Multidimensional Meet in the Middle Cryptanalysis of KATAN

Shahram Rasoolzadeh and Håvard Raddum

Simula Research Laboratory

Abstract. KATAN and KTANTAN are two lightweight families of hardware oriented block ciphers proposed by Cannière et al. at CHES 2009. They have different versions of 32-, 48- and 64-bit state, all of which work with an 80-bit key. Inspired by the Trivium stream cipher, these families have an innovative structure based on two non-linear feedback shift registers. Such a structure attracts the attention of cryptanalysts and consequently a variety of security analyses have been published. Although the KTANTAN family is already regarded as a broken cipher, the full-round KATAN family is still secure.

In this paper, by exploiting several properties of the KATAN round function as well as the slow diffusion of key bits, we propose some techniques to extend the number of rounds covered by multidimensional meet in the middle attack on all versions of the KATAN family of block ciphers. Our results show that this method can attack up to 206, 148 and 129 reduced-round versions of KATAN32, KATAN48 and KATAN64, respectively, with only 2 or 3 pairs of known plaintext. This cryptanalysis covers the highest number of rounds to date.

Our work is still far from a full-round attack, so it could not be considered as a threat to this family of block ciphers yet. We state that KATAN is still safe to use.

Keywords: KATAN, Multidimensional Meet in the Middle Attack, Lightweight Block Cipher.

1 Introduction

Lightweight embedded systems such as RFID tags and wireless sensor networks have become increasingly common over the past few years. Standard ciphers such as AES were not primarily designed for use in such a constrained environment, which imposes extreme restrictions on hardware footprint. Designing a secure and lightweight primitive is an interesting topic in cryptography.

In order to find solutions to this ever-increasing demand, lightweight cryptography has been developed as one of the most active areas in symmetric cryptography where a remarkable number of lightweight block ciphers have been proposed in the recent years. These lightweight block ciphers aim to balance security with the resource constraints, which leads to new and creative designs. Thus it is necessary to assess the security of these primitives carefully.

The KATAN and KTANTAN families of block ciphers designed by Cannière et al. and presented at CHES 2009 [1] are well known instances of such cryptographic primitives. All versions of these families have a structure based on two non-linear feedback shift registers (NLFSR). This simple round function, iterated a large number of rounds, allows an efficient hardware implementation and simultaneously meets the security requirements one would expect from this cipher.

In comparison with KATAN, KTANTAN has a weaker key schedule and slower diffusion of key bits. Hence, some full-round attacks on the whole KTANTAN family have been presented using a meet in the middle (MITM) technique [13,14,15]. In the case of KATAN, despite of a variety of reduced-round cryptanalyses in different models, including single key [12,2,4,5,6,8,9,10,11], related key [3,7] and physical [12,16,17,18] attacks; this cipher has remained secure and seems to have enough security margin.

Previous works on KATAN in the single key setting include algebraic and cube attacks [12], conditional differential [2], differential [6], all subkeys recovery (ASR) MITM [5,8], match-box MITM [9], multidimensional (MD) MITM [10] and dynamic cube [11], which are all summarized in Table 1. The attack which reaches the highest number of rounds on all three versions is MD MITM attack presented by Zhu and Gong that is able to cryptanalyse 175, 130 and 112 rounds of KATAN32, KATAN48 and KATAN64, respectively.

In this paper, by exploiting several properties of the round function and the slow key diffusion, we propose some new techniques to significantly increase the number of rounds attacked by the multidimensional meet in the middle attack on the KATAN family of block ciphers. Our results shows that this method can successfully attack up to 206, 148 and 129 rounds of KATAN32, KATAN48 and KATAN64, respectively, with only 2 or 3 plaintext/ciphertext pairs of known data. The proposed attacks achieve the highest number of rounds ever analyzed. The associated parameters are reported in Table 1.

This paper is organized as follows. Section 2 introduces the MITM and MD MITM attacks briefly and presents our technique for guessing subkey bits. Section 3 presents a brief description of KATAN and also a method to do faster partial encryptions or decryptions. In Section 4 we outline the key recovery attack on KATAN block ciphers using MD MITM cryptanalysis with all details and their complexities. Finally, Section 5 concludes the paper.

2 MITM and MD MITM Attacks

In this section we briefly introduce the basic MITM and MD MITM attacks and show when these attacks are better than exhaustive key search. The method of implementing basic MITM and 2D MITM attacks is discussed, and at the end of the section a technique that reduces the number of subkey bits needed to be guessed is presented.

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	-	[12]
	Conditional Differential	78	2^{22}	2^{22} CP	-	[2]
	Algebraic	79	14.7 min	20 CP	-	[12]
	MITM ASR	110	2^{77}	138 KP	$2^{75.1}$	[5]
	Differential	114	2^{77}	$2^{31.9}$ KP	-	[6]
	MITM ASR	119	$2^{79.1}$	144 CP	$2^{79.1}$	[8]
	Matchbox MITM	153	$2^{78.5}$	2^5 CP	2^{76}	[9]
	Dynamic Cube	154	$2^{78.5}$	2^{32}	2^{32}	[11]
	MD MITM	175	$2^{78.3}$	3 KP	$2^{79.6}$	[10]
	MD MITM	**201**	$\mathbf{2^{78}}$	**3 KP**	$\mathbf{2^{78}}$	**Sec.4.4**
	MD MITM	**206**	$\mathbf{2^{79}}$	**3 KP**	$\mathbf{2^{78}}$	**Sec.4.4**
KATAN48	Cube	40	2^{49}	$2^{25.9}$ CP	-	[12]
	Algebraic	64	6.4 hour	5 CP	-	[12]
	Conditional Differential	70	2^{34}	2^{34} CP	-	[2]
	MITM ASR	100	2^{78}	128 KP	2^{78}	[5]
	MITM ASR	105	$2^{79.1}$	144 CP	$2^{79.1}$	[8]
	Matchbox MITM	129	$2^{78.5}$	2^5 CP	2^{76}	[9]
	MD MITM	130	$2^{79.5}$	2 KP	2^{79}	[10]
	MD MITM	**146**	$\mathbf{2^{78.1}}$	**2 KP**	$\mathbf{2^{77}}$	**Sec. 4.2**
	MD MITM	**148**	$\mathbf{2^{79}}$	**2 KP**	$\mathbf{2^{77}}$	**Sec. 4.2**
KATAN64	Cube	30	2^{35}	$2^{20.7}$ CP	-	[12]
	Algebraic	60	3.2 hour	5 CP	-	[12]
	Conditional Differential	68	2^{35}	2^{35} CP	-	[2]
	MITM ASR	94	$2^{77.7}$	116 KP	$2^{77.7}$	[5]
	MITM ASR	99	$2^{79.1}$	142 CP	$2^{79.1}$	[8]
	MD MITM	112	$2^{79.5}$	2 KP	2^{79}	[10]
	Matchbox MITM	119	$2^{78.5}$	2^5 CP	2^{74}	[9]
	MD MITM	**126**	$\mathbf{2^{78.1}}$	**2 KP**	$\mathbf{2^{77}}$	**Sec. 4.3**
	MD MITM	**129**	$\mathbf{2^{79}}$	**2 KP**	$\mathbf{2^{77}}$	**Sec. 4.3**

KP: Known Plaintext CP: Chosen Plaintext

2.1 Basic MITM attack

The basic MITM attack is a generic technique presented by Diffie and Hellman to cryptanalyse DES [19]. Despite the fact that this technique is arguably much less common than differential or linear attacks on ciphers, there are some applications to specific block ciphers (such as KATAN and KTANTAN) where using MITM principles which are more successful than differential and linear attacks.

Let $E_{i,j}(k_f, S)$ denote the partial encryption of the 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.

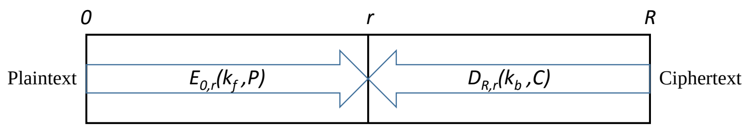

Fig. 1. Basic MITM Attack

The main idea of a MITM attack is that the subkeys in both parts of the cipher can be guessed separately. First, the attacker guesses k_f and computes $E_{0,r}(k_f, P)$ for a plaintext P. Next, he guesses k_b and computes $D_{R,r}(k_b, C)$ for the corresponding ciphertext. If

$$E_{0,r}(k_f, P) = D_{R,r}(k_b, C), \tag{1}$$

then k_f and k_b are candidates for representing the correct secret key.

Figure 1 illustrates the procedure of basic MITM attack and it can be described as follows:

Algorithm 1 Basic MITM attack

```
for $k_f \in K_f$ do
    Compute $v = E_{0,r}(k_f, P)$;
    Store $k_f$ into a 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 (Eq. (1) 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 the plaintext/ciphertext pairs then
        Return $(k_f, k_b)$ as correct key;
    end if
end for
```

2.2 MD MITM attack

A multidimensional MITM attack is an extended version of MITM attack where some internal states are guessed to divide the cipher into smaller sub-ciphers for easier MITM analysis. This extension attack is only available to block ciphers that have a greater key size than block size and usually needs the key size to be significantly larger than the block size.

The first MD MITM attack was presented by Zhu and Gong to cryptanalyse the KATAN family in [10]. In their algorithm, all subkey bits of the sub-ciphers for both forward and backward sides were guessed. We use a more efficient way of guessing subkey bits which lets us attack more rounds of the KATAN family of block ciphers.

The simplest MD MITM is 2D MITM attack where an internal state S is guessed and two MITM attacks are performed on the sub-ciphers divided by S. One MITM attack is done for the plaintext P and the internal state S, finding candidate subkeys k_{f1} and k_{b1}. The other connects the internal state S and the ciphertext C and finds candidate subkeys for k_{f2} and k_{b2}. Figure 2 illustrates the procedure of 2D MITM attack and it can be described algorithmically as follows:

Algorithm 2 2D MITM attack

```
for $k_{f1} \in K_{f1}$ do
    Compute $v_1 = E_{0,r1}(k_{f1}, P)$;
    Store $k_{f1}$ into a table $\mathcal{T}_1$ indexed by $v_1$;
end for
for $k_{b2} \in K_{b2}$ do
    Compute $v_2 = D_{R,r3}(k_{b2}, C)$;
    Store $k_{b2}$ into a table $\mathcal{T}_2$ indexed by $v_2$;
end for
for $s \in S$ do
    for $k_{b1} \in K_{b1}$ do
        Compute $v_1' = D_{r2,r1}(k_{b1}, s)$;
        Find $k_{f1}$ stored in $\mathcal{T}_1[v_1']$;
        Store $(k_{f1}, k_{b1})$ in a table $\mathcal{T}_1'$ indexed by $v_1'$;
    end for
    for $k_{f2} \in K_{f2}$ do
        Compute $v_2' = E_{r2,r3}(k_{f2}, s)$;
        Find $k_{b2}$ stored in $\mathcal{T}_2[v_2']$;
        Store $(k_{f2}, k_{b2})$ in a table $\mathcal{T}_2'$ indexed by $v_2'$;
    end for
    for $(v_1', v_2') \in (V_1, V_2)$ do
        Find $(k_{f1}, k_{b1})$ in $\mathcal{T}_1'[v_1']$ and $(k_{f2}, k_{b2})$ in $\mathcal{T}_2'[v_2']$;
        Check the candidate $(k_{f1}, k_{b1}, k_{f2}, k_{b2})$ on a few other known plain-text/ciphertext pairs;
        if $(k_{f1}, k_{b1}, k_{f2}, k_{b2})$ fits the plaintext/ciphertext pairs then
            Return $(k_f, k_b)$ as correct key;
        end if
    end for
end for
```

For having a valid attack that is faster than brute force of the key, the total number of candidate keys that we test must be less than the number of possible user-selected keys K. This is equal to the simplified condition $|s| + |v_1'| + |v_2'| < |K|$, where $|x|$ is the size of x in bits.

The 3D MITM attack is similar to the 2D MITM one, but with the difference that we must guess two internal states. The procedures of 3D MITM and further MD MITM attacks can be found in [10].

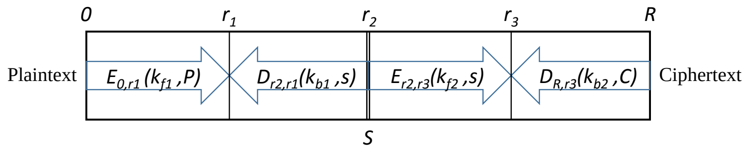

Fig. 2. 2D MITM Attack

2.3 Reducing the number of guessed subkey bits

The KATAN family has a low algebraic degree for its round function and subkey bits are *xor*ed onton the state at each round. This property causes some subkey bits to not get *and*ed with other variables and remain only *xor*ed to the state bits in a partial encryption or decryption. For example, in a simple MITM attack we can write:

$$\begin{cases} E_{0,r}(k_f, P) = E'_{0,r}(k'_f, P) \oplus L_f k''_f, \\ D_{R,r}(k_b, C) = D'_{R,r}(k'_b, C) \oplus L_b k''_b, \end{cases} \tag{2}$$

Here k'_f and k'_b are subsets of k_f and k_b such that $E_{1,r}(k_f, P)$ and $D_{R,r}(k_b, C)$ are nonlinearly dependent on them (i.e. they appear in the input to an *and* operation), and L_f and L_b are binary matrices, only *xor*ing some bits of k''_f or k''_b to the state bits. We need that:

$$\begin{cases} k'_f \cap k''_f = \emptyset \quad , \quad k'_f \cup k''_f = k_f \\ k'_b \cap k''_b = \emptyset \quad , \quad k'_b \cup k''_b = k_b \end{cases} \tag{3}$$

When the key schedule is linear (as is the case for KATAN) and k'_f and k'_b together determine the user-selected key, we can always find two binary matrices L'_f and L'_b of full rank which satisfy the condition

$$L'_f \cdot k'_f \oplus L'_b \cdot k'_b = L_f \cdot k''_f \oplus L_b \cdot k''_b, \tag{4}$$

that is, k''_f and k''_b can be expressed as linear combinations of k'_f and k'_b bits. Instead of checking equation (1) for the whole k_f and k_b, we can then check for

$$E'_{0,r}(k'_f, P) \oplus L'_f \cdot k'_f = D'_{R,r}(k'_b, C) \oplus L'_b \cdot k'_b, \tag{5}$$

where the left and right hand sides can be calculated by k'_f and k'_b in the forward and backward sides of MITM attack, respectively.

This technique enables us to calculate more rounds than that analyzed in [10], by guessing only subkey bits which are involved in a non-linear way to the partial encryption or decryption function. We will use this technique in Section 4, to cryptanalyse the KATAN family of block ciphers.

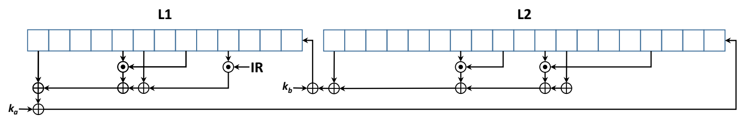

Fig. 3. Structure of KATAN32

3 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 KATAN32, KATAN48 and KATAN64, respectively. All three versions of this family have 254 rounds and use the same LFSR-type key schedule function 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 by 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 LSB of $L2$ and $L1$.

$$\begin{cases} f_a(L1) = L1[x_1] \oplus L1[x_2] \oplus (L1[x_3] \cdot L1[x_4]) \oplus (L1[x_5] \cdot IR) \oplus k_a, \\ f_b(L2) = L2[y_1] \oplus L2[y_2] \oplus (L2[y_3] \cdot L2[y_4]) \oplus (L2[y_5] \cdot L2[y_6]) \oplus k_b, \end{cases} \tag{6}$$

where $\oplus$ and $\cdot$ are bitwise *xor* and *and* operations, respectively, and $L[x]$ 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 Fig 3.

In each round, this update procedure (i.e. shifting and loading) is performed once, twice and three times for KATAN32, KATAN48 and KATAN64, respectively, with the same subkey bits. After 254 rounds the content of the registers is the ciphertext. Table 2 shows all the parameters associated to KATAN32/48/64.

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 is the

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

Table 3. Number of Early Calculated Round Functions in KATAN Block Ciphers

	KATAN32		KATAN48		KATAN64	
	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.
f_a	4	4	7	3	10	4
f_b	4	6	7	7	10	5

user-selected key:

$$sk_i = \begin{cases} k_i & 0 \le i < 80 \\ sk_{i-80} \oplus sk_{i-61} \oplus sk_{i-50} \oplus sk_{i-13} & 80 \le i < 508 \end{cases} \tag{7}$$

3.1 Faster Method for Partial Encryption/Decryption

Here we explain three features of the round functions used in the KATAN family of block ciphers which help us to present a faster method for partial encryption or decryption.

The first feature is that more than one pair of round functions (f_a and f_b) can be calculated before the subkey bits enter the state in a non-linear way. For example, in the case of KATAN32 encryption, by having the state of any round we can calculate 4 pairs of round functions before any subkey bits appear in the input to the *and*-functions. Only then do we need to guess on the value of this key bit, before that it is only **xor**ed to the state and by using the technique in Section 2.3 we do not need to guess them. Table 3 shows the number of round functions which can be calculated before the state depends non-linearly on the subkey bits, in both encryption and decryption mode. We call this the early calculating technique.

The second feature is that as in every round only two subkey bits get inserted into f_a and f_b, it is not necessary to guess all the needed subkey bits for a partial encryption or decryption at first and then do the calculation. Instead we can use a layered method for subkey bits guessing where the round states are calculated until a subkey bit must be guessed. Then we save the last calculated state in the memory and guess the corresponding subkey bit and calculate the next round functions until another subkey bit must be guessed. Calculation in this way will continue until reaching the desired state. When we need to backtrack and try another guess, we only need to go back to a state where we can make a new untried guess, and continue forward from there. In other words, we don't need to calculate all the rounds from the beginning for each new guess.

The third feature relates to only partially matching states, where we only calculate some bits of a desired state. In these situations, there are some round functions which do not participate in the value of target bits in a desired state. These round functions are the ones close to the partial state to match. We can therefore skip calculation of these round functions, and we do not need to guess on any of the key bits that go into them.

In the next section we will see how these three features will help us to do partial encryption or decryption faster than the typical way of guessing all the needed subkey bits and calculating all round functions from the start.

4 MD MITM Cryptanalysis of KATAN

In this section we present 2D MITM attacks on all three versions of the KATAN family with all details and complexities. Also a 3D MITM attack to KATAN32 is presented at the end of the section.

4.1 2D MITM attack on KATAN32

In this subsection we cryptanalyse a 179-round version of KATAN32 using a 2D MITM attack. We break these 179 rounds into two sub-ciphers by guessing S_{73}, the state of round 73. In the first dimension, P and S_{73} meet each other in round 44, i.e., we compute all 32 bits of S_{44} from both P and S_{73}. In the second dimension, S_{73} and C meet each other in the two LSBs $L2[0,1]$ in round 108, i.e. we compute $L2[0,1]$ of S_{108} from both S_{73} and C.

In the forward side of the first dimension, only 78 key bits of

$$k'_{f1} = \{k_0, ..., k_{76}, k_{78}\} \tag{8}$$

are involved non-linearly in the partial encryption from P to S_{44}. In the backward side, only 48 subkey bits of

$$k'_{b1} = \{sk_{96}, sk_{98}, sk_{100}, ..., k_{145}\} \tag{9}$$

are involved non-linearly in the partial decryption from S_{73} to S_{44}. As the union of k'_{f1} and k'_{b1} determines all 80 bits of the master key K, the condition of (4) is established and we can use the technique of (5) for *xor*ing the only linearly involved subkey bits onto S_{44} during partial encryption and decryption.

In the second dimension we only calculate the 2 bits $L2[0,1]$ of S_{108}. In the forward side 48 subkey bits of

$$k'_{f2} = \{sk_{146}, ..., sk_{184}, sk_{186}, sk_{188}, sk_{190}, sk_{192}, sk_{194}, sk_{195}, sk_{197}, sk_{201}, sk_{203}\} \tag{10}$$

are involved non-linearly in the partial encryption of the 2 LSBs of S_{108}. In the backward side, 85 subkey bits of $\{sk_{263}$, sk_{265}, sk_{267}, sk_{269}, sk_{271}, sk_{273}, sk_{275}, sk_{277}, sk_{279}, sk_{281}, $sk_{283}, ..., sk_{357}\}$ are involved non-linearly in the partial decryption. However, only 78 of these are independent so we only need to guess 78 independent subkey bits which are near the ciphertext side:

$$k'_{b2} = \{sk_{263}, sk_{279}, sk_{281}, sk_{283}, ...sk_{357}\} \tag{11}$$

Like in the first dimension, as the union of k'_{f2} and k'_{b2} determines all 80 bits of the master key, we can use the technique of (5) for *xor*ing the only linearly involved subkey bits onto the state S_{108}.

In the forward side of the first dimension, we guess 78 bits of k'_{f1} and calculate 32 bits of S_{44}. So for every value of S_{44}, we can expect 2^{78-32} candidates for k'_{f1} where $S_{44} = E'_{0,44}(k'_{f1}, P) \oplus L'_{f1} \cdot k'_{f1}$. The number of candidate keys that matches any S_{44} is 2^{46}. We then find another 46 bits which is calculable from both forward and backward side of this dimension. For these 46 bits we choose κ as below:

$$\begin{aligned}\kappa = \{&sk_{96}\ , sk_{98}\ , sk_{100}\ , sk_{101}\ , sk_{102}\ , sk_{104}\ , sk_{106}\ , ..., sk_{115}\ , sk_{117}\ , sk_{119}\ , ...,\\ &sk_{126}\ , sk_{128}\ , sk_{130}\ , sk_{132}\ , ..., sk_{137}\ , sk_{139}\ , sk_{141}\ , sk_{143}\ , sk_{145}\ ,\\ &sk_{116} \oplus sk_{103}\ , sk_{127} \oplus sk_{103}\ , sk_{138} \oplus sk_{103}\ , sk_{142} \oplus sk_{103}\ , sk_{118} \oplus sk_{105}\ ,\\ &sk_{131} \oplus sk_{105}\ , sk_{140} \oplus sk_{105}\ , sk_{144} \oplus sk_{105}\ , sk_{129} \oplus sk_{105} \oplus sk_{103}\}\end{aligned} \tag{12}$$

which is calculable from both k'_{f1} and k'_{b1}. Then for each value of (S_{44}, κ) there is only one matching k_{f1} on the average.

In the attack, we first check for matching in the first dimension, which gives only one candidate for (k'_{f1}, k'_{b1}) in average. From the (k'_{f1}, k'_{b1}) candidate we can calculate values of k'_{f2} and k'_{b2} and check for matching in the second dimension, where 2^{-2} of the key candidates will be accepted. By testing each key on some other plaintext/ciphertext pairs, we find the correct key. The whole algorithm of the attack is described as Algorithm 3.

This attack needs $\lceil 80/32 \rceil = 3$ pairs of known plaintext/ciphertext. The main memory complexity of the attack is storing the $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}'_2$ tables and it is equal to

$$2^{78} \times 78 + 2^{78} \times 2 + 2^{48} \times 2 \approx 2^{78} \times 80 \tag{13}$$

bits which is equal to 2^{78} of the 2^{80} possible keys.

Time complexity analysis For calculations in the forward side of the first dimension, first we encrypt the plaintext for 4 rounds and save the state in the memory before guessing the k_0 key bit. Again we encrypt one more round, save it and guess the next necessary key bit(s). We continue this way until we reach $S_{44} = E'_{0,44}(k_{f1}, P)$ and then we *xor* it with $L'_{f1} \cdot k_{f1}$. When backtracking to try another guess, we only go back to the last state where we still have untried guesses to make.

In the case of partial matching, when we are close to the state where we will do the matching, one of the two functions f_a and f_b may not affect the state to match, and so it does not need to be computed. For this reason, when counting the number of round encryptions we count the number of individual f_a and f_b computations, and multiply the number with 1/2 to reach the number of round encryptions. The computation time to create the table $\mathcal{T}_1$ is then equal to

$$\tfrac{1}{2}[8 + 2(2 + 2(2 + 2^2(\ldots(2 + 2^2(2))\ldots)))] \tag{14}$$

Algorithm 3 2D MITM attack to KATAN32

```
for $k_{f1} \in K'_{f1}$ do
    Compute all 32 bits of $v_1 = E'_{0,44}(k_{f1}, P) \oplus L'_{f1} \cdot k_{f1}$ for a plaintext $P$;
    Compute $\kappa$ from $k_{f1}$;
    Store $k_{f1}$ into a table $\mathcal{T}_1$ indexed by $v_1$ and $\kappa$;
end for
for $k_{b2} \in K'_{b2}$ do
    Compute the 2 least significant bits of $v_2 = D'_{178,107}(k_{b2}, C) \oplus L'_{b2} \cdot k_{b2}$ for the ciphertext $C$ corresponding to $P$;
    Store $k_{b2}$ into a table $\mathcal{T}_2$ indexed by $v_2$;
end for
for $s \in \mathbb{F}_2^{32}$ do
    for $k_{f2} \in K'_{f2}$ do
        Compute the 2 least significant bits of $v'_2 = E'_{73,107}(k_{f2}, s) \oplus L'_{f2} \cdot k_{f2}$;
        Store the $k'_{f2}$ into a table $\mathcal{T}'_2$ indexed by $v'_2$;
    end for
    for $k_{b1} \in K'_{b1}$ do
        Compute all 32 bits of $v'_1 = D'_{73,44}(k'_{b1}, s) \oplus L'_{b1} \cdot k'_{b1}$;
        Compute $\kappa'$ from $k_{b1}$;
        Find the value for $k_{f1}$ using index of $(v'_1, \kappa')$ in table $\mathcal{T}_1$;
        Compute values of $k_{f2}$ and $k_{b2}$ using both $k_{f1}$ and $k_{b1}$ values;
        if $\mathcal{T}'_2[k_{f2}] = \mathcal{T}_2[k_{b2}]$ then
            Test the candidate key using this plaintext/ciphertext pair and a few other pairs;
            if If one candidate key passes all tests then
                Return this candidate as correct key;
            end if
        end if
    end for
end for
```

round encryptions. Expressions of this kind, with t nested brackets, can be accurately simplified as follows:

$$\begin{aligned} 2 + 2^2(2 + 2^2(\ldots 2 + 2^2(2)\ldots)) &= \\ 2 + 2^3 + 2^5 + \ldots + 2^{2t+1} &= \\ 2 \times \sum_{i=0}^{t} 4^i &= \\ 2 \times \frac{4^{t+1} - 1}{4 - 1} &\approx \\ \frac{4}{3} \times 2^{2t+1} \end{aligned}$$

Inserting this into (14), and compensating with the few rounds where only one key needs to be guessed, gives the number of round encryptions to compute $\mathcal{T}_1$ as approximately 1.33×2^{78}.

In the backward side of the first dimension, after guessing a value for S_{73}, first we decrypt 4 f_a and 6 f_b round functions (equal to 5 rounds) and save the state of both registers in the memory. Then we guess the sk_{144} and sk_{145} subkey bits. Then we decrypt one more pair of round functions, save the state and guess the next necessary subkey bits. We continue this way until we reach $S_{44} = D'_{73,44}(k_{b1}, S_{73})$ and then we *xor* it with $L'_{b1} \cdot k_{b1}$. The computation time for these operations are

$$\tfrac{1}{2}[10 + 2^2(2 + 2^2(...(2 + 2^2(1 + 2^2(1))))...)))] \approx 0.71 \times 2^{48} \tag{15}$$

Unlike the first dimension where all round functions were calculated, in the second dimension some round functions do not need to be calculated because we only do partial matching, in two bits. In the forward side, we first encrypt 4 rounds, save the state and then guess the sk_{147} subkey bit. Again we encrypt one more round, save it and guess the next necessary key bits in the same manner as before. Note that key bits do not need to be guessed when they only affect the computations linearly. We count the number of round functions which have effect on the 2 LSBs of S_{108} and get the result as 0.67×2^{48} round encryptions.

Finally, in the backward side of the second dimension again we decrypt 4 f_a and 6 f_b round functions at first and save the state and then guess the sk_{354} and sk_{355} subkey bits. We continue as before and guess enough key material to compute all round functions which have an effect on the value of the 2 LSBs of S_{108}. The computation time for this is 2.25×2^{78}.

We now calculate the complexity of the whole attack in terms of 179-round KATAN32 encryptions. Creating the tables $\mathcal{T}_1$ and $\mathcal{T}_2$ is only done once. The guessing of the 32-bit value for S_{73} must be taken into account when doing computations that depend on this value. Finally, as we only have a 2-bit filter in the second dimension, we expect to test about 2^{78} full keys for correctness. The total complexity of the attack in terms of 179-round KATAN32 encryptions then becomes

$$\tfrac{1}{179}(1.33 \times 2^{78} + 2.25 \times 2^{78} + 2^{32}(0.67 \times 2^{48} + 0.71 \times 2^{48})) + 2^{78} \approx 2^{78.07} \tag{16}$$

4.2 2D MITM attack on KATAN48

In this subsection we present a 2D MITM attack to a 148-round version of KATAN48. We break these 148 rounds into two sub-ciphers by guessing the state S_{60}. In the first dimension, P and S_{60} meet each other in round 42 and in the second dimension, S_{60} and C meet each other in the LSB ($L2[0]$) of S_{85}.

For the first dimension, we guess 77 key bits in the forward side and 32 subkey bits in backward side. These key bits are

$$\begin{aligned} k'_{f1} &= \{k_0, ..., k_{76}\} \\ k'_{b1} &= \{sk_{88}, sk_{90}, sk_{92}, ..., k_{121}\}. \end{aligned} \tag{17}$$

In the second dimension we guess 31 subkey bits in the forward side and 78 subkey bits in backward side. These key bits are

$$\begin{aligned} k'_{f2} &= \{sk_{120}, ..., sk_{144}, sk_{146}, sk_{147}, sk_{148}, sk_{153}, sk_{154}, sk_{161}\} \\ k'_{b2} &= \{sk_{219}, sk_{220}, sk_{221}, sk_{223}, ...sk_{297}\}. \end{aligned} \tag{18}$$

Like in the attack for KATAN32, we define a value κ which we use to create unique states for all key guesses. The state size in KATAN48 is 48 bits, so we need the κ-value to be $77 - 48 = 29$ bits in the first dimension. This κ is defined as

$$\begin{aligned} \kappa = \{&sk_{88}\ , sk_{93}\ , ..., sk_{102}\ , sk_{106}, ..., sk_{115}\ , sk_{119}\ , sk_{120}\ , sk_{121}\ , sk_{116} \oplus sk_{90}\ , \\ &sk_{103} \oplus sk_{90}\ , sk_{117} \oplus sk_{104}\ , sk_{105} \oplus sk_{92}\ , sk_{118} \oplus sk_{92}\ \} \end{aligned} \tag{19}$$

The algorithm for the attack is essentially the same as the attack on KATAN32 in Section 4.1, only with changed round numbers. This attack needs $\lceil 80/48 \rceil = 2$ pairs of known plaintext/ciphertext and memory complexity of this attack is equal to

$$2^{77} \times 77 + 2^{78} \times 1 + 2^{31} \times 1 \approx 2^{77} \times 79 \tag{20}$$

bits which is about the same as the storage for 2^{77} keys.

Time complexity analysis For partial encryption and decryption calculations in this attack we use the same method used in Section 4.1. For KATAN48, in partial encryptions we can encrypt 3.5 rounds as early calculation and for partial decryptions, we can decrypt 3 f_a and 7 f_b round functions (equal to 2.5 rounds) as early calculation.

For the key guessing and the rest of the partial encryptions or decryptions we use same method as in the attack on KATAN32. For each guess, we save the last calculated state, guess the next subkey bit(s) and calculate one more pair of round functions. The only difference is that in KATAN48 one round consists of computing f_a and f_b two times each, and not once each as in KATAN32. Hence we scale the number of f_a and f_b invocations by $\frac{1}{4}$ to estimate the number of rounds. The computation times for the different phases of this attack are given below, but we omit the long expressions with nested brackets for better readability.

- Forward side of the first dimension: 1.04×2^{77}
- Backward side of the first dimension: 0.83×2^{32}
- Forward side of the second dimension: 1.00×2^{31}
- Backward side of the second dimension: 3.13×2^{78}

With these numbers, the time complexity of the attack in terms of 148-round KATAN48 encryptions is equal to

$$\tfrac{1}{148}(1.04 \times 2^{77} + 3.13 \times 2^{78} + 2^{48}(0.83 \times 2^{32} + 1.00 \times 2^{31})) + 2^{79} \approx 2^{79.03} \tag{21}$$

The heaviest part of the attack is testing candidate keys. Since we only match in one bit in the second dimension, we must expect to test 2^{79} keys. If we use two matching bits in the second dimension of the attack, we can cryptanalyse 146 rounds of KATAN48 with time complexity of $2^{78.06}$. The attack will be the same, except that S_{60} and C meet each other in the two LSBs of S_{84}.

4.3 2D MITM attack on KATAN64

This subsection presents our 2D MITM attack applied to a 129-round version of KATAN64. We break the 129 rounds into two sub-ciphers by guessing S_{51}. In the first dimension, P and S_{51} meet each other in round 43 and in the second dimension, S_{51} and C meet each other in the LSB ($L2[0]$) of S_{69}. The attack is mostly the same as before, but the state size and the round numbers have changed and there is an extra twist as we only need to guess 59 of the 64 state bits going forward in the second dimension (explained below).

For the first dimension, we guess 77 key bits in the forward side and 16 subkey bits in the backward side. The key bits guessed are

$$\begin{aligned} k'_{f1} &= \{k_0, ..., k_{76}\}, \\ k'_{b1} &= \{sk_{88}, ..., k_{103}\}. \end{aligned} \tag{22}$$

In the second dimension, we guess 20 subkey bits in the forward side and 78 subkey bits in the backward side. These subkey bits are

$$\begin{aligned} k'_{f2} &= \{sk_{102}, ..., sk_{116}, sk_{118}, sk_{120}, sk_{122}, sk_{123}, sk_{129}\} \\ k'_{b2} &= \{sk_{181}, sk_{183}, ..., sk_{259}\}. \end{aligned} \tag{23}$$

In the forward side of the second dimension 5 bits ($L2[38, 34, 29, 25, 17]$) of S_{51} have no effect on the LSB of S_{69}, so we do not need to guess the value of these bits when going forward in the second dimension. This means we can guess key bits for a few more rounds before having to stop to stay below exhaustive search complexity.

As before, to get unique states for each key guess in the forward side of the first dimension we need to define a κ-value computable from both k'_{f1} and k'_{b1}. In KATAN64, the κ-value should consist of $77 - 64 = 13$ bits, and is chosen as

$$\kappa = \{sk_{88}\ , sk_{89}\ , sk_{93}\ , ..., sk_{102}\ , sk_{90} \oplus sk_{103}\}. \tag{24}$$

This attack needs $\lceil 80/64 \rceil = 2$ pairs of known plaintext/ciphertext and the memory complexity of the attack is equal to

$$2^{77} \times 77 + 2^{78} \times 1 + 2^{20} \times 1 \approx 2^{77} \times 79 \tag{25}$$

bits, which is about the same size as storing 2^{77} keys.

Time complexity analysis For early calculation in the encryption, we can encrypt $\frac{10}{3}$ rounds before starting to guess key bits. For partial decryptions, we can decrypt 4 f_a and 5 f_b functions (equal to 1.5 rounds) as early calculation. In KATAN64 each f_a and f_b are computed three times each in every round, hence we will multiply the number of f_a and f_b invocations with $\frac{1}{6}$ to get the number of rounds needed to be computed in the attack.

The computation time in number of rounds for the four stages of this attack are given below.

- Forward side of the first dimension: 0.89×2^{77}
- Backward side of the first dimension: 0.83×2^{16}
- Forward side of the second dimension: 0.28×2^{20}
- Backward side of the second dimension: 1.75×2^{78}

The total time complexity of the attack, in terms of 129-round KATAN64 encryptions, is then equal to

$$\frac{1}{129}(0.89 \times 2^{77} + 1.75 \times 2^{78} + 0.83 \times 2^{64+16} + 0.28 \times 2^{59+20})) + 2^{79} \approx 2^{79.03} \tag{26}$$

We have the same trade-off as in the attack for KATAN48, between number of rounds and complexity. If we use two matching bits in the second dimension of the attack, we can cryptanalyse 126 rounds of KATAN64 with time complexity of $2^{78.06}$. However, we still believe the 129-round attack with complexity $2^{79.03}$ is valid.

4.4 3D MITM attack on KATAN32

In this subsection we present an attack on a 206-round version of KATAN32, using a 3D MITM attack. We split these 206 rounds into three sub-ciphers by guessing both S_{73} and S_{97}.

The first dimension is exactly the same as the first dimension of 179-round 2D MITM attack to KATAN32. In the second dimension, S_{73} and S_{97} meet each other in all 32 bits of S_{89}. In the third dimension, S_{97} and C meet each other in the LSB ($L2[0]$) of S_{134}.

In the second dimension, we guess 32 subkey bits in the forward side and 16 subkey bits in the backward side. These key bits are

$$\begin{aligned} k_{f2} &= \{sk_{146}, ..., sk_{177}\}, \\ k_{b2} &= \{sk_{178}, ..., sk_{193}\}. \end{aligned} \tag{27}$$

In this dimension we can not use the technique of (4), and we must guess all subkeys, including only *xor*ed subkey bits, for partial encryption or decryption. This is because we do not have enough key material in this dimension to express the only-linearly involved key bits as linear combinations of the other key bits.

In the third dimension we guess 48 subkey bits in the forward side and 78 subkey bits in the backward side. These particular bits are

$$k'_{f3} = \{sk_{194}, ..., sk_{233}, sk_{235}, sk_{236}, sk_{238}, sk_{240}, sk_{242}, sk_{246}, sk_{249}, sk_{255}\},$$

$$k'_{b3} = \{sk_{317}, sk_{321}, sk_{333}, sk_{335}, sk_{337}, sk_{338}, sk_{339}, sk_{341}, ..., sk_{411}\}. \tag{28}$$

In the attack, we first check for matching in the first dimension, which gives one candidate for (k'_{f1}, k'_{b1}) on the average for any given value s of S_{73}. The pair of (k'_{f1}, k'_{b1}) determines all 80 bits of the master key K. We compute (k_{f2}, k_{b2}) for this K, then save K in a table $\mathcal{T}'_1$ indexed by (k_{f2}, k_{b2}). For a fixed S_{73}, we are making 2^{48} guesses for k'_{b1}. Therefore, as (k_{f2}, k_{b2}) is also a 48-bit value, for every index of $\mathcal{T}'_1$ there will be one K in average.

Next, we check for matching in the second dimension. For every candidate value of (k_{f2}, k_{b2}), we find $K = \mathcal{T}'_1[(k_{f2}, k_{b2})]$ and calculate values for k'_{f3} and k'_{b3} from this K. We then check for matching in the third dimension, where 2^{-1} of the key candidates will be accepted. By testing each key passing the matching in the third dimension on some other plaintext/ciphertext pairs, we find the correct key. The algorithm of the attack is described in Algorithm 4.

The main memory complexity of the attack is storing the $\mathcal{T}_1$, $\mathcal{T}'_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}'_3$ tables. This memory complexity is equal to

$$2^{78} \times 78 + 2^{48} \times 80 + 2^{32} \times 32 + 2^{32+48} \times 1 + 2^{78} \times 1 \approx 2^{78} \times 83 \tag{29}$$

bits which is equal to 2^{78} of the 2^{80} possible keys.

Time complexity analysis In both sides of the second dimension there are 10 subkey bits which only get *xor*ed onto S_{89}, so the time for calculations in the second dimension is negligible compared to the other dimensions. The time for calculations in the third dimension are given below.

– Forward side of the third dimension: 2^{48}
– Backward side of the third dimension: 2^{78}

The time complexity of the total attack is then equal to

$$\frac{1}{206}(1.33 \times 2^{78} + 2^{78} + 2^{32}(0.71 \times 2^{48} + 2^{48})) + 2^{79} \approx 2^{79.02} \tag{30}$$

In the same way as in the attacks in the previous sections, if in the third dimension of the attack we use two matching bits, we can attack 201 rounds of KATAN32 with time complexity of $2^{78.04}$. In this case, instead of the one bit matching described above, the 2 LSBs of S_{130} are chosen for matching.

5 Conclusions

In this paper we have introduced new MD MITM attacks to KATAN family of block ciphers. Due to the low algebraic degree of the round function and the slow

Algorithm 4 3D MITM attack on KATAN32

```
for $k_{f1} \in K'_{f1}$ do
    Compute 32 bits of $v_1 = E'_{0,44}(k_{f1}, P) \oplus L'_{f1} \cdot k_{f1}$ for a plaintext $P$;
    Compute $\kappa$;
    Store $k_{f1}$ in a table $\mathcal{T}_1$ indexed by $v_1$ and $\kappa$;
end for
for $k_{b3} \in K'_{b3}$ do
    Compute the least significant bit of $v_3 = D'_{206,134}(k_{b3}, C) \oplus L'_{b3} \cdot k_{b3}$ for $C$ the ciphertext corresponding to $P$;
    Store the $v_3$ 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 the least significant bit of $v'_3 = E'_{97,134}(k_{f3}, t) \oplus L'_{f3} \cdot k_{f3}$;
    Store the $v'_3$ 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 32 bits of $v'_1 = D'_{73,44}(k_{b1}, s) \oplus L'_{b1} \cdot k_{b1}$;
        Compute $\kappa'$;
        Find $k_{f1} = \mathcal{T}_1[(v'_1, \kappa')]$;
        Compute 80-bit master key candidate $K$;
        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 $v_2 = E_{73,89}(k_{f2}, s)$;
        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 $v'_2 = D_{97,89}(k_{b2}, t)$;
        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 plaintext/ciphertext pair and a few other pairs;
            if candidate key matches the plaintext/ciphertext pairs then
                Return candidate key as the correct key
            end if
        end if
    end for
end for
```

key diffusion, we can apply some new techniques which significantly increases the number of rounds attacked. Our results show that this method can attack up to 206, 148 and 129 rounds out of 254 rounds of KATAN32, KATAN48 and KATAN64, respectively, with only 2 or 3 plaintext/ciphertext pairs of known data. These attacks cover the highest number of rounds to date and are able to cryptanalyse 29, 18 and 10 more rounds of KATAN32, KATAN48 and KATAN64, compared to the best previously published attacks. Our work is still quite a way from attacking full 254-round KATAN, so it could not be considered as a threat to this family of block ciphers yet. We state that KATAN is still safe to use.

References

1. C. De Cannière, O. Dunkelman, and M. Knezevic, "KATAN and KTANTAN - a Family of Small and Efficient Hardware-oriented Block Ciphers", *Cryptographic Hardware and Embedded Systems (CHES) 2009*, vol. 5747 of Lecture Notes in Computer Science, pp. 272-288, Springer, 2009.
2. S. Knellwolf, W. Meier, M. Naya-Plasencia, "Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems", In M. Abe, *ASIACRYPT 2010*, LNCS, vol. 6477, pp. 130145, Springer, 2010.
3. S. Knellwolf, W. Meier, M. Naya-Plasencia, "Conditional Differential Cryptanalysis of Trivium and KATAN" In A. Miri, S. Vaudenay, *Selected Areas in Cryptography, SAC 2011*, LNCS, vol. 7118, pp 200-212 , Springer, 2012.
4. S. Knellwolf, "Accelerated Key Search for the KATAN Family of Block Ciphers", *ECRYPT Workshop on Lightweight Cryptography*, November, 2011.
5. T. Isobe and K. Shibutani, "All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach", In L. R. Knudsen and H. Wu, *Selected Areas in Cryptography, SAC 2012*, LNCS, vol. 7707, pp. 202-221. Springer, 2012.
6. M. R. Albrecht and G. Leander, "An All-in-One Approach to Differential Cryptanalysis for Small Block Ciphers", In L. R. Knudsen and H. Wu, *Selected Areas in Cryptography, SAC 2012*, LNCS, vol. 7707, pp. 1-15. Springer, 2012.
7. T. Isobe, Y. Sasaki, and J. Chen, Related-Key Boomerang Attacks on KATAN32/48/64, In C. Boyd and L. Simpson, *ACISP 2013* , LNCS, vol. 7959, pp. 268-285, Springer, 2013.
8. T. Isobe and K. Shibutani, "Improved All-Subkeys Recovery Attacks on FOX, KATAN and SHACAL-2 Block Ciphers", *Fast Software Encryption, FSE 2014*, LNCS, vol. 8540, pp. 104-126, Springer, 2015.
9. T. Fuhr and B. Minaud, "Match Box Meet-in-the-Middle Attack Against KATAN", *Fast Software Encryption, FSE 2014*, LNCS, vol. 8540, pp. 61-81, Springer, 2015.
10. B. Zhu and G. Gong, "Multidimensional Meet-in-the-Middle Attack and Its Applications to KATAN32/48/64", Accepted in *IET Information Security*, 2015.
11. 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.
12. 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", In G. Gong, K. C. Gupta, *INDOCRYPT 2010*, LNCS, vol. 6498, pp. 176-196, Springer, 2010.
13. A. Bogdanov and C. Rechberger, "A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN". In A. Biryukov, G. Gong, D. R. Stinson *Selected Areas in Cryptography, SAC 2010*, LNCS, vol. 6544, pp. 229-240, 2010.
14. L. Wei, C. Rechberger, J. Guo, H. Wu, H. Wang, and S. Ling, "Improved meet-in-the-middle cryptanalysis of KTANTAN". In U. Parampalli and P. Hawkes, *Australasian Conference on Information Security and Privacy, ACISP 2011*, LNCS, vol. 6812, pp. 433-438. Springer, 2011.

15. M. Agren, "Some Instant- and Practical-Time Related-Key Attacks on KTANTAN32/48/64". In A. Miri, S. Vaudenay, *Selected Area in Cryptography SAC 2011*, LNCS, vol. 7118, pp. 213-229. Springer, 2012.
16. S. F. Abdul-Latip, R. Reyhanitabar, W. Susilo, and J. Seberry, "Fault Analysis of the KATAN Family of Block Ciphers", In Mark D. Ryan, B. Smyth and G. Wang, *Information Security Practice and Experience, ISPEC 2012*, LNCS, vol. 7232, pp.319-336, Springer, 2012.
17. L. Song and L. Hu, "Improved Algebraic and Differential Fault Attacks on the KATAN Block Cipher", In R. H. Deng and T. Feng, *Information Security Practice and Experience, ISPEC 2013*, LNCS, vol. 7863, pp 372-386, Springer, 2013.
18. F. M. Quedenfeld, "Algebraic Fault Analysis of KATAN", Cryptology ePrint Archive, report 2014/954, 2014.
19. W. Diffie, and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *IEEE Computer Society Press*, vol. 10(6), pp. 74-84, 1977.