by $((P_i, C_i), (P_i^*, C_i^*))$, for $i = 1, 2, 3$. Each of these slid pairs yields two input/output pairs for the FO function under the subkey $(KO_{1,1}, KO_{1,2}, KO_{1,3}, KO_{1,4}, KI_{1,1}, KI_{1,2}, KI_{1,3})$: the pair $(A, X_i^R \oplus Y_i^L)$ obtained from the first round, and the pairs $(C_i^L, C_i^R \oplus C_i^{*L})$ (for $i = 1, 2, 3$) obtained from the last round. Hence, we obtain at least four pairs of input/output to this round function, denoted by $(\mathcal{I}_j, \mathcal{O}_j)$ for $j = 1, \ldots, 4$, (we note that sometimes we obtain more such input/output pairs), which can then be used in the analysis.

5.2 Second Phase – Dividing *FO* Yet Another Time

In order to retrieve the subkey of the FO function efficiently, we divide its state once again. Similarly to Section 4, we consider the 7 leftmost bits of FO's output and use a meet-in-the-middle technique to find the subkeys $AKO_{1,1}$ and $AKO_{1,2}$. In the following, we use the notations introduced in Section 4 (with $FO1$ replacing $FO5$), including the equivalent description of the FO function.

The first step of this phase of the attack is as follows:

1. **Retrieving the subkeys $AKO_{1,1}$ and $AKO_{1,2}$:**
   (a) For each value of the subkey $AKO_{1,1}$, partially encrypt the four inputs $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,1}$ to obtain the four intermediate values at the state I_1^{1L}. Denote the four obtained values by Z_1, Z_2, Z_3, Z_4. Store in a hash table the 21-bit vector
   $$(Z_1 \oplus Z_2, Z_1 \oplus Z_3, Z_1 \oplus Z_4) \oplus (\mathcal{I}_1^R \oplus \mathcal{I}_2^R, \mathcal{I}_1^R \oplus \mathcal{I}_3^R, \mathcal{I}_1^R \oplus \mathcal{I}_4^R)_{\{15,14,\ldots,9\}}$$
   $$\oplus (\mathcal{O}_1^L \oplus \mathcal{O}_2^L, \mathcal{O}_1^L \oplus \mathcal{O}_3^L, \mathcal{O}_1^L \oplus \mathcal{O}_4^L)_{\{15,14,\ldots,9\}}.$$
   (b) For each value of the subkey $AKO_{1,2}$, partially encrypt the four inputs $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,2}$ to obtain the four intermediate values at the state I_1^{2L}. Denote the four obtained values by W_1, W_2, W_3, W_4. Access the hash table with the 21-bit vector $(W_1 \oplus W_2, W_1 \oplus W_3, W_1 \oplus W_4)$ and search for collisions.

By the analysis of the Sakurai-Zheng property presented in Section 4.3, a collision in the table is equivalent to satisfying

$$(\mathcal{O}_1^L \oplus \mathcal{O}_j^L)_{\{15,\ldots,9\}} \oplus (C_1^{*L} \oplus C_j^{*L})_{\{15,\ldots,9\}} = (C_1^R \oplus C_j^R)_{\{15,\ldots,9\}},$$

for $j = 2, 3, 4$. Obviously, this happens when the right keys are used, which means that of the $2^{16} \cdot 2^{16} = 2^{32}$ candidates for $AKO_{1,1}$ and $AKO_{1,2}$, about $2^{32} \cdot 2^{-21} = 2^{11}$ satisfy the relation for $j = 2, 3, 4$, including the right subkey guess.

In the next step of the attack, we try all these 2^{11} possible values and recover the values of the subkeys $AKI_{1,1}$ and $AKI_{1,2}$, while discarding some wrong key guesses. This step of the attack completes the evaluation of the Sakurai-Zheng relation in the 9 bits which were not earlier checked, as follows:

2. **Retrieving the subkeys $AKI_{1,1}$ and $AKI_{1,2}$:** For each remaining suggestion for $AKO_{1,1}$ and $AKO_{1,2}$ from Step 1, perform the following:

(a) For each value of the subkey $AKI_{1,1}$, partially encrypt the four plaintexts $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,1}$ to obtain the four intermediate values at the state I_1^{1R}. Denote the four obtained values by Z'_1, Z'_2, Z'_3, Z'_4. Store in a hash table the 27-bit vector

$$(Z'_1 \oplus Z'_2, Z'_1 \oplus Z'_3, Z'_1 \oplus Z'_4) \oplus (\mathcal{I}_1 \oplus \mathcal{I}_2, \mathcal{I}_1 \oplus \mathcal{I}_3, \mathcal{I}_1 \oplus \mathcal{I}_4)_{\{8,7,\ldots,0\}}$$
$$\oplus (\mathcal{O}_1^L \oplus \mathcal{O}_2^L, \mathcal{O}_1^L \oplus \mathcal{O}_3^L, \mathcal{O}_1^L \oplus \mathcal{O}_4^L)_{\{8,7,\ldots,0\}}.$$

(b) For each value of the subkey $AKI_{1,2}$, partially encrypt the four plaintexts $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,2}$ to obtain the four intermediate values at the state I_1^{2R}. Denote the four obtained values by W'_1, W'_2, W'_3, W'_4. Access the hash table with the 27-bit vector $(W'_1 \oplus W'_2, W'_1 \oplus W'_3, W'_1 \oplus W'_4)$ and search for collisions.

By the analysis of the Sakurai-Zheng property presented in Section 4.3, a collision in the table is equivalent to having

$$(\mathcal{O}_1^L \oplus \mathcal{O}_j^L)_{\{8,\ldots,0\}} \oplus (C_1^{*L} \oplus C_j^{*L})_{\{8,\ldots,0\}} = (C_1^R \oplus C_j^R)_{\{8,\ldots,0\}},$$

for $j = 2, 3, 4$. As for the right key guess, these three equations are necessarily satisfied, the correct value of the subkeys $AKI_{1,1}$ and $AKI_{1,2}$ is suggested by one of the collisions. At the same time, the number of suggested subkeys is $2^{11} \cdot 2^9 \cdot 2^9 \cdot 2^{-27} = 4$. Hence, we remain with only about 4 suggestions for the 50 subkey bits $AKO_{1,1}, AKO_{1,2}, AKI_{1,1}, AKI_{1,2}$.

The rest of the round subkey can be now found easily, as follows:

3. **Retrieving the rest of the round subkey:** For each remaining suggestion for $AKO_{1,1}, AKO_{1,2}, AKI_{1,1}, AKI_{1,2}$, perform the following:

(a) For each value of the subkey $AKO_{1,3}$, partially encrypt the four plaintexts $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,3}$ to obtain the four intermediate values at the state I_1^{3L}. Denote the four obtained values by $Z''_1, Z''_2, Z''_3, Z''_4$. Use the values $Z''_1 \oplus Z''_2, Z''_1 \oplus Z''_3, Z''_1 \oplus Z''_4$ to compute the 21-bit vector $(\mathcal{O}_1^R \oplus \mathcal{O}_2^R, \mathcal{O}_1^R \oplus \mathcal{O}_3^R, \mathcal{O}_1^R \oplus \mathcal{O}_4^R)_{\{15,14,\ldots,9\}}$ and compare it with the known values.

(b) For each value of the subkey $AKI_{1,3}$, partially encrypt the four plaintexts $(\mathcal{I}_1, \ldots, \mathcal{I}_4)$ through the function $FI_{1,3}$ to obtain the four intermediate values at the state I_1^{3R}. Denote the four obtained values by $W''_1, W''_2, W''_3, W''_4$. Use the values $W''_1 \oplus W''_2, W''_1 \oplus W''_3, W''_1 \oplus W''_4$ to compute the 27-bit vector $(\mathcal{O}_1^R \oplus \mathcal{O}_2^R, \mathcal{O}_1^R \oplus \mathcal{O}_3^R, \mathcal{O}_1^R \oplus \mathcal{O}_4^R)_{\{8,\ldots,0\}}$ and compare it with the known values.

(c) For each remaining value of the subkeys $AKO_{1,3}, AKI_{1,3}$, encrypt $\mathcal{I}_1$ through the entire FO except for the last subkey addition to obtain the value of the state X_1^3, and compute the subkeys $AKO_{1,4}, AKO_{1,5}$ using the formula

$$(X_1^{3L}, X_1^{3R}) \oplus (AKO_{1,4}, AKO_{1,5}) = (\mathcal{O}^L, \mathcal{O}^R).$$

The condition in Step 3(a) offers a 21-bit filtering on the key value. Thus, only the correct suggestion of the subkey $AKO_{1,1}, AKO_{1,2}, AKI_{1,1}, AKI_{1,2}, AKO_{1,3}$ is expected to remain after that step. Steps 3(b) and 3(c) each yield single values of the subkeys $AKI_{i,3}, AKO_{1,4}$, and $AKO_{1,5}$, thus yielding a single suggestion for the 107-bit equivalent round subkey.

The most time-consuming step in the second phase of the attack is Step 2, which is composed of 2^{10} simple operations performed for each of 2^{11} suggestions of $AKO_{1,1}, AKO_{1,2}$. As this step is much faster than 2^{18} encryptions, the overall complexity of the attack is dominated by the encryption of 2^{18} chosen plaintexts performed at the beginning of the attack. Therefore, the data, memory and time complexities of the attack are as low as 2^{18}.

5.3 Retrieving the Rest of the Secret Key

By guessing the 7 leftmost bits of the subkeys K'_2 and K'_6 and using the retrieved subkey of KO_1, the adversary obtains 2^{14} suggestions for the key words K_1, K_3, K_5, K_8 and the subkey words K'_2, K'_4, K'_6. By the construction of the subkeys, this yields also the key words K_2, K_4, and with an additional 16-bit guess of K_6 this yields 2^{30} suggestions for the full secret key (K_7 can be derived from K_6 and K'_6). Hence, the secret key can be found by an exhaustive search over 2^{30} possible values. In order to further improve this complexity, the adversary must analyze also other rounds, which can be obtained, e.g., by peeling off the first round and applying a related-key slide attack on a 7-round variant. However, this would require an increase in the data complexity and in the number of related-keys.

5.4 Experimental Verification

To verify this attack, we have executed two experiments. The first experiment took 1,000,000 random keys, and generated 2^{18} pairs as suggested in Section 5.1. We verified that the amount of slid pairs indeed follows the expected distribution, and that on very rare occasions (about 130), one can expect an additional wrong value to be suggest by two pairs which are not slid pairs (as expected from a random process). Table 3 reports the number of slid pairs found in each experiment compared with the expected outcome.

The second experiment we conducted was running the full attack algorithm for 10,000 different keys. Out of these 10,000 experiments, 7,643 ended with success (i.e., the full 107-bit key was recovered), whereas the remaining experiments failed (all but one due to lack of sufficient slid pairs). This part of the experiment was timed using the basic *clock_gettime* call. The running times of the failed cases and the successful cases varied — as the key recovery phase was not invoked unless 3 slid pairs were found. For failed attempts, the average running time was 0.05216 seconds, whereas the average running time for successful attacks was 0.14949 seconds for both data generation and the key recovery phase. The experiment was carried on an Intel i7-3520 machine running at 2.9 GHz, running Linux 3.2.0-23, compiled with gcc 4.6.3 (with a single optimization flag

"Slid" Pairs	0	1	2	3	4	5
Theory ($Poi(4)$)	18,316	73,263	146,525	195,367	195,367	156,293
Experiment	18,324	73,461	146,699	195,390	194,541	156,609
"Slid" Pairs	6	7	8	9	10	11
Theory ($Poi(4)$)	104,196	59,540	29,770	13,231	5,292	1,925
Experiment	104,266	59,338	29,860	13,330	5,348	1,916
"Slid" Pairs	12	13	14	15	16	17
Theory ($Poi(4)$)	641	197	56	15	4	1
Experiment	657	190	54	15	2	0

Table 3. The Results of 1,000,000 Experiments to Generate (and Locate) the Slid Pairs

	Total	Successful Trials	Failed Trials
Number	10,000	7,643	2,367
(Expected)	10,000	7,619	2,381
Average Running Time (sec)	0.12655	0.14949	0.05216
Maximal Running Time (sec)	0.21640	0.21640	0.15159
Minimal Running Time (sec)	0.05096	0.13706	0.05096
Standard Deviation (sec)	0.04162	0.00546	0.00337

Table 4. The Results of 10,000 Full Runs of Our Related-Key Attack

"-O2"). All relevant parts of the code were taken from the Misty1's submission to the NESSIE project. We report in Table 4 the statistical data concerning this experiment.

Hence, we conclude that our attack on MISTY1 with no FL layers is valid and of practical complexities.

5.5 Applicability to Other Variants of MISTY1

The same attack can be applied to cases with any number of rounds of MISTY1 without the FL functions. In these cases, slightly more data is needed to ensure that there are indeed four input/output pairs to the analyzed round function (as the first and last rounds may not share the exact subkeys). In exchange, one can easily retrieve the full key by applying the analysis to the first round (in addition to the last round).

An interesting observation is that a small modification of the attack applies to the stronger MISTY1 variant in which FL functions are applied after every round, discussed in Section 4.8. Indeed, in such a variant, the FL keys also satisfy the cyclic property satisfied by the FO keys (see Table 2), and hence, the basic related-key slide attack can be applied. However, as a single round in this modified construction contains not only a Feistel round but also an FL function, the detection of the slid pairs becomes more complicated, which requires guessing the subkeys of the last rounds right hand side FL function (the one in K^*), and repeating the attack of Section 4.5 for each such guess. This results in an attack with data complexity of roughly 2^{18} chosen plaintexts, and time complexity of about 2^{68} encryptions.

When the adversary has access to encryption under 8 related-keys which compose all the word-wise cyclic shifts of a single key, one can run a significantly more efficient attack by attacking the FL functions (like in Section 3). This is done by using the fact that when a related-key slid pair is encrypted (under the respective keys), it maintains the slid pair property [5, 11]. The adversary considers the above pair of structures of size 2^{17} each, and generates from each plaintext, a sequence of a few adaptively chosen plaintexts, by sequential encryption. Then, in order to check whether a given pair of values is a slid pair, the adversary looks at the subsequent plaintext pairs, and checks a sequence of conditions of the form $FL1(P^{iL}) = P^{*iR}$, which must hold simultaneously (where $FL1$ is the FL function applied to the left half of the state in the first round). Using the procedure of Section 3, one can identify the correct $FL2$ key in time of a few operations for each candidate slid pair, and use a few additional pairs from the encrypted sequence as a filtering check. This allows to retrieve the right slid pairs, along with the two subkey words $KL_{1,1}, KL_{1,2}$ in time of about 2^{34} (since 2^{34} candidate pairs are checked). Depending on the exact number of rounds in the MISTY1 variant, one can either perform the same attack on other FL functions (taking into consideration slid pairs with other offsets), or apply the attack of Section 4.5. Hence, with data complexity of about 2^{24} adaptively chosen plaintexts encrypted under 8 related-keys (where 2^{21} plaintexts are encrypted under each key), one can recover the full key in time of about 2^{36} encryptions (or even less).

It should be mentioned that unlike the Square attack presented in Section 4, this attack was completely thwarted by the designers of KASUMI [27]. In KASUMI, though the key schedule is simpler than that of MISTY1, it contains distinct round constants which are added to every round subkey. These constants seem to prevent any related-key slide attack.

6 Summary and Conclusions

In this paper, we considered attacks with a practical time complexity against reduced variants of MISTY1. We presented an attack on 5-round MISTY1 with all the FL layers present which requires 2^{38} encryptions, and an extremely efficient related-key attack on 8-round MISTY1 without the FL layers requiring 2^{18} encryptions. The related-key attack was fully verified experimentally.

Our attacks point at three weaknesses in the components of MISTY1:

1. The 3-round Feistel structure of the FO and FI functions allows to divide the state of FO into four smaller parts of $7, 9, 7, 9$ bits each, whose interaction is limited.
2. The FL function can be divided into sixteen 2-bit functions applied in parallel, and the same holds for a sequential application of several FL functions.
3. The key schedule of $MISTY1$ without the FL functions lacks round constants, and hence, makes this variant susceptible to related-key slide attacks.

We note that this seems to be the first case of a "reasonable" cipher variant whose security completely collapses (up to the point of a practical time complexity attack) against this sort of related-key attacks.

As we showed, combinations of these weaknesses can be deployed by an adversary to mount practical-time attacks on 5-round MISTY1 with all FL layers, and on the full MISTY1 without the FL layers (the latter in the related-key model).

In comparison, it is interesting to see how these weakness were handled by ETSI's SAGE task force working for the GSM association in the design of KASUMI (based on MISTY1):

1. The FI function was strengthened by adding a fourth round to the Feistel structure, while the FO function remained with a 3-round structure. As we showed in Section 4.8, this thwarts the Square attack only partially, still allowing to divide the state into two 16-bit parts. As a result, a Square attack with complexity of 2^{68} operations is applicable to 5-round KASUMI.
2. A rotation by one bit was added to the FL function, thus making it impossible to divide it into 16 independent functions.
3. Round constants were inserted into the round subkeys, thwarting the related-key slide attack completely. On the other hand, the key schedule was simplified, which led to a practical-time related-key attack on the full KASUMI [9], which does not apply to MISTY1.

Our conclusion is that while our results clearly do not pose any threat to the security of the full MISTY1 block cipher, they point out weaknesses of its components, which should be avoided in future designs based on MISTY1.

Acknowledgements

The first author was supported in part by the German-Israeli Foundation for Scientific Research and Development through grant No. 2282-2222.6/2011. The second author was supported by the Alon Fellowship.

References

1. Steve Babbage and Laurent Frisch, *On MISTY1 higher order differential cryptanalysis*, proceedings of ICISC 2000, Lecture Notes in Computer Science 2015, pp. 22–36, Springer-Verlag, 2001.
2. Eli Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, J. Cryptology Vol. 7 No. 4, pp. 229–246, 1994.
3. Eli Biham, Alex Biryukov, and Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 1999, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, 1999.
4. Eli Biham, Alex Biryukov, and Adi Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds*, Advances in Cryptology, proceedings of EUROCRYPT 1999, Lecture Notes in Computer Science 1592, pp. 12–23, Springer-Verlag, 1999.

5. Eli Biham, Orr Dunkelman, and Nathan Keller, *A Unified Approach to Related-Key Attacks*, proceedings of Fast Software Encryption 2008, Lecture Notes in Computer Science 5086, pp. 73–96, Springer-Verlag, 2008.
6. Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
7. Joan Daemen, Lars Knudsen, and Vincent Rijmen, *The Block Cipher Square*, proceedings of Fast Software Encryption 1997, Lecture Notes in Computer Science 1267, pp. 149–165, Springer-Verlag, 1997.
8. Orr Dunkelman and Nathan Keller, *An Improved Impossible Differential Attack on MISTY1*, Advanced in Cryptology, proceedings of ASIACRYPT 2008, Lecture Notes in Computer Science 5350, pp. 441–454, Springer-Verlag, 2008.
9. Orr Dunkelman, Nathan Keller, and Adi Shamir, *A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony*, Advanced in Cryptology, proceedings of CRYPTO 2010, Lecture Notes in Computer Science 6223, pp. 393–410, Springer-Verlag, 2010.
10. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, proceedings of Fast Software Encryption 2000, Lecture Notes in Computer Science 1978, pp. 213–230, Springer-Verlag, 2001.
11. Soichi Furuya, *Slide Attacks with a Known-Plaintext Cryptanalysis*, proceedings of Information and Communication Security 2001, Lecture Notes in Computer Science 2288, pp. 214–225, Springer, 2002.
12. ISO/IEC, *ISO/IEC 18033-3:2010 Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers*, 2010.
13. Keting Jia and Leibo Li, *Improved Impossible Differential Attacks on Reduced-round MISTY1*, proceedings of WISA 2012, to appear in Lecture Notes in Computer Science. Available online at: http://www.wisa.or.kr/.
14. Keting Jia, Honbo Yu, and Xiaoyun Wang, *A Meet-in-the-Middle Attack on the Full KASUMI*, Cryptology ePrint Archive: Report 2011/466.
15. Lars R. Knudsen, *The Security of Feistel Ciphers with Six Rounds or Less*, Journal of Cryptology, Vol. 15, No. 3, pp. 207–222, 2002.
16. Lars R. Knudsen and David Wagner, *Integral Cryptanalysis*, proceedings of Fast Software Encryption 2002, Lecture Notes in Computer Science 2365, pp. 112–127, Springer-Verlag, 2002.
17. Ulrich Kühn, *Cryptanalysis of Reduced-Round MISTY*, Advances in Cryptology, proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science 2045, pp. 325–339, Springer-Verlag, 2001.
18. Ulrich Kühn, *Improved cryptanalysis of MISTY1*, proceedings of Fast Software Encryption 2002, Lecture Notes in Computer Science 2365, pp. 61–75, Springer-Verlag, 2002.
19. Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman, *Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1*, proceedings of CT-RSA 2008, Lecture Notes in Computer Science 4964 pp. 370–386, Springer-Verlag, 2008.
20. Mitsuru Matsui, *Block encryption algorithm MISTY*, proceedings of Fast Software Encryption 1997, Lecture Notes in Computer Science 1267, pp. 64–74, Springer-Verlag, 1997.
21. Mitsuru Matsui, *A Description of the MISTY1 Encryption Algorithm*, RFC 2994, November 2000.
22. NESSIE, *Portfolio of recommended cryptographic primitives.*

23. Kouichi Sakurai and Yuliang Zheng, *On Non-Pseudorandomness from Block Ciphers with Provable immunity Against Linear Cryptanalysis*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E80-A, No. 1, pp. 19-24, 1997.
24. Nobuyuki Sugio, Hiroshi Aono, Sadayuki Hongo, and Toshinobu Kaneko, *A Study on Higher Order Differential Attack of KASUMI*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E90-A, No. 1, pp. 14–21, 2007.
25. Xiaorui Sun and Xuejia Lai, *Improved Integral Attacks on MISTY1*, proceedings of Selected Areas in Cryptography 2009, Lecture Notes in Computer Science 5867, pp. 266–280, Springer-Verlag, 2009.
26. Hidema Tanaka, Kazuyuki Hisamatsu, and Toshinobu Kaneko, *Strength of MISTY1 without FL function for higher order differential attack*, proceedings of AAECC-99, Lecture Notes in Computer Science 1719, pp. 221–230, Springer-Verlag, 1999.
27. 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.