A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN

Andrey Bogdanov and Christian Rechberger

Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Belgium
{andrey.bogdanov,christian.rechberber}@esat.kuleuven.be

Abstract. In this paper we describe a variant of existing meet-in-the-middle attacks on block ciphers. As an application, we propose meet-in-the-middle attacks that are applicable to the KTANTAN family of block ciphers accepting a key of 80 bits. The attacks are due to some weaknesses in its bitwise key schedule[1]. We report an attack of time complexity $2^{75.170}$ encryptions on the full KTANTAN32 cipher with only 3 plaintext/ciphertext pairs and well as $2^{75.044}$ encryptions on the full KTANTAN48 and $2^{75.584}$ encryptions on the full KTANTAN64 with 2 plaintext/ciphertext pairs. All these attacks work in the classical attack model without any related keys.
In the differential related-key model, we demonstrate differentials holding with probability 1 for up to 218 rounds. This shows that a strong related-key property can translate to a successful attack in the non-related-key setting. Having extremely low data requirements, these attacks are valid even in RFID-like environments where only a very limited amount of text material may be available to an attacker.
Keywords: cryptanalysis, meet-in-the-middle attacks, block cipher, key schedule, lightweight cipher, key-recovery, RFID

1 Introduction

A number of new cipher designs have been proposed recently, targeting use cases with severe implementation constraints imposed. Block cipher design methods have advanced to a stage where strong arguments for the resistance of the design against large classes of attacks such as differential and linear cryptanalysis are possible. However, if aggressive design decisions have been made forced by a restrictive application scenario, some other, more dedicated analysis techniques may turn out useful for attacking the cipher.

[1] The SAC 2010 pre-proceedings version of this paper [6] was based on a key-schedule from a previous version of the reference code which contained errors. This paper is based on the corrected reference code available under [1].

Cryptographic techniques move into applications like sensor nodes, RFID tags, or the "the Internet of things" at large. The ever increasing demand for security and privacy in these very constrained environments requires new cryptographic primitives, like tiny yet efficient ciphers. A number of designs and implementation techniques have been proposed recently to address this need. Stream ciphers like Trivium [10, 8], Grain [17, 18], or Mickey [3], or block ciphers like DESL [22], PRESENT [4], HIGHT [19], mCrypton [23], KATAN and KTANTAN [9], or the hash functions based on PRESENT [5] are among the important ones.

Motivation. The economical and physical constraints force designers to make design decisions which are often considered to be "on the edge". In this context it is often argued that block ciphers are better understood than stream ciphers and are hence more trustworthy. Some recent designs like PRESENT and KATAN/KTANTAN have come with strong arguments that large classes of attacks shown powerful in the past are not applicable. The technique used is to provide bounds on various non-random properties, like differential or linear characteristics. Indeed, whereas in the eStream project a number of lightweight stream ciphers were broken, sometimes even with practical attack complexities, none of the recently proposed block ciphers have been broken so far.

KTANTAN [9] accepts a key of 80 bits. It was designed to resist differential and linear attacks, and exhibits strong bounds in the non-related-key model, an upper bound 2^{-b} on the probability of every differential/linear characteristic over 128 rounds being an essential design criterion ($b \in \{32, 48, 64\}$ is the block size of the cipher). Even if related keys are considered, a much less realistic setting, designers report no differential characteristic with a higher probability than 2^{-b} for 150 out of the 254 rounds. In [2], Albrecht et al. study algebraic approaches to amplify differential attacks on this family of ciphers.

Table 1. Results on MITM cryptanalysis for KTANTAN

attack/bound	cipher $b \in \{32, 48, 64\}$	#rounds (of 254)	time [encryptions]	data compl. [PT/CT pars]
[9], RK diff. bound	KTANTANb	150	$\mathcal{O}(2^b)$	$\mathcal{O}(2^b)$
[9], DC and LC bound	KTANTANb	128	$\mathcal{O}(2^b)$	$\mathcal{O}(2^b)$
this paper, MITM attack	KTANTAN32	254	$2^{75.170}$	3
this paper, MITM attack	KTANTAN48	254	$2^{75.044}$	2
this paper, MITM attack	KTANTAN64	254	$2^{75.584}$	2

Contributions and Outline. Section 2 considers a framework for MITM attacks. In Section 4, based on this framework, we propose a key-recovery attack on the KTANTAN block cipher family (briefly described in Section 3), requiring only very few known plaintext/ciphertext pairs. Hence this kind of attacks is even valid in very restrictive RFID-like environments and protocols where only a very limited number of transactions are foreseen in the lifetime of a tag. The parameters and complexities of our attacks are provided in Table 1. Some properties we use translate to probability-1 related-key differentials over many rounds which are outlined in Table 4. Note that the data complexity of our attacks is the lowest possible and exactly corresponds to that of a brute-force attack. We conclude with a discussion on links to other works, high-level design choices for low-resource ciphers, and future work in Section 5.

2 Framework for MITM Attacks

2.1 Basic MITM Attack

The basic meet-in-the-middle (MITM) approach will be a starting point for our attack. MITM techniques are arguably much less common than differential or linear attacks on ciphers. There are some applications of MITM principles to block ciphers like DES or AES, see e.g. [7, 11, 12, 14, 15, 20] for dedicated attacks, and e.g. [24, 25] for meet-in-the-middle attacks on a higher level.

The basic MITM technique is due to Diffie and Hellman [13]. Let $\varphi_{i,j}$ denote the partial transform of an R-round block cipher beginning in round i and ending directly after round j under some fixed key, $1 \leq i \leq j \leq R$, see Figure 1. Then if $\varphi_{1,\alpha}$ and $\varphi_{\alpha+1,R}$ use subkeys with distinct key bits, the key can be as a rule recovered much more efficiently than by brute force over two subkeys. The central idea here, as also applied to reduced DES in [15], is that the subkeys in both parts of the cipher can be guessed independently. Each guess of the first subkey allows the adversary to compute $\varphi_{1,\alpha}(p)$ and of the second subkey to obtain $\varphi^{-1}_{\alpha+1,R}(c)$. The right key will be among those fulfilling the equation $\varphi_{1,\alpha}(p) = \varphi^{-1}_{\alpha+1,R}(c)$.

2.2 The 3-Subset MITM Approach

Here we consider a variant of the basic MITM attack. The idea is to remove restrictions on the choice of key bits, thereby potentially allowing attacks where an attack is not possible with the basic MITM approach. Instead of considering two subsets of key bits, we consider three subsets.

The attack consists of two parts. In the *MITM stage*, we filter out some wrong key candidates and reduce the key space. In the *key testing stage*, we look for the right key in the reduced key space.

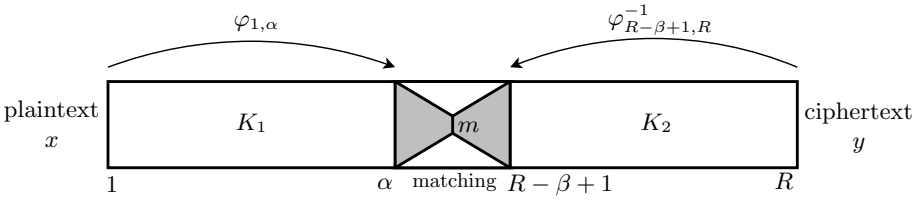

Fig. 1. MITM

Let $K = k_{\ell-1}k_{\ell-2}\dots k_1k_0$ be the ℓ-bit key. Then if $K_1 = \{k_i : k_i \text{ used by } \varphi_{1,\alpha}\}$ and $K_2 = \{k_i : k_i \text{ used by } \varphi_{R-\beta+1,R}\}$, then $A_0 = K_1 \cap K_2$ is the set of key bits used both by the first α and last β rounds, see Figure 1. Moreover, $A_1 = K_1 \backslash K_1 \cap K_2$ and $A_2 = K_2 \backslash K_1 \cap K_2$ are the sets of key bits used by K_1 only and by K_2 only, respectively. We further assume that $K_1 \cup K_2 = K$.

For the attack we need n plaintext/ciphertext pairs $\{(x_i, y_i)\}$, $i = 1, \dots, n$. Let x_i be plaintext and y_i ciphertext.

MITM Stage. The meet-in-the-middle part of the attack on R rounds of the cipher can be performed as follows:

- For each guess of key bits in A_0:
  - For each guess of key bits in A_1:
    - Compute $v = \varphi_{1,\alpha}(x)$
  - For each guess of key bits in A_2:
    - Compute $u = \varphi^{-1}_{R-\beta+1,R}(y)$
  - Perform matching in the middle between the values of v and the values of u on m bits, $1 \le m \le b$ (see Subsection 4.3) and add surviving key candidates to the list $\mathcal{K}$ of surviving keys. We expect to have false positives with probability 2^{-m} which is called the *false positive rate* of a MITM attack.

Key Testing Stage. In this stage, we test the surviving key candidates from $\mathcal{K}$ using some plaintext-ciphertext pairs in a brute-force manner. Generally speaking, it is not necessary to use additional plaintext-ciphertext pairs. The number of the texts needed is defined by the unicity

distance of the cipher which essentially depends on the block size and key length. If we can significantly reduce the key space in the MITM stage (by ruling out a large part of the keys), the complexity of the key testing stage will be negligible with respect to the MITM stage. Generally speaking, however, this is not necessarily the case.

Attack Complexity. The computational complexity of the attack will be dominated by

$$C_{\text{comp}} = \underbrace{2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})}_{\text{MITM stage}} + \underbrace{(2^{\ell-m} + 2^{\ell-m-b} + 2^{\ell-m-2b} + \dots)}_{\text{key testing stage}}. \quad (1)$$

If A_1 and A_2 are both non-empty and $|A_1| + |A_2| > 2$, then the attack becomes more efficient than exhaustive search provided that the false positive rate is low enough.

The MITM stage requires exactly one plaintext/ciphertext pair. However, of the b bits only m are used for matching. That is, the information contained in the other $b - m$ state bits is not used in this stage and can be used in the key testing stage. For the key testing stage, more pairs might be required, depending on the relation between the key length and the block size. This results in data complexity

$$C_{\text{data}} = \left\lceil \frac{\ell}{b} \right\rceil$$

depending on the block size b and the key length ℓ. The memory complexity is defined by matching in the MITM stage. For small sets A_1 and A_2, it is negligible.

3 A Short Description of KTANTAN

KTANTAN is a block cipher which accepts an 80-bit user-supplied key. Versions with block size $b \in \{32, 48, 64\}$ bit have been specified. Each version has 254 rounds. While the definition of a round transform differs from version to version, the key schedule remains the same. Throughout the paper, we refer to the KTANTAN version with b-bit blocks as KTANTANb.

3.1 Round Transform

KATAN and KTANTAN share the specification of a round transform, as the operations on the state are exactly the same up to the key schedule.

The state of the cipher is represented as two disjunct parts L_1 and L_2. The transform of round r is based on two Boolean functions $f_{1,r}$ and $f_{2,r}$, having L_1 and L_2 as their domain, correspondingly:

$$f_{1,r}(L_1) = L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \cdot L_1[x_4]) \oplus (L_1[x_5] \cdot IR_r) \oplus \kappa_{1,r}$$
$$f_{2,r}(L_2) = L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \cdot L_2[y_4]) \oplus (L_2[y_5] \cdot L_2[y_6]) \oplus \kappa_{2,r},$$

where x_i, y_i are the numbers of active bit positions, IR_r is the round constant bit in round r, and $\kappa_{1,r}$, $\kappa_{2,r}$ are the bits of the extended key defined by the key schedule for round r. The lengths of L_1 and L_2 as well as the bit positions x_i, y_i are specific for each KTANTAN version.

Once $f_{1,r}$ and $f_{2,r}$ are computed, the registers L_1 and L_2 are shifted, the MSB of each register falls off and the LSB is set to the output of $f_{2,r}$ and the output of $f_{1,r}$, respectively. KTANTAN32 applies transformations $f_{1,r}$ and $f_{2,r}$ once in a round. One round of KTANTAN48 and KTANTAN64 updates the registers using $f_{1,r}$ and $f_{2,r}$ two and three times, respectively.

Table 2. Version-specific parameters of KTANTAN

b	$\|L_1\|$	$\|L_2\|$	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. Round constant bits IR_r: IR_1 first, IR_{254} last

```
1111111000 1101010101 1110110011 0010100100 0100011000 1111000010
0001010000 0111110011 1111010100 0101010011 0000110011 1011111011
1010010101 1010011100 1101100010 1110110111 1001011011 0101110010
0100110100 0111000100 1111010000 1110101100 0001011001 0000001101
1100000001 0010
```

3.2 Key Schedule

The functions $f_{1,r}$ and $f_{2,r}$ require input from the key schedule, which is a function mapping the 80-bit user-supplied key $K = k_{79}k_{78}\ldots k_1k_0$ to $\kappa_{1,r}$ and $\kappa_{2,r}$ for each round r. It is exactly this part where KTANTAN

differs from KATAN. This difference, together with some properties of the data transform, makes KTANTAN vulnerable to our attack.

An 8-bit round counting LFSR is used to control the key schedule. It is defined by the feedback polynomial

$$\zeta^8 + \zeta^7 + \zeta^5 + \zeta^3 + 1$$

and its initial state is all ones. The value of IR_r is specified as the most significant bit of this LFSR in round r. Let $l_{7,r}l_{6,r}...l_{1,r}l_{0,r}$ denote the 8-bit state of the LFSR in round r.

The key schedule of KTANTAN chooses two bits of K in each round. This is done by applying two layers of MUX logic. First, K is divided into 5 chunks W_i of 16 bits each: $K = W_4||W_3||W_2||W_1||W_0$. One bit out of each chunk is selected:

$$\omega_{i,r} = \text{MUX16to1}(W_i, l_{7,r}l_{6,r}l_{5,r}l_{4,r}), i = 0, \ldots, 4,$$

where the LFSR bits define the position in W_i to choose. Second, two out of these five bits are chosen controlled by the other half of the LFSR state:

$$\kappa_{1,r} = \overline{l_{3,r}} \cdot \overline{l_{2,r}} \cdot \omega_{0,r} \oplus (l_{3,r} \vee l_{2,r}) \cdot \text{MUX4to1}(\omega_{4,r}\omega_{3,r}\omega_{2,r}\omega_{1,r}, l_{1,r}l_{0,r})$$
$$\kappa_{2,r} = \overline{l_{3,r}} \cdot l_{2,r} \cdot \omega_{4,r} \oplus (l_{3,r} \vee \overline{l_{2,r}}) \cdot \text{MUX4to1}(\omega_{3,r}\omega_{2,r}\omega_{1,r}\omega_{0,r}, \overline{l_{1,r}l_{0,r}}).$$

4 Low Data-Complexity Attacks on KTANTAN

In here, we apply the MITM framework described in Section 2 to all 3 variants of the full KTANTAN with 254 rounds. The resulting attack requires an extremely small number of known plaintext/ciphertext pairs (basically, the minimum due to the unicity distance) and negligible memory. The MITM techniques make use of the fact that several key bits remain unused by the KTANTAN key schedule in large connected parts of the cipher. More precisely, it is the rounds at the beginning and end of KTANTAN which we are most interested in to make the attack work.

4.1 Related-Key Differentials of Probability 1

We start with a note on probability-1 related-key differentials of KTANTAN over many rounds.

The key observation here is that if certain key bits are not used over many rounds, they can be flipped without affecting the data transformation. In other words, the R-round related-key differential $(0, \Delta) \mapsto 0$

holds with probability 1, where 0 is the zero input and output difference in the data transformation and Δ is the key difference with ones at key bit position not used in the R rounds (first key bit positions are on the left-hand side). Some of the longest related-key differentials of this type we found for KTANTANb are given in Table 4. These differentials are due to the fact that large connected subsets of rounds do not use some key bits.

We notice that it seems possible that these properties can be turned into low-complexity differential related-key attacks on KTANTAN. However, we are mostly concerned with attacks that do not require related keys, and hence continue by exploiting this property in another way.

Table 4. Related key differentials for KTANTANb, $b \in \{32, 48, 64\}$

covered rounds	#rounds	differential	probability
$\varphi_{1,218}$	218	$(0, 00000000800000000000) \mapsto 0$	1
$\varphi_{1,140}$	140	$(0, 00000000800000040000) \mapsto 0$	1
$\varphi^{-1}_{81,254}$	174	$(0, 00000000000000010000) \mapsto 0$	1
$\varphi^{-1}_{82,254}$	173	$(0, 00000000000100010000) \mapsto 0$	1

4.2 Application of the MITM Framework to KTANTAN

Depending on the KTANTAN version, different properties of the key schedule are exploited in our attack. This is due to the fact that all three versions of KTANTAN have different numbers of register clocks in one round. The versions with a larger block size have heavier rounds with more diffusion which complicates the partial matching phase. Effectively, this might reduce the number of rounds in the middle for which matching is possible.

In Table 5 we give a summary of the properties and parameters of our attacks. We aim for the full, and if this is not possible for the highest number of rounds. Considering variants with a more reduced number of rounds would lead to more neutrals key bits, and generally lower attack complexities.
To illustrate the meaning of this table, let us consider the first entry. We attack the full 254-round KTANTAN32. The two basic properties of the KTANTAN key schedule which make our attack on KTANTAN32 possible can be formulated as:

Table 5. Details of the proposed attacks

b	R	α	A_1	$R-\beta$	A_2	matching bits m	complexity C_{comp} MITM	key test	total
32	254	111	32,39,44,61,66,75	131	3,20,41,47,63,74	8	75.000	72	75.170
48	254	111	32,39,44,61,66,75	131	3,20,41,47,63,74	10	75.000	70	75.044
64	254	123	32,44,61,66,75	131	3,20,41,47,63,74	47	75.584	33	75.584

Fact 1 *$\varphi_{1,\alpha}$ does not use key bits $\{k_{32}, k_{39}, k_{44}, k_{61}, k_{66}, k_{75}\}$ for $1 \leq \alpha \leq 111$.*

Fact 2 *$\varphi_{254-\beta+1,254}$ does not use key bits $\{k_3, k_{20}, k_{41}, k_{47}, k_{63}, k_{74}\}$ for $1 \leq \beta \leq 131$.*

This means that if we fix $\alpha = 111$ and $\beta = 131$, then $\varphi_{1,111}$ and $\varphi_{123,254}$ will have 6 *neutral* key bits in each direction. Moreover, one can efficiently match v and u in 8 bits, despite being 21 rounds apart due to the slow diffusion in one round of KTANTAN32, which is demonstrated in Subsection 4.3.

4.3 Partial-Matching Phase

In here, we describe in more detail the matching procedure that is used as a sub-routine in the MITM stage.

Procedure. The starting point are two completely determined internal states u and v several rounds apart (for KTANTAN32, one at round 111 and the other one at round 131). If we considered a cipher where those middle rounds were cut away, then the matching phase would be trivial as we would simply check if the $u = v$. With a probability of about 2^{-b} this check would give a false positive, but overall the number of key candidates is reduced to about 2^{80-b}. Remember that $b \in \{32, 48, 64\}$. Hence the number of key candidates is small enough to not influence the attack complexity during the key testing state.

In order to bridge this gap and to obtain a result on the full cipher, we drop the requirement to match on every state bit but allow for a much smaller number of matched bits m. This will increase the number of false positives, but in a way that does not noticeably influence any property of the attack. In more detail, we find that m bits (for KTANTAN32, $m = 8$) will still match with probability 1 (see below for details). This means that we will have reduced the number of key candidates to 2^{80-m} after

the MITM stage. In total, we hence need only between 2 (for block size $b = \{48, 64\}$) or 3 (for $b = 32$) known plaintext/ciphertext pairs.

We note that this procedure can be implemented in an essentially memoryless way, as every match can immediately be tested with another plaintext/ciphertext pair. Also in our estimate of the attack complexity, we do not consider any implementation optimizations that e.g. would also be possible for a brute force search that does not use any shortcut attacks. Examples of such optimizations would e.g. be a reuse of computations from one key guess to the next.

Details on the Partial Matching Phase. In the following we trace those bits that remain unaffected during the middle rounds for the block size of $b = 32$. '1' means affected, '0' means not affected. $k1$ and $k2$ denote disturbances caused by the unknown neutral bits from the opposite chunk at the respective rounds. For other block sizes, we refer to Appendix A.

```
10 bits match as follows

forward part:
forward R=111: k1=1, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=112: k1=0, k2=0  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=113: k1=0, k2=0  0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=114: k1=0, k2=0  0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=115: k1=0, k2=0  0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=116: k1=0, k2=0  1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0   1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=117: k1=0, k2=0  0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0   0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=118: k1=0, k2=0  0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0   1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=119: k1=0, k2=0  0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0   0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=120: k1=0, k2=0  1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0   1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=121: k1=0, k2=0  1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0   1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=122: k1=0, k2=0  1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0   1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=123: k1=1, k2=0  1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0   0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=124: k1=0, k2=0  1 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0   1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=125: k1=0, k2=0  1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0   1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=126: k1=0, k2=0  1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0   1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=127: k1=0, k2=1  1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0   1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

backward part:
backward R=131: k1=0, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
backward R=130: k1=1, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1
backward R=129: k1=0, k2=0  0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1   0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1
backward R=128: k1=0, k2=0  0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0   0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0
backward R=127: k1=0, k2=0  0 1 0 1 0 1 1 0 0 0 1 0 0 1 0 0   1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0

The highest number of matching bits is obtained at round 127:
 forward R=127: k1=0, k2=1  1 1 1 1 1 1 1 1 0 0 0 1 0 1 0   1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
backward R=127: k1=0, k2=0  0 1 0 1 0 1 1 0 0 0 1 0 0 1 0 0   1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0
```

As can be seen, at three positions in the state both in the forward and in the backward part no changes happen, which is the property needed for the matching part.

5 Discussion and Future Work

The recently proposed lightweight block cipher KTANTAN is susceptible to a class of meet-in-the-middle attacks that seems to put less constraints on the selection of key bits than some earlier meet-in-the-middle attacks

on block ciphers. We proposed key-recovery attacks with an extremely low number of known plaintexts. The approach we describe is inspired by recent advances in MITM preimage attacks on hash functions like those that succeeded in breaking MD5 or Tiger [16, 27], even though it remains an open problem to transfer most of the techniques there from the keyless hash setting to the cipher setting. The MITM approach may be seen as a way to turn very strong related-key properties into attacks in the single-key setting, complementing e.g. the work on the self-synchronized stream-cipher Moustique [21]. Even though the time complexity of our attack remains high, optimizations may result in reduced time complexities, by e.g. allowing the attacker to choose the plaintext instead (possible in many protocols), or asking for more plaintext/ciphertext pairs. Also, implementation techniques that speed-up brute force search, such as determining a good sequence of keys to guess and save computations that way, are likely to carry over to the meet-in-the-middle attack.

Among the ciphers most vulnerable to the meet-in-the-middle attacks, are those with little key-dependency in the sense that large parts of the cipher depend on a subset of key bits only. This is opposed to substitution-permutation networks which usually use subkeys of the block length in each round. For such ciphers, it is often difficult to mount a meet-in-the-middle attack even on a small number of rounds because of the strong key dependency. However, this approach as a rule results in a much higher number of XOR-operations needed for the key addition which in turn leads to higher area and/or time requirements and, thus, to a lower efficiency. An optimal trade-off between the level of resistance and the amount of key dependency remains, however, an area of research.

Acknowledgements. Andrey Bogdanov was supported in part by a visiting postdoctoral fellow grant from the Fund for Scientific Research - Flanders (FWO) within the FWO research project "Linear codes and cryptography" G.0317.06. This work was also sponsored by the Research Fund K.U.Leuven grant (OT/08/027) "A mathematical theory for the design of symmetric primitives", by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the European Commission under contract ICT-2007-216646 (ECRYPT II).

The authors are grateful to the designers of KATAN/KTANTAN for clarifying the issues with the KTANTAN key schedule and would like to thank the anonymous reviewers of SAC 2010 whose insightful comments improved the presentation of the paper.

References

1. Bit-sliced reference code of KATAN and KTANTAN. Available from http://www.cs.technion.ac.il/~orrd/KATAN/katan.c (2010)
2. Albrecht, M., Cid, C., Dullien, T., Faugre, J.C., Perret, L.: Algebraic Precomputations in Differential Cryptanalysis. ECRYPT Tools for Cryptanalysis Workshop 2010 (2010)
3. Babbage, S., Dodd, M.: The MICKEY Stream Ciphers. In: Robshaw and Billet [26], pp. 191–209
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer (2007)
5. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash Functions and RFID Tags: Mind the Gap. In: Oswald, E., Rohatgi, P. (eds.) CHES. Lecture Notes in Computer Science, vol. 5154, pp. 283–299. Springer (2008)
6. Bogdanov, A., Rechberger, C.: Generalized Meet-in-the-Middle Attacks: Cryptanalysis of the Lightweight Block Cipher KTANTAN. Preproceedings of SAC 2010 (2010)
7. Chaum, D., Evertse, J.H.: Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In: Williams, H.C. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 218, pp. 192–211. Springer (1985)
8. De Cannière, C.: Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. Lecture Notes in Computer Science, vol. 4176, pp. 171–186. Springer (2006)
9. De Cannière, C., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES. Lecture Notes in Computer Science, vol. 5747, pp. 272–288. Springer (2009)
10. De Cannière, C., Preneel, B.: Trivium. In: Robshaw and Billet [26], pp. 244–266
11. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In: Nyberg, K. (ed.) FSE. Lecture Notes in Computer Science, vol. 5086, pp. 116–126. Springer (2008)
12. Demirci, H., Taskin, I., Çoban, M., Baysal, A.: Improved Meet-in-the-Middle Attacks on AES. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 5922, pp. 144–156. Springer (2009)
13. Diffie, W., Hellman, M.: Exhaustive Cryptanalysis of the NBS Data Encryption standard. Computer 10(6), 74–84 (1977), IEEE Computer Society Press
14. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-round AES. Cryptology ePrint Archive, Report 2010/322 (2010), http://eprint.iacr.org/
15. Dunkelman, O., Sekar, G., Preneel, B.: Improved Meet-in-the-Middle Attacks on Reduced-Round DES. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 4859, pp. 86–100. Springer (2007)
16. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. Cryptology ePrint Archive, Report 2010/016 (2010), http://eprint.iacr.org/

17. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain Family of Stream Ciphers. In: Robshaw and Billet [26], pp. 179–190
18. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. IJWMC 2(1), 86–93 (2007)
19. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES. Lecture Notes in Computer Science, vol. 4249, pp. 46–59. Springer (2006)
20. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A Practical Attack on KeeLoq. In: Smart, N.P. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4965, pp. 1–18. Springer (2008)
21. Käsper, E., Rijmen, V., Bjørstad, T.E., Rechberger, C., Robshaw, M.J.B., Sekar, G.: Correlated Keystreams in Moustique. In: Vaudenay, S. (ed.) AFRICACRYPT. Lecture Notes in Computer Science, vol. 5023, pp. 246–257. Springer (2008)
22. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 4593, pp. 196–210. Springer (2007)
23. Lim, C.H., Korkishko, T.: mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J., Kwon, T., Yung, M. (eds.) WISA. Lecture Notes in Computer Science, vol. 3786, pp. 243–258. Springer (2005)
24. Merkle, R.C., Hellman, M.E.: On the Security of Multiple Encryption. Commun. ACM 24(7), 465–467 (1981)
25. van Oorschot, P.C., Wiener, M.J.: A Known Plaintext Attack on Two-Key Triple Encryption. In: EUROCRYPT. pp. 318–325 (1990)
26. Robshaw, M.J.B., Billet, O. (eds.): New Stream Cipher Designs - The eSTREAM Finalists, Lecture Notes in Computer Science, vol. 4986. Springer (2008)
27. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 5479, pp. 134–152. Springer (2009)

A More Details on Partial Matching

A.1 Block Size 48 Bits

The details for the partial matching phase for the attack on KTANTAN with $b = 48$ are as follows:

```
10 bits match as follows

forward part:
forward R=111: k1=1, k2=1  000000000000000000   000000000000000000000000000000
forward R=112: k1=0, k2=0  110000000000000000   110000000000000000000000000000
forward R=113: k1=0, k2=0  001100000000000000   001100000000000000000000000000
forward R=114: k1=0, k2=0  000011000000000000   000011000000000000000000000000
forward R=115: k1=0, k2=0  100000110000000000   100000110000000000000000000000
forward R=116: k1=0, k2=0  011000001100000000   111000001100000000000000000000
forward R=117: k1=0, k2=0  000110000110000000   001110000011000000000000000000
forward R=118: k1=0, k2=0  000001100000110000   100011100000110000000000000000
forward R=119: k1=0, k2=0  110000011000011000   111000111000001100000000000000
forward R=120: k1=0, k2=0  111100000110000110   111110001110000011000000000000
forward R=121: k1=0, k2=0  001111000011000001   101111100011100000110000000000
forward R=122: k1=0, k2=0  110011110000110000   111011111000111000011000000000
forward R=123: k1=1, k2=0  111100111100001100   111110111111000111000001100000

backward part:
backward R=131: k1=0, k2=1  000000000000000000  000000000000000000000000000000
backward R=130: k1=1, k2=1  000000000000000000  000001100000111100111100000011
```

```
backward R=129: k1=0, k2=0  0000011100011011011  00011110001111111111110001111
backward R=128: k1=0, k2=0  0001110001101101100  01111000111111111111000111100
backward R=127: k1=0, k2=0  0111000110110110000  11100011111111111100011110000
backward R=126: k1=0, k2=0  1100011111011011011  10001111111111111000111100000
backward R=125: k1=0, k2=0  0001111101111111111  00111111111111110111110000011
backward R=124: k1=0, k2=0  0111110111111111100  11111111111111011111000001100
backward R=123: k1=0, k2=0  1111011111111111011  11111111111101111100000110000

The highest number of matching bits is obtained at round 123:
 forward R=123: k1=1, k2=0  1111001111000001100  11111011111000111000001100000
backward R=123: k1=0, k2=0  1111011111111111011  11111111111101111100000110000
```

A.2 Block Size 64 Bits

The details for the partial matching phase for the attack on KTANTAN with $b = 64$ are as follows:

```
47 bits match as follows

forward part:
forward R=123: k1=1, k2=0  00000000000000000000000000  00000000000000000000000000000000000000
forward R=124: k1=0, k2=0  00000000000000000000000000  11100000000000000000000000000000000000
forward R=125: k1=0, k2=0  00000000000000000000000000  00011100000000000000000000000000000000
forward R=126: k1=0, k2=0  00000000000000000000000000  00000011100000000000000000000000000000
forward R=127: k1=0, k2=1  11000000000000000000000000  00000000011100000000000000000000000000
forward R=128: k1=0, k2=0  11111000000000000000000000  00000000000011100000000000000000000000
forward R=129: k1=0, k2=0  11111111000000000000000000  00000000000000011100000000000000000000
forward R=130: k1=0, k2=0  00011111111000000000000000  10000000000000000011100000000000000000
forward R=131: k1=0, k2=0  11000011111111000000000000  11110000000000000000011100000000000000

backward part:
backward R=131: k1=0, k2=1  00000000000000000000000000  00000000000000000000000000000000000000

The highest number of matching bits is obtained at round 131:
 forward R=131: k1=0, k2=0  11000011111111000000000000  11110000000000000000011100000000000000
backward R=131: k1=0, k2=1  00000000000000000000000000  00000000000000000000000000000000000000
```