Improved Multi-Dimensional Meet-in-the-Middle Cryptanalysis of KATAN

Shahram Rasoolzadeh and Håvard Raddum

Simula Research Laboratory

Abstract. We study multidimensional meet-in-the-middle attacks on the KATAN block cipher family. Several improvements to the basic attacks are introduced and explained. The most noteworthy of these is the technique of guessing only non-linearly involved key bits, which reduces the search space by a significant factor. The optimizations decreases the complexity of multidimensional meet-in-the-middle attacks, allowing more rounds of KATAN to be efficiently attacked than previously reported.

Keywords: lightweight cipher, KATAN, meet-in-the-middle, reducing complexity

1 Introduction

The KATAN family of block ciphers was designed by Cannière et al., presented at CHES 2009 [1], and has become a well-known instance of a lightweight block cipher. All versions of the KATAN family have a structure based on non-linear feedback shift registers (NLFSR). The simple round function, iterated a large number of times, allows an efficient hardware implementation and simultaneously meets the security requirements one would expect from this cipher.

KATAN has received a substantial amount of cryptanalysis since it was presented. Previous work on KATAN in the single key setting includes algebraic and cube attacks [2], conditional differential [3], differential [5], all subkeys recovery (ASR) MITM [4, 6], match-box MITM [7], multidimensional (MD) MITM [8] and dynamic cube [9]. These are all summarized in Table 1, where we also include the work of this paper.

We focus on the multidimensional (MD) meet-in-the-middle (MITM) attacks on KATAN that were introduced in [8]. The main contribution of our work is reduction in attack complexity through several optimizations. In particular, we introduce a technique of removing key bits that are not used in any non-linear operations, and show that under some mild conditions it is not necessary to guess these bits in a MITM attack.

The paper is organized as follows: Section 2 gives a brief description of KATAN, and Section 3 describes MD MITM attacks with focus on KATAN. In Section 4 we explain the optimizations we can do, and report on the improved results in Section 5. Section 6 concludes the paper.

Table 1. Summary Result of Single-Key Attacks on KATAN Family

Version	Type	Round	Time	Data	Memory	Ref.
KATAN32	Cube	60	2^{39}	$2^{30.3}$ CP	-	[2]
	Cond. Diff.	78	2^{22}	2^{22} CP	-	[3]
	Algebraic	79	14.7 min	20 CP	-	[2]
	MITM ASR	110	2^{77}	138 KP	$2^{75.1}$	[4]
	Differential	114	2^{77}	$2^{31.9}$ KP	-	[5]
	MITM ASR	119	$2^{79.1}$	144 CP	$2^{79.1}$	[6]
	Matchbox MITM	153	$2^{78.5}$	2^{5} CP	2^{76}	[7]
	Dynamic Cube	154	$2^{78.5}$	2^{32}	2^{32}	[9]
	MD MITM	175	$2^{78.3}$	3 KP	$2^{79.6}$	[8]
	MD MITM	**201**	**$2^{78.1}$**	**3 KP**	**$2^{78.1}$**	**Tab. 5**
	MD MITM	**206**	**$2^{79.0}$**	**3 KP**	**$2^{78.1}$**	**Tab. 5**
KATAN48	Cube	40	2^{49}	$2^{25.9}$ CP	-	[2]
	Algebraic	64	6.4 hour	5 CP	-	[2]
	Cond. Diff.	70	2^{34}	2^{34} CP	-	[3]
	MITM ASR	100	2^{78}	128 KP	2^{78}	[4]
	MITM ASR	105	$2^{79.1}$	144 CP	$2^{79.1}$	[6]
	Matchbox MITM	129	$2^{78.5}$	2^{5} CP	2^{76}	[7]
	MD MITM	130	$2^{79.5}$	2 KP	2^{79}	[8]
	MD MITM	**146**	**$2^{78.1}$**	**2 KP**	**2^{77}**	**Tab. 3**
	MD MITM	**148**	**$2^{79.0}$**	**2 KP**	**2^{77}**	**Tab. 3**
KATAN64	Cube	30	2^{35}	$2^{20.7}$ CP	-	[2]
	Algebraic	60	3.2 hour	5 CP	-	[2]
	Cond. Diff.	68	2^{35}	2^{35} CP	-	[3]
	MITM ASR	94	$2^{77.7}$	116 KP	$2^{77.7}$	[4]
	MITM ASR	99	$2^{79.1}$	142 CP	$2^{79.1}$	[6]
	MD MITM	112	$2^{79.5}$	2 KP	2^{79}	[8]
	Matchbox MITM	119	$2^{78.5}$	2^{5} CP	2^{74}	[7]
	MD MITM	**126**	**$2^{78.1}$**	**2 KP**	**2^{77}**	**Tab. 4**
	MD MITM	**129**	**$2^{79.0}$**	**2 KP**	**2^{77}**	**Tab. 4**

2 Description of KATAN

KATAN is a NLFSR-based family of block ciphers with block sizes of 32, 48 and 64 bits. These will be referred to as KATAN32, KATAN48 and KATAN64, respectively. All three versions have 254 rounds and use the same LFSR-type key schedule, accepting an 80-bit user-selected key.

The plaintext is loaded into two registers $L1$ and $L2$. The least significant bit (LSB) of each register, numbered 0, is the rightmost one, and the LSB of the plaintext is loaded into the LSB of $L2$ while its most significant bit (MSB) is loaded into the MSB of $L1$. To update the state registers, $L1$ and $L2$ are shifted to the left by one bit, where the newly computed bits generated according to the following equations, are loaded into the LSBs of $L2$ and $L1$.

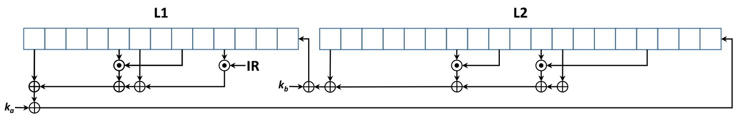

Fig. 1. Structure of KATAN32

Table 2. Parameters of KATAN family

size	$\lvert L1\rvert$	$\lvert L2\rvert$	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5	y_6
32	13	19	12	7	8	5	3	18	7	12	10	8	3
48	19	29	18	12	15	7	6	28	19	21	13	15	6
64	25	39	24	15	20	11	9	38	25	33	21	14	9

$$\begin{cases} f_a(L1) = L1[x_1] \oplus L1[x_2] \oplus (L1[x_3] \cdot L1[x_4]) \oplus (L1[x_5] \cdot IR) \oplus k_a \\ f_b(L2) = L2[y_1] \oplus L2[y_2] \oplus (L2[y_3] \cdot L2[y_4]) \oplus (L2[y_5] \cdot L2[y_6]) \oplus k_b \end{cases} \quad (1)$$

Here $\oplus$ and $\cdot$ are bitwise *xor* and *and* operations, $L[x]$ denotes the x-th bit of L, IR is a round-dependent constant, and k_a and k_b are two subkey bits. For round i, $0 \leq i \leq 253$, k_a and k_b are equal to sk_{2i} and sk_{2i+1}, respectively which are generated by the key schedule. The structure of KATAN32 is shown in Figure 1.

Applying this update procedure (i.e. shifting and loading) once is denoted as a *step* in this paper. One round consists of one, two or three steps with the same subkey bits, for KATAN32, KATAN48 and KATAN64, respectively. After 254 rounds the content of the registers is the ciphertext. Table 2 shows the parameters and tap positions associated with the different versions of KATAN.

The key schedule of KATAN is a linear key schedule based on an 80-bit LFSR defined by the polynomial $x_{80} + x_{61} + x_{50} + x_{13} + 1$. This LFSR generates $2 \times 254 = 508$ subkey bits according to the following rule, where the k_i's are the user-selected key bits:

$$sk_i = \begin{cases} k_i & 0 \leq i < 80 \\ sk_{i-80} \oplus sk_{i-61} \oplus sk_{i-50} \oplus sk_{i-13} & 80 \leq i < 508 \end{cases} \quad (2)$$

3 Meet-in-the-Middle Attacks

In this section we will briefly recall meet-in-the-middle attacks, and introduce the necessary notions for basic MITM and multidimensional MITM attacks. Throughout the paper, the notation $|x|$ means the bit-size of x.

3.1 Basic MITM

The basic MITM attack is a generic technique presented by Diffie and Hellman to cryptanalyse DES [10]. Despite the fact that this technique is arguably less common than differential or linear attacks on ciphers, there are some applications to specific block ciphers (such as KATAN) where using MITM principles are more successful than differential and linear attacks.

Let $E_{i,j}(k_f, S)$ denote the partial encryption of the cipher block S, beginning from the start of round i and ending at the start of round j, where k_f is a particular sequence of subkeys corresponding to these $j - i$ rounds. Similarly, let $D_{j,i}(k_b, S)$ denote the partial decryption of S, beginning from the start of round j and ending at the start of round i, where k_b is the sequence of subkeys corresponding to these $j - i$ rounds. Let K_f and K_b be the total set of subkey sequences that k_f and k_b, respectively, can be drawn from.

Assume we have a cipher with R rounds. The main idea of a MITM attack is that the partial keys in both parts of the cipher can be guessed separately. First, the attacker goes through all values of k_f, computes $E_{0,r}(k_f, P)$ for a plaintext P and save the cipher states in a table. The part of the cipher from round 0 to r is called the *forward* side. Next, the attacker goes through all values of k_b and computes $D_{R,r}(k_b, C)$ for the corresponding ciphertext. The rounds from r to R are called the *backward* side. If

$$E_{0,r}(k_f, P) = D_{R,r}(k_b, C), \tag{3}$$

then k_f and k_b are candidates for representing the correct secret key, and can be tested on other known plaintext/ciphertext pairs. We sum up the basic MITM attack in the following algorithm.

Algorithm 1 Basic MITM attack

```
for $k_f \in K_f$ do
    Compute $v = E_{0,r}(k_f, P)$;
    Store $k_f$ in table $\mathcal{T}$ indexed by $v$;
end for
for $k_b \in K_b$ do
    Compute $v' = D_{R,r}(k_b, C)$;
    Find the corresponding $k_f$ in $\mathcal{T}[v']$ if it exists (then (3) holds for $(k_f, k_b)$);
    Check the candidate $(k_f, k_b)$ on a few other known plaintext/ciphertext pairs;
    if $(k_f, k_b)$ fits all plaintext/ciphertext pairs then
        Return $(k_f, k_b)$ as the correct key;
    end if
end for
```

3.2 Multidimensional MITM Attack

Multidimensional MITM attacks were first introduced in 2011 by Zhu and Gong in [8]. In a two-dimensional (2D) MITM attack we guess a cipher block state S somewhere in the middle of the cipher, and assume for the rest of the attack that the value of S is known. This breaks the encryption in two parts, called *dimensions* in the context of MITM attacks. The idea is to combine two basic MITM attacks for each dimension, where S serves as a "ciphertext" in the first dimension and a "plaintext" in the second dimension.

Note that the total complexity of a 2D MITM attack is the complexity of guessing S multiplied with some part of the complexity of the two MITM attacks. To have a MD MITM attack that is faster than exhaustive key search it is therefore necessary that the key size is significantly larger than the block size. In the case of the KATAN family, the key size is 80 bits and the block sizes are 32, 48 or 64 bits.

In principle, the basic MITM attack can be extended into any number of dimensions. For a d-dimensional MITM attack we guess $d-1$ cipher states in the encryption procedure, dividing the cipher into d dimensions. Then we perform d basic MITM attacks in parallel in each dimension. To be faster than exhaustive search it is a necessary condition that $(d-1) \times |S| < |K|$. Hence we can implement 2D MITM attack to all 3 versions of KATAN and also 3D MITM attack to KATAN32.

MD MITM matching when $|\boldsymbol{v}| < |\boldsymbol{K_f}|$ In the basic MITM attack the values to match are given as the value of the cipher block $v = E_{0,r}(k_f, P)$ at some point in the middle of the cipher. When $|v|$ is smaller than $|K_f|$ there will be several k_f-values giving the same cipher state v. As we will see, this is the case for KATAN. When matching from the backward side we could get a large number of (k_f, k_b) candidates to test.

In the first $d-1$ dimensions of d-dimensional MITM attack we would prefer to have a unique k_f-value in each cell in the table built from the forward side, so we get at most one (k_f, k_b) candidate to test for each k_b-value in the backward side. We can achieve this by extending v to (v, u), where u is some value that can be computed both from k_f and from k_b. If $|u|$ is large enough, the (v, u)-value will uniquely identify k_f in the forward side, and k_b in the backward side. Exactly how u is computed is not very important, the important thing is that u is big enough to give uniqueness, and that it can be computed from both k_f and k_b independently.

3.3 Implementing MD MITM on KATAN

We will use 2D MITM and 3D MITM attacks to cryptanalyze the KATAN block ciphers. The 2D or 3D MITM attacks have the same structures, and we summarize the attacks by giving their algorithms. Algorithm 2 for 2D MITM is shown and explained below. Algorithm 3 for 3D MITM attack is similar, and given in Appendix A.

In the algorithms we use $r_1, \ldots, r_5$ as the specific rounds where either MITM matching or the guessing of a full state takes place. The numbers r_2 and r_4 denote the states that break the full cipher into dimensions, and r_1, r_3 and r_5 are the rounds where MITM matchings take place in the first, second and (in case of 3D) third dimensions, respectively. The value of the state separating dimensions 1 and 2 is denoted by s, and the value separating dimensions 2 and 3 is denoted by t.

The matching in the first dimension will always determine a candidate for the full key, so the matchings in the other dimensions will just work like a filter to eliminate wrong key candidates. For this reason we do not need to do matchings in the value of the full state in dimensions 2 and 3, only a few bits are needed to filter out a large fraction of key candidates from dimension 1. The number of bits to match in the other dimension(s) is denoted by n.

Algorithm 2 2D MITM attack to KATAN32/48/64

```
for $k_{f1} \in K_{f1}$ do
    Compute full state of $v_1 = E_{0,r_1}(k_{f1}, P)$ for a plaintext $P$;
    Compute $u$ from $k_{f1}$;
    Store $k_{f1}$ into table $\mathcal{T}_1$ indexed by $v_1$ and $u$;
end for
for $k_{b2} \in K_{b2}$ do
    Compute the $n$ least significant bits of $D_{R,r_3}(k_{b2}, C)$ denoted by $v_2$, for the ciphertext $C$ corresponding to $P$;
    Store $v_2$ in table $\mathcal{T}_2$ indexed by $k_{b2}$;
end for
for $s \in \mathbb{F}_2^{|S|}$ do
    for $k_{f2} \in K_{f2}$ do
        Compute the $n$ least significant bits of $E_{r_2,r_3}(k_{f2}, s)$ denoted by $v'_2$;
        Store $v'_2$ in table $\mathcal{T}'_2$ indexed by $k_{f2}$;
    end for
    for $k_{b1} \in K_{b1}$ do
        Compute full state of $v'_1 = D_{r_2,r_1}(k_{b1}, s)$;
        Compute $u'$ from $k_{b1}$;
        Find $k_{f1} = \mathcal{T}_1[v'_1, u']$;
        Compute values of $k_{f2}$ and $k_{b2}$ using both $k_{f1}$ and $k_{b1}$;
        if $\mathcal{T}'_2[k_{f2}] = \mathcal{T}_2[k_{b2}]$ then
            Test the candidate key using this pair and a few other pairs of known plaintext/ciphertext pairs;
            if candidate key passes all tests then
                Return candidate as correct key;
            end if
        end if
    end for
end for
```

3.4 Complexity

We evaluate the time and memory complexities of Algorithm 2.

The time complexity comes from all the partial encryptions we need to do for all values of partial keys and all values of the guessed state in round r_2. We give the time complexity in terms of full R-round encryptions. The first for-loop in Algorithm 2 iterates $2^{|K_{f1}|}$ times, executing a r_1/R encryption every time. Similarly, the second for-loop costs $2^{|K_{b2}|} \times (R - r_3)/R$ encryptions. The complexity for the other two for-loops can be computed similarly, but these must be done once for every guess of the state S in round r_2, and hence must be multiplied with $2^{|S|}$. Finally, the matching $\mathcal{T}_2'[k_{f2}] = \mathcal{T}_2[k_{b2}]$ succeeds with probability 2^{-n}, which will invoke a full trial encryption on a second pair of plaintext/ciphertext. This matching is done $2^{|s|+|k_{b1}|}$ times, so the time complexity for the full 2D MITM attack can be given as

$$\frac{r_1}{R}2^{|K_{f1}|} + \frac{R - r_3}{R}2^{|K_{b2}|} + 2^{|S|}\left(\frac{r_2 - r_1}{R}2^{|K_{f2}|} + \frac{r_3 - r_2}{R}2^{|K_{b1}|}\right) + 2^{|s|+|k_{b1}|-n}$$

R-round KATAN encryptions.

The memory complexity comes from storing the tables $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_2'$. The table $\mathcal{T}_1$ has $2^{|K_{f1}|}$ cells, each holding one k_{f1} key. The storage needed for $\mathcal{T}_1$ is then $|k_{f1}|2^{|K_{f1}|}$ bits. The tables $\mathcal{T}_2$ and $\mathcal{T}_2'$ has $2^{|k_{b2}|}$ and $2^{|k_{f2}|}$ cells respectively, where each cell holds n bits. The total memory complexity for Algorithm 2 is then be given as

$$|k_{f1}|2^{|K_{f1}|} + n(2^{|k_{f2}|} + 2^{|k_{b2}|})$$

bits.

4 Reducing the Complexity of MITM Attacks

It is possible to apply several optimizations to the MITM attacks we have described that helps speeding them up. Some of them are rather obvious or has little impact on the total complexity, and will only be briefly mentioned in Section 4.2. However, the first optimization described in Section 4.1 can make an attack faster by a high factor. The technique is general and may be applicable to other ciphers than KATAN.

4.1 Only Guessing Non-Linearly Involved Key Bits

During a MITM attack we need to compute $v = E_{0,r}(k_f, P)$, for some guess of k_f. It may happen (as in the case of KATAN) that v only depends linearly on some of the bits in k_f. Let these bits of k_f be $k_{f,l}$, and let the bits that are involved non-linearly in computing v be $k_{f,n}$. By the notion "v depends linearly on $k_{f,l}$" we mean that v can be written as $v = v' \oplus l_f$ where v' can be computed only from $k_{f,n}$, and l_f is a bit-string that can be computed as a linear transformation of $k_{f,l}$. Another way of looking at it is that $v' = E'_{0,r}(k_{f,n}, P)$, where E' is the same as E, except that we simply ignore, or set to 0, the $k_{f,l}$ bits.

We can do the same from the backward side and split k_b into disjoint subsets $k_{b,l}$ and $k_{b,n}$, such that $k_{b,l}$ is only involved linearly when computing $D_{R,r}(k_b, C)$. Similarly to the forward side, we can define $D'_{R,r}(k_{b,n}, C)$ to be equal to D, except that we ignore the $k_{b,l}$ bits that go into the computation of D. Finally, if $v = D_{R,r}(k_b, C)$ and $v' = D'_{R,r}(k_{b,n}, C)$ we set $v = v' \oplus l_b$ where l_b is some linear transformation of $k_{b,l}$.

Let l_f be given as $l_f = k_{f,l}M_f$ where M_f is a $|k_{f,l}| \times |v|$ matrix, and similarly define $l_b = k_{b,l}M_b$. Then (3) can be written as

$$E'_{0,r}(k_{f,n}, P) \oplus k_{f,l}M_f = D'_{R,r}(k_{b,n}, C) \oplus k_{b,l}M_b \tag{4}$$

Assume that the key schedule of the cipher we are investigating is linear, and that $k_{f,n}$ and $k_{b,n}$ together uniquely define the user selected key. Then it is possible to find two matrices N_f and N_b such that $k_{f,l} = (k_{f,n}, k_{b,n})N_f$ and $k_{b,l} = (k_{f,n}, k_{b,n})N_b$. Moreover, we can split N_f and N_b into their top and bottom parts N_{*1} and N_{*2} such that

$$\begin{cases} k_{f,l} = (k_{f,n}, k_{b,n})N_f = k_{f,n}N_{f1} \oplus k_{b,n}N_{f2} \\ k_{b,l} = (k_{f,n}, k_{b,n})N_b = k_{f,n}N_{b1} \oplus k_{b,n}N_{b2} \end{cases} \tag{5}$$

Inserting this into (4), rearranging terms and setting $A_f = N_{f1}M_f \oplus N_{b1}M_b$ and $A_b = N_{b2}M_b \oplus N_{f2}M_f$ we get

$$E'_{0,r}(k_{f,n}, P) \oplus k_{f,n}A_f = D'_{R,r}(k_{b,n}, C) \oplus k_{b,n}A_b \tag{6}$$

The crucial observation is that $k_{f,l}$ and $k_{b,l}$ have disappeared from the equation used for matching, hence we do not need to guess these bits when computing matching values for round r. Only bits that are involved in non-linear operations need to be guessed. Also, we have rearranged the terms such that the left hand side can be computed only from $k_{f,n}$ and the right hand side only from $k_{b,n}$.

When the assumptions given here hold, we can use one side of (6) as the values to put in the table for the MITM matchings. This technique may speed up the attack by a factor $2^{|k_{f,l}|}$ and $2^{|k_{b,l}|}$ in the forward and backward sides, allowing us to cryptanalyse more rounds of the cipher.

KATAN48 Example To clarify the technique, we will give the details of MITM matching for the first dimension of a 2D MITM attack to KATAN48. Assume that we know the whole 48-bit value of S_{60}, and want to do a MITM matching in all 48 bits of S_{42}. In the forward side, for computing $E'_{0,42}$ from the plaintext, instead of guessing all the subkey bits going into these rounds we only need to guess the 77 non-linearly involved key bits. These bits are

$$k_{f1,n} = \{k_0, \ldots, k_{76}\}.$$

The other key bits used in this 42-round encryption are only involved linearly, so we set $k_{f1,l}$ to be

$$k_{f1,l} = \{k_{77}, \ldots, sk_{83}\}.$$

The bits $sk_{80}, \ldots, sk_{83}$ can be computed from $k_{f1,n}$, so they would not need to be guessed in any case, even if we did not use the technique described in this section. The speed-up in reduced complexity comes only from not having to guess k_{77}, k_{78}, k_{79}, and the complexity in the forward side will therefore be reduced by a factor 2^3.

In the backward side, instead of guessing all 36 subkey bits of $\{sk_{84}, \ldots, sk_{119}\}$ for computing $D'_{60,42}$ from S_{60}, we only need to guess the 32 non-linearly involved subkey bits

$$k_{b1,n} = \{sk_{86}, sk_{88}, sk_{90}, \ldots, sk_{119}\}.$$

The other 4 bits only gets *xor*ed to the state bits, so

$$k_{b1,l} = \{sk_{84}, sk_{85}, sk_{87}, sk_{89}\}.$$

None of the $k_{b1,l}$ bits can be computed from $k_{b1,n}$, so the complexity in the backward side gets reduced by a factor 2^4.

In this example, M_f will be a 7×48 matrix and M_b will be a 4×48 matrix. The matrices N_{f1} and N_{f2} will be of sizes 77×7 and 32×7, respectively, and N_{b1} and N_{b2} of sizes 77×4 and 32×4. We have computed these matrices and verified the correctness of the technique described in this section. The actual matrices for this example can be found in Appendix B.

4.2 Other Optimizations

To maximize the number of rounds that can be attacked we briefly mention a few other optimizations that should be used in a MITM attack on KATAN.

First, we notice that after the plaintext has been loaded into the registers, in the first few steps the key bits are only *xor*ed onto the state and are not involved in the non-linear operations. We can therefore clock the registers up until the first key-bits go into a non-linear operation, and treat the state at this point as the (modified) plaintext. The first operation of the cipher is then to add the first key bits onto their correct cells, before clocking continues as normal. This will save us a few f_a and f_b steps in the beginning.

Secondly, if we do a MD MITM attack and only do partial matching in a dimension, some of the f_a and f_b operations may not affect the bits used for matching. This applies to f_a and f_b in the rounds close to the state used for matching. Of course, we do not need to actually compute these instances of f_a and f_b and so save a little in the complexity.

Third, when running through all values values of $k_{f,n}$ or $k_{b,n}$ to compute matching values, we do not need to start computing from the beginning of E' or D' for each guess. Instead, for the first guess we will store the states after each round up until the matching state. For the second guess, we will only backtrack to the previous state, change the value of one of the key bits there, and recompute one round from the stored state to get a new key guess and a new matching value. In general, we will only backtrack to a state where we can make an untried key guess, and compute forward from the state stored there.

In KATAN we must guess two key bits in each round, and do *st* steps. If we compute r rounds, and naively start from the beginning for each key guess, we must compute $4^r \times r \times st$ steps to try all keys. When storing all intermediate states and only do minimal backtracking, the expression for the number of steps we must compute to run through all keys is

$$4(st + 4(st + 4(st + \dots 4(st)\dots), \tag{7}$$

where there are r brackets. This expression can be written as $4st\sum_{i=0}^{r-1} 4^i$. Using the equality $\sum_{i=0}^{r-1} x^i = \frac{x^r-1}{x-1}$ we can simplify (7) to

$$4(st + 4(st + 4(st + \dots 4(st)\dots) = 4st\frac{4^r - 1}{3} \approx 4^r\frac{4}{3}st \tag{8}$$

The speed-up of this technique compared to starting from the beginning for each guess is approximately a factor $3r/4$. In addition to guessing only non-linearly involved key bits, this acceleration contributes the most in reducing the attack complexity.

All optimizations described in this section have been used to get the final complexities for the MITM attacks on KATAN given in Section 5.

5 Results on KATAN

We have applied the optimization techniques from Section 4 to MD MITM attacks on KATAN. The optimizations allow us to do the attacks in [8] with a lower complexity, and to push the attacks for more rounds before meeting the exhaustive search limit. In this section we give the details of the cryptanalysis giving the best attacks on each of KATAN32/48/64.

5.1 2D MITM on KATAN48

In all our 2D MITM attacks to KATAN48, we guess S_{60} to break the cipher into two dimensions. In the first dimension, P and S_{60} meet each other in S_{42} and for this meeting we guess the 77 key bits $k_{f1,n} = \{k_0, \dots, k_{76}\}$ in the forward side and the 32 subkey bits $k_{b1,n} = \{sk_{86}, sk_{88}, sk_{90}, \dots, sk_{119}\}$ in the backward side. The used value for u that is computable from both $k_{f1,n}$ and $k_{b1,n}$ is equal to

$$\begin{gathered} sk_{86}, sk_{88}, sk_{93}, \dots, sk_{102}, sk_{106}, \dots, sk_{115}, sk_{119}, sk_{90} \oplus sk_{103}, \\ sk_{90} \oplus sk_{116}, sk_{91} \oplus sk_{104}, sk_{91} \oplus sk_{117}, sk_{92} \oplus sk_{105}, sk_{92} \oplus sk_{118} \end{gathered}$$

We can adjust the values of R, r_3 and n to tune the complexity of the attack. The details for some of these values in the second dimension giving the best attacks are given in Table 3, where T indicates time complexity and M indicates memory complexity.

Table 3. Results for 2D MITM attack to KATAN48

R	r_3	$\|k_{f2,n}\|$	$k_{f2,n}$ indexes	$\|k_{b2,n}\|$	$k_{b2,n}$ indexes	n	T	M
148	85	31	$120, \ldots, 144, 146, 147, 148, 153, 154, 161$	77	$219, \ldots, 295$	1	$2^{79.04}$	$2^{76.96}$
146	84	31	$120, \ldots, 146, 150, 151, 152, 159$	78	$213, 215 \ldots, 291$	2	$2^{78.09}$	$2^{77.02}$
145	83	32	$120, \ldots, 145, 147, 148, 149, 150, 155, 157$	78	$211, \ldots, 289$	3	$2^{77.20}$	$2^{77.05}$
143	82	31	$120, \ldots, 148, 151, 153, 155$	78	$207, 209, \ldots, 285$	5	$2^{75.69}$	$2^{77.12}$
142	82	32	$120, \ldots, 148, 151, 153, 155$	78	$205, 207, \ldots, 283$	6	$2^{75.16}$	$2^{77.15}$
141	81	30	$120, \ldots, 147, 149, 151, 153$	78	$203, 205, \ldots, 281$	7	$2^{74.60}$	$2^{77.19}$
139	80	29	$120, \ldots, 145, 147, 149, 151$	77	$199, 201, 203, \ldots, 277$	7	$2^{74.24}$	$2^{77.07}$

5.2 2D MITM on KATAN64

We break the cipher into two dimensions by guessing S_{51}. In the first dimension, P and S_{51} meet each other in round 42. For this matching we guess the 77 key bits $k_{f1,n} = \{k_0, \ldots, k_{76}\}$ in the forward side and the 16 subkey bits $k_{b1,n} = \{sk_{86}, \ldots, sk_{101}\}$ in the backward side. The used value for u is equal to

$$sk_{86}, \ldots, sk_{89}, sk_{93}, \ldots, sk_{101}$$

Again we can adjust the values of R, r_3 and n to tune the complexity of the attack. The details for some of these values in the second dimension giving the best attacks are given in Table 4.

In the second dimension there is an extra twist. As all 64 bits of S_{51} do not participate in the value of the n matching bits in r_3, we do not need to guess all of the 64 state bits. Algorithm 2 should then be modified so only part of s is guessed in the for-loop $s \in \mathbb{F}_2^S$, and the for-loop $k_{b1} \in K_{b1}$ is changed to also include the remaining bits of s. This modification allows us to guess more subkey bits in $k_{f2,n}$.

5.3 3D MITM on KATAN32

For KATAN 32, we found that a 3D MITM attack gives the highest number of rounds that can be attacked. As in the 2D MITM attacks above, for the 3D MITM attacks to KATAN32 we have the same details for full-state matching dimensions. We guess S_{73} and S_{97} to break the cipher into three dimensions.

Table 4. Results for 2D MITM attack to KATAN64

R	r_3	$\|s\|$	$\|k_{f2,n}\|$	$k_{f2,n}$ indexes	$\|k_{b2,n}\|$	$k_{b2,n}$ indexes	n	T	M
129	69	59	20	$102, \ldots, 116, 118, 120, 122, 123, 129$	78	$179, 181, \ldots, 257$	1	$2^{79.03}$	$2^{76.98}$
127	67	60	16	$102, \ldots, 112, 114, 116, 118, 119, 125$	78	$175, 177, \ldots, 253$	2	$2^{78.06}$	$2^{77.02}$
126	67	64	16	$102, \ldots, 112, 114, 116, 118, 119, 125$	78	$173, 175, \ldots, 251$	3	$2^{77.14}$	$2^{77.05}$
124	65	64	14	$102, \ldots, 108, 110, 112, \ldots, 115, 119, 121$	78	$169, 171, \ldots, 247$	5	$2^{75.46}$	$2^{77.12}$
123	65	64	14	$102, \ldots, 108, 110, 112, \ldots, 115, 119, 121$	78	$167, 169, \ldots, 245$	6	$2^{74.81}$	$2^{77.15}$
123	65	64	16	$102, \ldots, 108, 110, \ldots, 115, 117, 119, 121$	78	$167, 169, \ldots, 245$	7	$2^{74.45}$	$2^{77.19}$

Table 5. Results for 3D MITM attack to KATAN32

R	r_5	$\|k_{f3,n}\|$	$k_{f3,n}$ indexes	$\|k_{b3,n}\|$	$k_{b3,n}$ indexes	n	T	M
206	134	48	194, . . . , 233, 235, 236, 238, 240, 242, 246, 249, 255	78	317, 321, 333, 335, 337, 338, 339, 341, . . . , 411	1	$2^{79.03}$	$2^{78.05}$
201	130	47	194, . . . , 234, 236, 238, 239, 241, 242, 245, 247	78	307, 323, 325, 327, . . . , 401	2	$2^{78.05}$	$2^{78.07}$
199	129	48	194, . . . , 232, 234, . . . , 239, 241, 243, 245	78	303, 319, 321, 323, . . . , 397	3	$2^{77.13}$	$2^{78.22}$
197	128	46	194, . . . , 228, 230, . . . , 235, 237, 238, 239, 241, 243	78	313, 315, 317, 319, . . . , 393	4	$2^{76.17}$	$2^{78.10}$
196	128	47	194, . . . , 235, 237, 238, 239, 241, 243	78	297, 313, 315, 317, . . . , 391	5	$2^{75.37}$	$2^{78.22}$
195	128	47	194, . . . , 235, 237, 238, 239, 241, 243	78	295, 311, 312, 315, . . . , 389	6	$2^{74.67}$	$2^{78.26}$
194	128	47	194, . . . , 235, 237, 238, 239, 241, 243	78	293, 309, 311, 313, . . . , 387	7	$2^{74.13}$	$2^{78.31}$

In the first dimension, P and S_{73} meet each other in S_{44}. For this matching we guess the 78 key bits $k_{f1,n} = \{k_0, \ldots, k_{76}, k_{78}\}$ in the forward side and the 48 subkey bits $k_{b1,n} = \{sk_{96}, sk_{98}, sk_{100}, \ldots, sk_{145}\}$ in the backward side. The used value for u that is computable from both $k_{f1,n}$ and $k_{b1,n}$ is equal to

$$\begin{gathered} sk_{96}, sk_{98}, sk_{100}, sk_{101}, sk_{102}, sk_{104}, sk_{106}, \ldots, sk_{115}, sk_{117}, sk_{119}, \ldots, sk_{126}, \\ sk_{128}, sk_{130}, sk_{132}, \ldots, sk_{137}, sk_{139}, sk_{141}, sk_{143}, sk_{145}, sk_{103} \oplus sk_{116}, \\ sk_{103} \oplus sk_{127}, sk_{103} \oplus sk_{138}, sk_{103} \oplus sk_{142}, sk_{105} \oplus sk_{118}, sk_{105} \oplus sk_{131}, \\ sk_{105} \oplus sk_{140}, sk_{105} \oplus sk_{144}, sk_{103} \oplus sk_{105} \oplus sk_{129} \end{gathered}$$

In the second dimension, S_{73} and S_{97} meet each other in S_{89}, and for this matching we guess 32 bits $k_{f2} = \{sk_{146}, \ldots, sk_{177}\}$ in the forward side and the 16 bits $k_{b2} = \{sk_{178}, \ldots, sk_{193}\}$ in the backward side. Varying R and r_5, the details of the attack are given in Table 5.

6 Conclusions

In this work we have introduced several optimizations for doing a MD MITM attack. Applying these techniques on KATAN allows efficient attacks on more rounds than published previously, before reaching the exhaustive search bound. The tables in Section 5 gives an idea of how the time and memory complexities scale when adjusting the attack parameters. The conclusion is that there exist valid attacks on 206-round KATAN32, 148-round KATAN48 and 129-round KATAN64.

The technique of only guessing non-linearly involved key bits in a MITM attack may be of independent interest besides the application to KATAN. There may be other ciphers where the technique can be applied, reducing the security margin. Investigating optimizations to MITM attacks on other ciphers may be a topic for further research.

References

1. C. De Cannière, O. Dunkelman, and M. Knezevic, "KATAN and KTANTAN - a Family of Small and Efficient Hardware-oriented Block Ciphers", *CHES 2009*, LNCS, vol. 5747, pp. 272-288, Springer, 2009.
2. G. V. Bard, N. T. Courtois, J. Nakahara, P. Sepehrdad, and B. Zhang, "Algebraic, Aida/Cube and Side Channel Analysis of KATAN Family of Block Ciphers", *INDOCRYPT 2010*, LNCS, vol. 6498, pp. 176-196, Springer, 2010.
3. S. Knellwolf, W. Meier, M. Naya-Plasencia, "Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems", *ASIACRYPT 2010*, LNCS, vol. 6477, pp. 130–145, Springer, 2010.
4. T. Isobe and K. Shibutani, "All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach", *SAC 2012*, LNCS, vol. 7707, pp. 202-221. Springer, 2012.
5. M. R. Albrecht and G. Leander, "An All-in-One Approach to Differential Cryptanalysis for Small Block Ciphers", *SAC 2012*, LNCS, vol. 7707, pp. 1-15. Springer, 2012.
6. T. Isobe and K. Shibutani, "Improved All-Subkeys Recovery Attacks on FOX, KATAN and SHACAL-2 Block Ciphers", *FSE 2014*, LNCS, vol. 8540, pp. 104-126, Springer, 2015.
7. T. Fuhr and B. Minaud, "Match Box Meet-in-the-Middle Attack Against KATAN", *FSE 2014*, LNCS, vol. 8540, pp. 61-81, Springer, 2015.
8. B. Zhu and G. Gong, "Multidimensional Meet-in-the-Middle Attack and Its Applications to KATAN32/48/64", *Cryptography and Communications*, vol. 6, pp. 313-333, Springer 2014.
9. Z. Ahmadian, Sh. Rasoolzadeh, M. Salmasizadeh and M. R. Aref, "Automated Dynamic Cube Attack on Block Ciphers: Cryptanalysis of SIMON and KATAN", Cryptology ePrint Archive, report 2015/040, 2015.
10. W. Diffie, and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *IEEE Computer Society Press*, vol. 10(6), pp. 74-84, 1977.

A Algorithm for 3D MITM attack

A.1 Complexity

The complexity of Algorithm 3 can be computed similarly to the complexity of Algorithm 2, except that we must guess two full internal states, s and t. The time complexity can be given as

$$\tfrac{r_1}{R}2^{|k_{f1}|} + \tfrac{R-r_5}{R}2^{|k_{b3}|} + \tfrac{r_5-r_4}{R}2^{|t|+|k_{f3}|} +$$

$$2^{|s|}\left(\tfrac{r_2-r_1}{R}2^{|k_{b1}|} + \tfrac{r_3-r_2}{R}2^{|k_{f2}|} + \tfrac{r_4-r_3}{R}2^{|t|+|k_{b2}|}\right) + 2^{|s|+|t|+|k_{b2}|-n}$$

R-round encryptions of KATAN32. The memory complexity for storing the tables are

$$|k_{f1}|2^{|k_{f1}|} + n(2^{|k_{b3}|} + 2^{|t|+|k_{f3}|}) + 80 \cdot 2^{|k_{b1}|} + |k_{f2}|2^{|k_{f2}|}$$

bits.

Algorithm 3 3D MITM attack to KATAN32

```
for $k_{f1} \in K_{f1}$ do
    Compute $v_1 = E_{0,r_1}(k_{f1}, P)$ for a plaintext $P$;
    Compute $u$;
    Store $k_{f1}$ in a table $\mathcal{T}_1$ indexed by $v_1$ and $u$;
end for
for $k_{b3} \in K_{b3}$ do
    Compute $v_3$, the $n$ least significant bits of $D_{R,r_5}(k_{b3}, C)$ for the ciphertext $C$ corresponding to $P$;
    Store $v_3$ in a table $\mathcal{T}_3$ indexed by $k_{b3}$;
end for
for $t \in \mathbb{F}_2^{32}$ and $k_{f3} \in K_{f3}$ do
    Compute $v'_3$, the $n$ least significant bits of $E_{r_4,r_5}(k_{f3}, t)$;
    Store $v'_3$ into a table $\mathcal{T}'_3$ indexed by $(t, k_{f3})$;
end for
for $s \in \mathbb{F}_2^{32}$ do
    for $k_{b1} \in K_{b1}$ do
        Compute $v'_1 = D_{r_2,r_1}(k_{b1}, s)$;
        Compute $u'$;
        Find $k_{f1} = \mathcal{T}_1[(v'_1, u')]$;
        Compute 80-bit master key candidate $K$ from $k_{f1}$ and $k_{b1}$;
        Compute $(k_{f2}, k_{b2})$ from $K$;
        Store $K$ in table $\mathcal{T}'_1$ indexed by $(k_{f2}, k_{b2})$;
    end for
    for $k_{f2} \in K_{f2}$ do
        Compute $v_2 = E_{r_2,r_3}(k_{f2}, s)$;
        Store $k_{f2}$ in table $\mathcal{T}_2$ indexed by $v_2$;
    end for
    for $t \in \mathbb{F}_2^{32}$ and $k_{b2} \in K_{b2}$ do
        Compute $v'_2 = D_{r_4,r_3}(k_{b2}, t)$;
        Find $k_{f2} = \mathcal{T}_2[v'_2]$;
        Find $K = \mathcal{T}'_1[(k_{f2}, k_{b2})]$;
        Calculate values of $k_{f3}$ and $k_{b3}$ from $K$;
        if $\mathcal{T}'_3[t, k_{f3}] = \mathcal{T}_3[k_{b3}]$ then
            Test the candidate key using this pair and a few other pairs of known plaintext/ciphertext;
            if candidate key matches the plaintext/ciphertext pairs then
                Return candidate key as the correct key
            end if
        end if
    end for
end for
```

B Matrices for KATAN48 Example

The values for M_f and M_b are given below, where $O_{r \times c}$ denotes an all-zero matrix of size of $r \times c$.

$$M_f = \left[\begin{array}{cccccc|c|cccccccc|c}
1&0&0&0&0&0& & 0&0&0&0&0&0&1&1& \\
0&0&0&0&1&1& & 0&0&0&0&0&0&0&0& \\
0&0&0&0&0&0& & 0&0&0&0&1&1&0&0& \\
0&0&1&1&0&0& O_{7\times 23} & 0&0&0&0&0&0&0&0& O_{7\times 11} \\
0&0&0&0&0&0& & 0&0&1&1&0&0&0&0& \\
1&1&0&0&0&0& & 0&0&0&0&0&0&0&0& \\
0&0&0&0&0&0& & 1&1&0&0&0&0&0&0&
\end{array}\right],$$

$$M_b = \left[\begin{array}{c|cccccc|c|cc}
 & 0&0&0&0&0&0 & & 1&1 \\
O_{4\times 23} & 0&0&0&0&1&1 & O_{4\times 17} & 0&0 \\
 & 0&0&1&1&0&0 & & 0&0 \\
 & 1&1&0&0&0&0 & & 0&0
\end{array}\right].$$

The matrices N_{f1}, N_{f2} and N_{b1} are equal to:

$$N_{f1} = \left[\begin{array}{ccccccc}
0&0&0&1&0&0&0 \\
0&0&0&0&1&0&0 \\
0&0&0&0&0&1&0 \\
0&0&0&0&0&0&1 \\ \hline
\multicolumn{7}{c}{O_{6\times 7}} \\ \hline
1&0&0&0&0&0&0 \\
0&1&0&0&0&0&0 \\
0&0&1&0&0&0&0 \\
\multicolumn{7}{c}{O_{6\times 7}} \\ \hline
0&0&0&1&0&0&0 \\
0&0&0&0&1&0&0 \\
0&0&0&0&0&1&0 \\
0&0&0&0&0&0&1 \\ \hline
\multicolumn{7}{c}{O_{6\times 7}} \\ \hline
1&0&0&0&0&0&0 \\
0&1&0&1&0&0&0 \\
0&0&1&0&1&0&0 \\
0&0&0&0&0&1&0 \\
0&0&0&0&0&0&1 \\
\multicolumn{7}{c}{O_{6\times 7}} \\ \hline
1&0&0&0&0&0&0 \\
0&1&0&0&0&0&0 \\
0&0&1&0&0&0&0 \\
\multicolumn{7}{c}{O_{34\times 7}}
\end{array}\right], \quad
N_{f2} = \left[\begin{array}{ccccccc}
0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0 \\
1&0&0&0&0&0&0 \\
0&1&0&0&0&0&0 \\
0&0&1&0&0&0&0 \\ \hline
\multicolumn{7}{c}{O_{27\times 7}}
\end{array}\right], \quad
N_{b1} = \left[\begin{array}{cccc}
\multicolumn{4}{c}{O_{4\times 4}} \\ \hline
1&0&0&0 \\
0&0&0&0 \\
0&1&0&0 \\
0&0&1&0 \\
0&0&0&0 \\
0&0&0&1 \\ \hline
\multicolumn{4}{c}{O_{13\times 4}} \\ \hline
1&0&0&0 \\
0&0&0&0 \\
0&1&0&0 \\
0&0&1&0 \\
0&0&0&0 \\
0&0&0&1 \\ \hline
\multicolumn{4}{c}{O_{5\times 4}} \\ \hline
1&0&0&0 \\
0&0&0&0 \\
0&1&0&0 \\
0&0&1&0 \\
0&0&0&0 \\
0&0&0&1 \\ \hline
\multicolumn{4}{c}{O_{31\times 4}} \\ \hline
1&0&0&0 \\
0&0&0&0 \\
0&1&0&0 \\
0&0&1&0 \\
0&0&0&0 \\
0&0&0&1
\end{array}\right].$$

and N_{b2} is equal to $O_{32\times 4}$.