Improved Cryptanalysis of MISTY1

Ulrich Kühn

Dresdner Bank AG
IS-STA Software Technology and Architecture
Research & Innovations
Jürgen-Ponto-Platz 1
D-60301 Frankfurt, Germany
Ulrich.Kuehn@dresdner-bank.com
ukuehn@acm.org

Abstract. The block cipher MISTY1 [9] proposed for the NESSIE project [11] is a Feistel network augmented with key-dependent linear FL functions. The proposal allows a variable number of rounds provided that it is a multiple of four.
Here we present a new attack – the Slicing Attack – on the 4-round version, which makes use of the special structure and position of these key-dependent linear FL functions. While the FL functions were introduced to make attacks harder, they also present a subtle weakness in the 4-round version of the cipher.
Keywords: Block cipher, Cryptanalysis, Impossible Differential, Slicing Attack.

1 Introduction

The MISTY1 block cipher [9] is a proposal for the NESSIE project [11] in the class *Normal-Legacy* with a block size of 64 bits and a key length of 128 bits. It is designed to be resistant against differential [2] and linear [8] cryptanalysis. Another feature of the design is the use of key-dependent linear functions FL to avoid possible attacks other than differential and linear cryptanalysis.

The best previous attacks on versions of MISTY1 without the linear functions attack were on 5 rounds by higher-order differentials [12] and 6 rounds with impossible differentials [7]. Additionally, the 4-round version including most of the linear functions, leaving out the layer of final applications of the FL functions, has been attacked by impossible differentials as well as collision searching [7]. Very recent results [6] using integral cryptanalysis yield attacks on 5 rounds of MISTY1 without the final FL layer as well as on 4 rounds, also without the final FL layer, having a very low data complexity.

In this paper we present attacks on the 4-round version of MISTY1 with all FL functions by impossible differentials and by a new method called the Slicing Attack. The slicing attack makes use of the position and the structure of the key-dependent linear functions to derive knowledge about the key; further key bits can then be found with impossible differentials, or, in the chosen plaintext/ciphertext model, by the meet-in-the-middle technique. Table 1 shows a summary of the attacks.

While the computational effort for the attack using only impossible differentials is very high, the slicing attack is surprisingly efficient; the existence of this attack shows that augmenting the Feistel network with the linear FL functions, which makes some attacks much harder, also introduces a new line of attack that has to be considered a subtle weakness not being present in the underlying Feistel network.

Rounds	FL	Complexity			Comments
		Time	Data	Memory	
5	none	2^{17}	11×2^7		[12]
6	none	2^{61}	2^{54}		[7], Section 4.1
4	most	$2^{90.4}$	2^{23}		[7], Section 4.2
4	most	2^{62}	2^{38}		[7], Section 4.2
4	most	2^{89}	2^{20}		[7], Section 4.2
4	most	2^{76}	2^{28}		[7], Section 4.2
4	most	2^{27}	25		[6]
5	most	2^{48}	2^{34}		[6]
4	all	2^{116}	$2^{27.5}$	$2^{29.5}$	Impossible diff. (this paper)
4	all	2^{45}	$2^{22.25}$	$2^{31.2}$	Slicing Attack, preprocessing (this paper)
4	all	$2^{81.6}$	$2^{27.2}$	$2^{31.2}$	Slicing + impossible diff. (this paper)
4	all	2^{48}	$2^{23.25}$	2^{33}	Slicing Attack in chosen plaintext / ciphertext model (this paper)

Table 1. Summary of the new and the best previously known attacks on MISTY1. A memory unit is one block of 64 bits. Versions of MISTY1 with "most" FL functions do not have the final FL layer.

This paper is outlined as follows. In Section 2 the MISTY1 design is described, Section 3 presents the attack on 4-round MISTY1 using impossible differentials alone, Section 4 introduces the Slicing Attack, and finally Section 5 draws some conclusions.

2 The Structure of MISTY1

The MISTY1 [9] proposal for the NESSIE project [11] is a block cipher with a 64-bit block and a 128-bit key. It consists of a Feistel network augmented by applying key-dependent linear functions FL to the left resp. right half of the data in every second round, starting with the first, and additionally after all the rounds (see left half of Figure 1). While the cipher is proposed with 8 rounds, the proposal allows a variable number of rounds provided that it is a multiple of four. In this paper we will only consider the 4-round version.

The bijective round function FO is a 3-round network with a structure shown in the right half of Figure 1. This network uses a bijective inner round function FI, which itself is a 3-round network with the same structure, employing two bijective S-boxes $S9$ and $S7$, which are 9 bits resp. 7 bits wide; the key to FI is 16 bits wide. The details of the internal structure of FI will be of no further concern in this paper.

The FL function is a linear or affine function for any fixed key; its internal structure is a 2-round Feistel network (see Figure 2) with the round functions being bitwise boolean AND resp. bitwise OR with key material.

The key scheduling takes a 128-bit key consisting of 16-bit values $K_1, \ldots, K_8$ and, as a first step, computes additional 16-bit values $K'_t = \mathrm{FI}_{K_{t+1}}(K_t)$, $1 \leq t \leq 8$, $K_9 := K_1$. It produces three streams of sub-keys $\mathrm{KO}_i = (\mathrm{KO}_{i1}, \ldots, \mathrm{KO}_{i4})$, $\mathrm{KI}_i = (\mathrm{KI}_{i1}, \ldots, \mathrm{KI}_{i3})$, and $\mathrm{KL}_i = (\mathrm{KL}_{i1}, \mathrm{KL}_{i2})$ as follows (i is identified with $i - 8$ for $i > 8$):

KO_{i1}	KO_{i2}	KO_{i3}	KO_{i4}	KI_{i1}	KI_{i2}	KI_{i3}	KL_{i1}	KL_{i2}
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)

3 Differential Attack on 4-Round MISTY1

The attack given in this section works against the 4-round version of MISTY1 with all FL functions, improving the result of [7] as there the final applications of the FL functions were left out. The attack applies impossible differentials [1, 5] and uses particular properties of the key scheduling, i.e. the fact that the keys for the final FL functions and the fourth round have some key bits in common.

To be concrete, these sub-keys are $\mathrm{KO}_4 = (K_4, K_6, K_3, K_8)$ and $\mathrm{KI}_4 = (K'_1, K'_5, K'_7)$ for the fourth round's FO resp. $\mathrm{KL}_5 = (K_3, K'_1)$ and $\mathrm{KL}_6 = (K'_5, K_7)$ for the final FLs. The values K'_1, K'_5, and K_3 are used twice.

For the attack we use Property 1 of FO from [7]:

Property 1. If the output difference of FO in round i is of the form (β, β) with nonzero β from input with difference (α_l, α_r), then the input and output differences of FI in the third round are zero; thus the sub-keys KO_{i3}, KO_{i4}, and KI_{i3} cannot influence the output difference and are consequently of no concern. The inputs to the first FI with difference α_l yield an output difference α_r under the keys KO_{i1} and KI_{i1}, while the second FI yields output difference β from the inputs with difference α_r under the keys KO_{i2} and KI_{i2}.

The attack makes use of the 3-round impossible differential $(0, \alpha) \overset{3R}{\not\to} (0, \beta)$ with nonzero α, β; this impossible differential works for any Feistel network with bijective round functions, even when FL functions are used [7, Section 4.2]. The attack proceeds as follows.

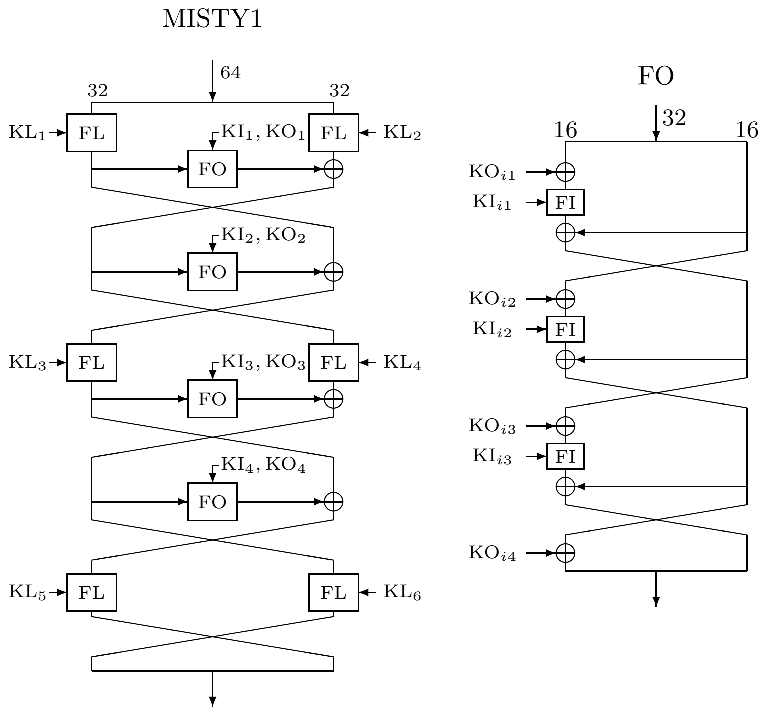

Fig. 1. Global structure of MISTY1 with four rounds (left) and structure of outer round functions FO (right).

1. *Data Collection.* Build a structure of $2^{27.5}$ chosen plaintexts $P_i = (x, y, a_i, b_i)$ where all the (a_i, b_i) are different and obtain the corresponding ciphertexts $C_i = (c_i, d_i, e_i, f_i)$ by encryption under the unknown key.
2. *Processing.* After guessing $\mathrm{KL}_6 = (K_5', K_7)$ and $\mathrm{KL}_5 = (K_3, K_1')$ obtain $\tilde{C}_i = (\tilde{c}_i, \tilde{d}_i, \tilde{e}_i, \tilde{f}_i)$ by $(\tilde{c}_i, \tilde{d}_i) = \mathrm{FL}^{-1}_{\mathrm{KL}_6}(c_i, d_i)$, $(\tilde{e}_i, \tilde{f}_i) = \mathrm{FL}^{-1}_{\mathrm{KL}_5}(e_i, f_i)$, and compute $w_i = \tilde{e}_i \oplus \tilde{f}_i$. Every matching pair (i, j) with $w_i = w_j$ results in a difference $\tilde{C}_i \oplus \tilde{C}_j = (\alpha_l, \alpha_r, \delta, \delta)$. For each such matching pair (i, j) do the following steps.
   - (Round 1 of FO) Guess the value of $K_4 = \mathrm{KO}_{41}$ ($K_1' = \mathrm{KI}_{41}$ is already known) and check if
     $$\mathrm{FI}_{K_1'}(\tilde{c}_i \oplus K_4) \oplus \mathrm{FI}_{K_1'}(\tilde{c}_j \oplus K_4) = \tilde{d}_i \oplus \tilde{d}_j, \tag{1}$$
     where K_1' is known from the guess of KL_5. Expect a single guess for K_4 to fulfill this condition.
   - (Round 2 of FO) Independently, guess the value of $K_6 = \mathrm{KO}_{42}$ ($K_5' = \mathrm{KI}_{42}$ is already known) and check the condition
     $$\mathrm{FI}_{K_5'}(\tilde{d}_i \oplus K_6) \oplus \mathrm{FI}_{K_5'}(\tilde{d}_j \oplus K_6) = \tilde{e}_i \oplus \tilde{e}_j = \tilde{f}_i \oplus \tilde{f}_j, \tag{2}$$

where K_5' is known from the guess of KL_6. Again, expect a single guess for K_6 fulfilling this condition.

Any values of K_4 and K_6 that satisfy (1) and (2) must be wrong as they would cause the impossible differential to hold. Use a map of 2^{32} bits – which can be reused for each guess of KL_5, KL_6 – to mark these wrong guesses of (K_4, K_6).

Analysis. First, we determine the work needed for a structure of size 2^m with m to be determined later; note that m is necessarily bounded by $m \leq 32$ due to the block-size of 64 bits. For each C_i and all guesses of KL_5 and KL_6 the decryption through FL^{-1} takes $2^{32}(1+2^{32}) \approx 2^{64}$ computations of FL^{-1} so for the structure about 2^{64+m} computations of FL^{-1} are needed. Checking the conditions on K_4 and K_6 for each matching pair needs work of $2 \cdot 2 \cdot 2^{16}$ computations of FI.

For the structure we expect about $\binom{2^m}{2}/2^{16} \approx 2^{2m-17}$ matching pairs for each guess of the 32 bits in KL_5. Each matching pair is expected to discard a single wrong guess of 32 bits (K_4, K_6) for each guess of the 32 bits of KL_6.

Thus, for the whole structure we expect in total about $2^{2m-17} \cdot 2^{32} \cdot 2^{32} = 2^{2m+47}$ wrong keys of 96 bit to be discarded. Assuming that the wrong keys appear at random with equal probability, finding all wrong keys is the coupon collector's problem [3, 10]. Therefore, with about $2^{96} \ln(2^{96}) \approx 67 \cdot 2^{96}$ keys of 96 bits being discarded we expect only the right key to remain. Thus, $m = 27.5$ yielding a structure of size $2^{27.5}$ is sufficient. As only the right key is expected to remain, the bitmap – which is reused for each guess of KL_5 and KL_6 – is expected to contain only a single unmarked position for the correct guess of KL_5 and KL_6.

The attack needs a single structure of $2^{27.5}$ chosen plaintexts, $2^{64+27.5} = 2^{91.5}$ computations of FL^{-1}, and $2^{32} \cdot 2^{32} \cdot 2^{18} \cdot 2^{2 \cdot 27.5-17} = 2^{120}$ computations of FI, roughly equivalent to 2^{116} encryptions.

As we need to store only the ciphertexts C_i for the structure, a working copy of all $\tilde{C}_i$, and the w_i, the memory consumption can be bounded by $2^{28.5}$ blocks of 64 bits each. In the processing step the map of 2^{32} bits to mark wrong guesses of (K_4, K_6) needs much less memory than the working copy of the ciphertexts.

Remark 1. The reduced number of chosen plaintexts required for this attack in comparison to the attack given in [7, Section 4.2] (which did not include the final FL functions) is due to the fact that here the use of one single structure allows to make efficient use of the plaintexts; this technique can also be applied to [7, Section 4.2] with a significant reduction in the plaintext requirements.

Remark 2. While the chosen plaintext requirements as well as the memory consumption are well in reach of today's attackers, the work factor makes the attack only an academic possibility. But nevertheless, it is much faster than guessing the 128-bit key by brute force.

Experimental results. This attack has been in part verified experimentally. All key words except K_6 (used as KO_{42}) and K_7 (second half of KL_6) were assumed to be known, thus reducing the work factor involved and also reducing

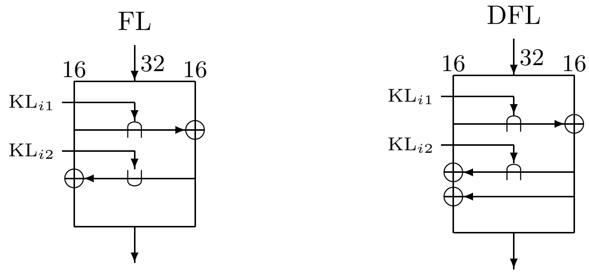

Fig. 2. Structure of FL (left), $\cap$ denotes the bitwise AND operation, $\cup$ the bitwise OR operation. When looking only at differential behavior, the structure on the right results (note the changed operation in the second round).

the map to 2^{16} bits. Due to memory constraints only $N = 5{\cdot}2^{23}$ chosen plaintexts were used; due to time constraints only 511 passes with wrong values of K_7 in addition to the correct value of K_7 have been tested.

Assume now a pass with a fixed guess for K_7. As K_4 is known and thus fixed, we expect only a fraction of 2^{-16} of the pairs to fulfill (1), thus we expect about $M = \frac{N(N-1)}{2}\cdot 2^{-16}\cdot 2^{-16} = 204800$ pairs to have output XOR (β, β) and to fulfill (1); each of these pairs is expected to remove one guess of K_6 from the map. With $r = M/2^{16}$ we expect $2^{16}\exp(-r) \approx 2880$ candidates for K_6 to remain unmarked in the map (see [10, Theorem 4.18] for this instance of the occupancy problem).

The observed mean of removed guesses for K_6 (including collisions) in the experiments was 204759, the mean of remaining candidates for K_6 was 2883, thus matching the theory very accurately. The correct K_6 was still in the map when using the correct value of K_7. For each guess of K_7 about 80 minutes of CPU-time were needed on a PC with Pentium III (800MHz), 256 MBytes of RAM plus 512 MBytes of swap space; about 640 MBytes of memory were used.

4 The Slicing Attack on 4-Round MISTY1

In this section a new kind of attack is presented that makes essential use of presence, position and structure of the key-dependent FL functions. This attack bypasses the components of the cipher that provide the provable security against differential and linear cryptanalysis.

4.1 Differential Properties of the FL Function

The FL function is a linear (or affine) function for any fixed key. It consists of a 2-round Feistel network with the round function being a bitwise AND resp. OR operation with the key bits (see Figure 2).

Notation. Denote bit i of a value a by $a[i]$ counting the bits from LSB to MSB starting with 0.

As only bitwise operations without any shifts or other means of diffusion are involved, FL basically consists of 16 parallel versions of a cipher with a 2-bit block. Let (a, b) be the input and (c, d) be the output of FL with 16-bit values a, b, c, d. Then block i consists of the bits $(a[i], b[i])$ and $(c[i], d[i])$.

In the following the round functions are analysed algebraically in order to obtain a closed description for the differential behavior of FL. Let k denote a key bit and x an input. Then the round functions are as follows:

$$\begin{aligned} x \cap k &:= xk \\ x \cup k &:= x \oplus k \oplus xk. \end{aligned}$$

Now let x^* denote a second input and let $x' = x \oplus x^*$; then the differential behavior of these operations is as follows:

$$\begin{aligned} (x \cap k) \oplus (x^* \cap k) &= xk \oplus x^*k = x'k \\ (x \cup k) \oplus (x^* \cup k) &= (x \oplus k \oplus xk) \oplus (x^* \oplus k \oplus x^*k) = x' \oplus x'k. \end{aligned}$$

Therefore, for differences the FL function has the effective description given on the right hand side of Figure 2. Call this function DFL.

4.2 Slicing 4-Round MISTY1

The attack in the previous section employed the 3-round impossible differential $(0, \alpha) \not\rightarrow (0, \beta)$. Any sub-key for the last round (including the FL functions that follow) that yields the output difference of the impossible differential must be wrong. This is used to discard all the wrong keys.

Another view on this situation is as follows. It focuses on the changes to the nonzero difference in the half of the data that is the right half of the input. This is shown in Figure 3. The input difference in the right half is $\alpha \neq 0$, causing an nonzero output difference α' of the FL function in round 1. The first round's FO has a zero output difference, so no further change occurs here. The difference is modified again in round 3 by an FL function ($\alpha'' \neq 0$) and by XOR with the output difference γ of FO, yielding δ'. Finally, it is modified through the output transformation by FL yielding a difference δ in the left half of the ciphertext. This is shown in the right part of Figure 3.

The output difference γ of FO in round 3 must be nonzero, as can be seen as follows. The input difference of the FO in round 2 is $\alpha' \neq 0$, so $\beta' \neq 0$, as FO is bijective. Therefore, in round 3, the input to FO is also nonzero, thus causing $\gamma \neq 0$.

It should be noted that the difference of concern here – right half of plaintext difference, left half of ciphertext difference – is changed only by the keys to FL with the single exception of the XOR with the difference γ in round 3.

Therefore, any set of keys $(\mathrm{KL}_2, \mathrm{KL}_4, \mathrm{KL}_6)$ to a stack of three instances of DFL that yields δ from α implies that $\gamma = 0$, thus it must be wrong. The

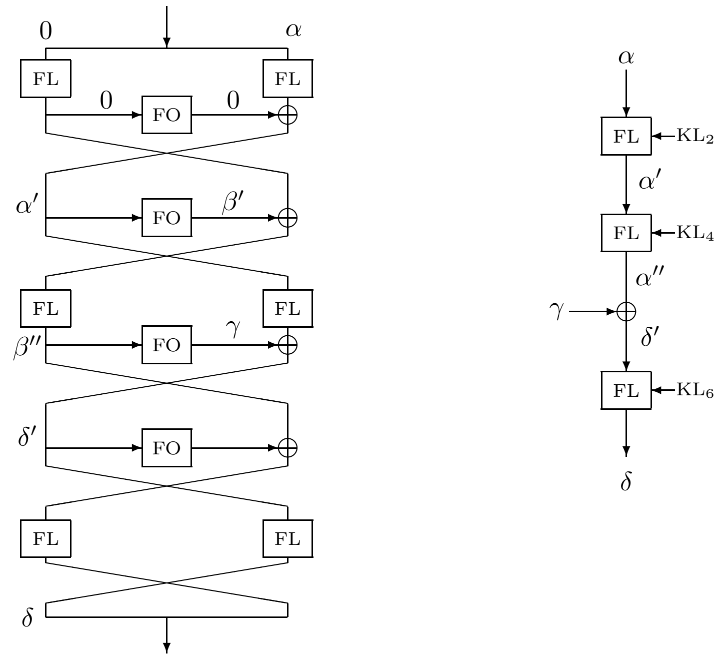

Fig. 3. Slicing MISTY1. The differential path of the data from the right half of the input to the left half of the output is shown on the right side. The difference γ is known to be nonzero.

result is that we are dealing only with a slice of 4-round MISTY1. Note that this property does not hold for the underlying Feistel network without the key-dependent linear functions and that an extension to more rounds seems not to work.

Definition 1. *The slice of three instances of* DFL *consists of 16 parallel instances of the same key-dependent function. Denote it by F, indexed by the 6-bit key k, i.e. $F_k : \{0,1\}^2 \rightarrow \{0,1\}^2$. The blocks are located in the same places as for* FL*.*

In the following some properties of F are shown that will subsequently be used for the attack.

Lemma 1. *Depending on the key, F realises one of six different bijective functions. Thus F has six classes of equivalent keys. There are four classes with 11 keys and two with 10 keys.*

Proof. From the structure of F it is clear that F is a bijective function, therefore $F_k(a) \neq 0$ for nonzero a. As the input and output of F are differences, it follows that $F_k(0) = 0$ for any k. On the remaining three inputs F realises a permutation, of which there are $3! = 6$ different. Checking all 64 possible keys gives the number of keys per class showing that all of these functions are indeed realised. $\square$

Notation. Let $\mathcal{K}_1, \ldots, \mathcal{K}_6$ denote the six classes of equivalent keys for F and $\mathcal{F}_i$ denote the function realised by any of the keys in $\mathcal{K}_i$ for $i \in \{1, \ldots, 6\}$.

Proposition 1. *For any nonzero a, b there are exactly two $i \in \{1, \ldots, 6\}$ such that $\mathcal{F}_i(a) = b$ holds.*

4.3 Attacking the Slice

As a consequence of these classes of equivalent keys it should be clear that the best one can hope for is to find a vector of 16 functions that never implies the output difference of the third round's FO being zero. Further conditions to distinguish right vs. wrong keys must come from the key scheduling or other means besides the slicing attack (see Section 4.4).

Definition 2. *Let $\alpha = (\alpha_l, \alpha_r)$ be the input to the slice of three* DFL *functions and $\delta = (\delta_l, \delta_r)$ its output. As both α and δ come from plaintext resp. ciphertext differences, still call this a* pair *and denote it as $\alpha \to \delta$. A vector $(f_{15}, \ldots, f_0) \in \{\mathcal{F}_1, \ldots, \mathcal{F}_6\}^{16}$ is called* valid *for $\alpha \to \delta$ if for each $i \in \{0, \ldots, 15\}$ the 2-bit block $(\alpha_l[i], \alpha_r[i])$ is mapped to $(\delta_l[i], \delta_r[i])$ by f_i.*

As one cannot distinguish between functions with a zero input and output, any pair that causes a zero input / output to any of the 16 parallel instances of F cannot be used. Such a pair is called a *bad pair* whereas a pair with only nonzero input and output blocks for each of the 16 instances of F is called a *good pair*.

From Proposition 1 it follows that each good pair has 2^{16} valid vectors of functions while there are $6^{16} \approx 2^{41.4}$ vectors in total. With $2^{41.4}/2^{16} = 2^{25.4}$ good pairs there are $2^{41.4}$ valid vectors. Assuming that the valid vectors appear at random with equal probability, this is the coupon collector's problem [3, 10] Therefore, with about $2^{41.4}\ln(2^{41.4}) \approx 2^{46.2}$ valid vectors from about $2^{30.2}$ good pairs all valid vectors are expected to be found. As it is known that an invalid vector exists, this one is expected be be singled out.

The chance that a random pair is a good pair is about $(9/16)^{16} \approx 2^{-13.3}$. Therefore about $2^{43.5}$ pairs are needed which can be gained from about $2^{22.25}$ chosen plaintexts.

Now we are ready to state the actual Slicing Attack:

1. *Data Collection.* Build a structure of $2^{22.25}$ plaintexts $P_i = (r, s, t_i, u_i)$ with constant (r, s) and (t_i, u_i) being arbitrary but all different; obtain the ciphertexts $C_i = (v_i, w_i, x_i, y_i)$ encrypted under the unknown key.

2. *Filtering.* For each pair (i, j), $i < j$, check if $\alpha = (\alpha_l, \alpha_r) = (t_i \oplus t_j, u_i \oplus u_j)$, $\delta = (\delta_l, \delta_r) = (v_i \oplus v_j, w_i \oplus w_j)$ form a good pair, i.e. $(\alpha_l[m], \alpha_r[m]) \neq 0$, $(\delta_l[m], \delta_r[m]) \neq 0$ for all $0 \leq m \leq 15$. For all good pairs store (α, δ) in a table T.
3. *Processing, Outer Loop.* For each of the 6^6 assignments of $(f_5, \ldots, f_0)$ do the following. First, select all those good pairs $\alpha \to \delta$ such that $(f_5, \ldots, f_0)$ is valid for the corresponding six blocks; store the selected good pairs in a table T'.
   Initialise a bit map B of $6^{10} < 2^{26}$ bits, then execute the inner loop:
   - *Processing, Inner Loop.* For all good pairs in T' set the bits in B that correspond to the valid vectors $(f_{15}, \ldots, f_6)$ for the rightmost 10 blocks. Finding these can be done by using a preprocessed table to get the possibilities for each of the 10 blocks.

   After all pairs in T' are processed check which bits in B are still cleared. These correspond to possibly invalid vectors.

Analysis. The filtering is expected to keep $\binom{2^{22.25}}{2}/2^{13.3} \approx 2^{30.2}$ good pairs, thus from the discussion above it is clear that the algorithm is expected to single out one invalid vector $(f_{15}, \ldots, f_0)$.

As by Proposition 1 exactly 2 out of 6 functions are valid for each given input and output of a 2 bit block the chance that a good pair in T is included in T' is about $\left(\frac{1}{3}\right)^6 \approx 2^{-9.5}$. Therefore T' is expected to have a size of $2^{30.2}/2^{9.5} < 2^{21}$.

The total running time consists of three components: first, the time for filtering, second, the time for constructing table T' in step 3, and, third, the time spent in the inner loop.

The filtering takes $2^{43.5}$ checks to find the good pairs. The building of T' is done 6^6 times where each time about $2^{30.2}$ checks have to be done (a check can be done with look-up tables, taking only constant time). This step thus takes a total of roughly 2^{46} checks.

Each execution of the inner loop sets about $2^{10} \cdot 2^{21} = 2^{31}$ bits, so the total time spent here in all iterations of step 3 is roughly $6^6 \cdot 2^{31} \approx 2^{47}$ elementary operations like computing indices plus setting bits in the bitmap etc.

This sums up to running time roughly equivalent to 2^{45} encryptions. The memory consumption can be bounded by the size of the plaintexts and ciphertexts, the tables T and T', and the bitmap in the inner loop, totaling to about $2^{31.2}$ blocks.

Remark 3. While the slicing attack does not directly reveal key bits, it gains knowledge about the class of equivalent keys of the real key $(\mathrm{KL}_2, \mathrm{KL}_4, \mathrm{KL}_6)$. This class contains at most $11^{16} \approx 2^{55.4}$ keys. Comparing this to the initial set of 2^{96} keys shows a gain in knowledge of about 40 bits.

Remark 4. When also considering the key scheduling, the real key is (K'_3, K_5), (K'_4, K_6), (K'_5, K_7) with $K'_5 = \mathrm{FI}_{K_6}(K_5)$, so that the real entropy is only 80 bits. But for the keys in the equivalence class the same 16-bit condition holds, so that about 2^{40} keys are expected to remain. This is also a gain in knowledge about the key of about 40 bits.

4.4 Finding the Real Key Bits

When using the knowledge gained in the slicing attack in a subsequent step of analysis the work factor of the slicing attack is only involved as additive work to what follows.

A simple way is brute force, namely enumeration of the about 2^{40} keys in the equivalence class and guessing the remaining 48 key-bits, requiring expected $\frac{1}{2}2^{88}$ encryptions, about 2 known plaintexts / ciphertexts and de facto no memory.

Better methods are given below. One uses impossible differentials in the usual chosen plaintext model of attack, the other uses the chosen plaintext/ciphertext model to efficiently find the complete key.

Improving the impossible differential attack. The differential attack of Section 3 can be improved significantly by using the information from the slicing attack; this is faster than the brute force method at the cost of more chosen plaintexts. It makes use of the fact that only 16 key-bits (K_1' in KL_5) have to be guessed in addition to enumerating the about 2^{40} keys in the equivalence class.

The attack proceeds as follows after having knowledge about the correct equivalence class, again using Property 1 of FO and the 3-round impossible differential $(0, \alpha) \not\to (0, \beta)$ (see Section 3).

1. *Data Collection.* Build a structure of 2^m plaintexts $P_i = (x, y, a_i, b_i)$ with constant x, y and random but different (a_i, b_i) and obtain the corresponding ciphertexts $C_i = (c_i, d_i, e_i, f_i)$. The number m is determined later in the analysis to be $m = 27.2$.
2. *Enumerate the Keys.* The keys in the equivalence class can be enumerated by stepping through all 2^{32} assignments of K_5' and K_6; then set $K_5 = \mathrm{FI}^{-1}_{K_6}(K_5')$ and enumerate all possible assignments of K_3', K_4', and K_7 by considering separately each 2-bit block using a precomputed table. For each assignment compute $K_4 = \mathrm{FI}^{-1}_{K_5}(K_4')$ and $K_3 = \mathrm{FI}^{-1}_{K_4}(K_3')$.
   For each 16-bit value for K_1' set $\mathrm{KL}_5 = (K_3, K_1')$, $\mathrm{KL}_6 = (K_5', K_7)$ and do the following step:
   (a) *Find wrong keys.* For all ciphertexts C_i compute $(\tilde{e}_i, \tilde{f}_i) = \mathrm{FL}^{-1}_{\mathrm{KL}_5}(e_i, f_i)$, $(\tilde{c}_i, \tilde{d}_i) = \mathrm{FL}^{-1}_{\mathrm{KL}_6}(c_i, d_i)$, and build two lists

$$
\begin{aligned}
u_i &= \tilde{e}_i \oplus \tilde{f}_i \\
w_i &= (\mathrm{FI}_{K_1'}(K_4 \oplus \tilde{c}_i) \oplus \tilde{d}_i, \mathrm{FI}_{K_5'}(K_6 \oplus \tilde{d}_i) \oplus \tilde{e}_i).
\end{aligned}
$$

   If for any i, j there is a match $u_i = u_j$ and $w_i = w_j$, go to the next guess of the keys, otherwise keep the guessed keys. The rest of the key bits, i.e. K_2 and K_8, can be found by brute force and inverting the key schedule (to find K_1).

Analysis. The outer loop for the enumeration of the keys in the equivalence class has 2^{32} iterations with a single application of FI^{-1} taking place. For each assignment of K_5' and K_6 about 2^8 values are expected to be found for

(K'_3, K'_4, K_7). While for some functions and fixed key-bits no suitable keys exist, this is not a problem because these events can be found efficiently with the precomputed table.

The costs here are 2^{32} computations of FI^{-1} and some table operations which is much less than the enumeration of all about 2^{40} keys. For each of these keys two FI^{-1} computations take place; in total this is about $2^{32} + 2 \cdot 2^{40} \approx 2^{41}$ computations of FI^{-1}, independent of the size 2^m of the structure.

It is easy to see that a pair (i, j) with $u_i = u_j$ yields a symmetric difference (β, β). Each pair (i, j) with $u_i = u_j$ and additionally $w_i = w_j$ fulfills the two conditions

$$\mathrm{FI}_{K'_1}(K_4 \oplus \tilde{c}_i) \oplus \mathrm{FI}_{K'_1}(K_4 \oplus \tilde{c}_j) = \tilde{d}_i \oplus \tilde{d}_j \quad \text{(first round of FO)}$$
$$\mathrm{FI}_{K'_5}(K_6 \oplus \tilde{d}_i) \oplus \mathrm{FI}_{K'_5}(K_6 \oplus \tilde{d}_j) = \tilde{e}_i \oplus \tilde{e}_j \quad \text{(second round of FO)},$$

thus fulfilling Property 1 of FO. Therefore this guess of the key must be wrong and is discarded; a correct key never fulfills these conditions. This ensures the correctness of the algorithm.

Per guessed key about $2 \cdot 2^m$ applications of FL^{-1} and the same number of applications of FI are done, for all key guesses (about 2^{40} from the equivalence class times 2^{16} from K'_1) in total about 2^{57+m} applications of FL^{-1} and 2^{57+m} applications of FI.

For each pair (i, j) there is a chance of about 2^{-16} to fulfill $u_i = u_j$ and a chance of about 2^{-32} to fulfill $w_i = w_j$, thus a chance of 2^{-48} to discard a key of 56 bits. Modeling the keys discarded by each pair as random and assuming an equal probability, we expect about $2^{56}2^{-48} = 2^8$ keys being discarded by any pair. The task of discarding all wrong keys is the coupon collector's problem [3, 10]. Therefore, with about $2^{56} \ln(2^{56}) \approx 2^{61.3}$ keys discarded by about $2^{61.3}/2^8 = 2^{53.3}$ pairs only the right key is expected to remain. This implies a choice of $m = 27.2$ and a structure of $2^{27.2}$ chosen plaintexts.

The work needed sums up to 2^{41} applications of FI^{-1}, $2 \cdot 2^{83.2}$ applications of FL^{-1}, and $2^{84.2}$ applications of FI. This is roughly equivalent to about $2^{81.6}$ encryptions.

The memory consumption can be bounded by the number of ciphertexts, a working copy for decryptions by FL^{-1}, and the tables for the u_i and w_i. This sums to roughly $2^{29.2}$ blocks which is less than needed for the slicing attack.

Attack in the chosen plaintext / ciphertext model. In this model the slicing attack can also be used to find an equivalence class for the sub-keys $\mathrm{KL}_1 = (K_1, K'_7)$, $\mathrm{KL}_3 = (K_2, K'_8)$, and $\mathrm{KL}_5 = (K_3, K'_1)$ with chosen ciphertext queries; call this the *backward slice* and denote its equivalence class by $\mathcal{K}_b$ in contrast to the *forward slice* with class $\mathcal{K}_f$ of the chosen plaintext attack.

This preprocessing steps together take $2 \cdot 2^{45}$ work, $2^{22.25}$ chosen plaintexts queries, $2^{22.25}$ chosen ciphertext queries and $2^{31.2}$ blocks of memory. Note that adaptiveness of the queries is not necessary here.

Now we use the fact that K_3 in KL_5 can be computed from K'_3, K'_4, and K_5 from the forward slice; a similar property holds for K_7. This can be used with the meet-in-the-middle technique parametrised by $0 \leq N \leq 16$ to allow a time/memory tradeoff:

1. *Global Loop.* Step through all values for the highest N bits of K_6.
   (a) *Enumerate forward slice.* Step through all values of the lower $16-N$ bits of K_6 and the 16 bits of K'_5; compute $K_5 = \mathrm{FI}^{-1}_{K_6}(K'_5)$. Enumerate all values for K'_3, K'_4, K_7 that are in $\mathcal{K}_f$ for the fixed sub-key values using a precomputed table. Compute $K_4 = \mathrm{FI}^{-1}_{K_5}(K'_4)$, and $K_3 = \mathrm{FI}^{-1}_{K_4}(K'_3)$. Store the 128 bits K_3, K'_3, K_4, K'_4, K_5, K'_5, K_6, K_7 in a hash table T indexed by (K_3, K_7) allowing later to retrieve all entries with the same index.
   (b) *Enumerate backward slice.* For all values of K'_1 and K_2 compute $K_1 = \mathrm{FI}^{-1}_{K_2}(K'_1)$ and enumerate the values for K'_8, K'_7, and K_3 in $\mathcal{K}_b$, again using a precomputed table. For each of these also compute $K_8 = \mathrm{FI}^{-1}_{K_1}(K'_8)$, $K_7 = \mathrm{FI}^{-1}_{K_8}(K'_7)$.
      i. *Check the keys.* Retrieve all entries with the same (K_3, K_7) from the hash table T. Complete the key scheduling and do one or if necessary two trial encryptions to check whether it is the correct key.

Analysis. First look at the steps (a) and (b) that are executed inside the global loop. Step (a) is expected to enumerate about $2^{40}/2^N = 2^{40-N}$ sub-key values while performing about $2^{16-N} \cdot 2^{16} + 2 \cdot 2^{40-N}$ computations of FI^{-1} (like in the analysis above there might be values for K'_5 such that no valid sub-keys are found, but the total work to find these is much less than the enumeration of all the 2^{40-N} sub-keys). The expected size of T is 2^{40-N} values à 128 bits which is 2^{41-N} blocks.

Step (b) is enumerating the about 2^{40} keys in $\mathcal{K}_b$ with about $2^{32} + 2 \cdot 2^{40}$ computations of FI^{-1}. In T we expect to find $2^{40} \cdot 2^{40-N} \cdot 2^{-32} = 2^{48-N}$ matches, therefore the completion of the key schedule and the trial encryption is expected to be done about 2^{48-N} times.

In total, taking the global loop into account, the time needed is about $2^{41-N+N} + 2^{41+N} = 2^{41} + 2^{41+N}$ computations of FI^{-1} roughly equivalent to about $2^{39} + 2^{39+N}$ encryptions. The full cipher is expected to be run about $2^N \cdot 2^{48-N} = 2^{48}$ times.

With $N = 10$ we can efficiently reuse the memory used in the slicing attack, needing about $1.5 \cdot 2^{49}$ work and 2^{31} memory, With $N = 8$ the time is dominated by the 2^{48} trial encryptions with a memory requirement of 2^{33} blocks.

4.5 MISTY variants and the Slicing Attack

As the slicing attack allows to attack the 4-round version of MISTY1 very efficiently, one might ask whether this attack applies also to the MISTY1 variant KASUMI [4] which is used in 3rd generation cellular phones. In comparison to MISTY1 the FO and FI functions are modified and, more important here, the FL functions – with bit-rotations added in the round function – are moved to be part of KASUMI's round function. As KASUMI is a plain 8-round Feistel network with no key-dependent operations being performed outside the round function, the slicing attack does not apply.

In MISTY1 the slicing attack is possible because of the position of the FL functions; avoiding this requires to move the FL functions. The attack is also efficient because it is easy to determine all keys resp. vectors of parallel functions in the slice that map an input XOR to an output XOR. To prevent the slicing attack it would thus be necessary to add a new design criterion besides those given in [9]. A possible fix might be adding bit-rotations to FL's round functions (like in KASUMI's FL) to avoid the parallelism, but whether this prevents the attack is left to future research. On the other hand the slicing attack seems to work only for the 4-round version of MISTY1, thus using more than 4 rounds should prevent this attack.

5 Conclusion

While for the impossible differential attack on the 4-round version of MISTY1 presented in this paper the chosen plaintext requirements and the memory consumption are certainly in range of today's attackers, the high work factor involved does not threaten the cipher.

On the other hand the slicing attack is made possible by the position and structure of the FL functions. It shows that augmenting the Feistel network with key-dependent functions can introduce subtle weaknesses that are not present in the Feistel network itself; one special feature is that the slicing attack completely bypasses the components that provide the provable security of the cipher. Furthermore, this is surprisingly efficient, it is clearly in range of today's possibilities.

While the MISTY1 proposal allows any multiple of four as the number of rounds, the results in this paper show that the 4-round version should be avoided, thus leaving the recommended number of 8 rounds as a minimum.

The author would like to thank David Wagner for helpful discussions and for suggesting the use of the chosen plaintext / ciphertext model. Thanks are also due to the anonymous referees of the 2nd NESSIE workshop and FSE 2002 whose comments helped to improve the paper.

References

[1] E. Biham, A. Biryukov, and A. Shamir. Miss in the middle attacks on IDEA and Khufu. In L. Knudsen, editor, *Fast Software Encryption, 6th international Workshop*, volume 1636 of *Lecture Notes in Computer Science*, pages 124–138, Rome, Italy, 1999. Springer-Verlag.

[2] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer Verlag, Berlin, 1993.

[3] K. L. Chung. *Elementary Probability Theory with Stochastic Processes*. Springer Verlag, 1979.

[4] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms – Document 2: KASUMI Specification, Version 1.0. 3G TS 35.202, December 23, 1999. `http://www.etsi.org/dvbandca/3GPP/3GPPconditions.html`.

[5] L. R. Knudsen. DEAL — A 128-bit block cipher. Technical Report 151, Department of Informatics, University of Bergen, Bergen, Norway, Feb. 1998.

[6] L. R. Knudsen and D. Wagner. Integral cryptanalysis. These Proceedings, pages 114–129.

[7] U. Kühn. Cryptanalysis of Reduced-Round MISTY. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 325–339. Springer Verlag, 2001.

[8] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, pages 386–397, Berlin, 1993. Springer-Verlag. Lecture Notes in Computer Science Volume 765.

[9] M. Matsui. New block encryption algorithm MISTY. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68, Haifa, Israel, 20–22 Jan. 1997. Springer-Verlag.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[11] NESSIE. New European Schemes for Signature, Integrity, and Encryption. `http://www.cryptonessie.org`.

[12] H. Tanaka, K. Hisamatsu, and T. Kaneko. Strength of MISTY1 without FL function for higher order differential attack. In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Proc. Applied algebra, algebraic algorithms, and error-correcting codes: 13th international symposium, AAECC-13*, volume 1719 of *Lecture Notes in Computer Science*, pages 221–230, Hawaii, USA, 1999. Springer Verlag.