Practical Key Recovery Attack on MANTIS_5

Christoph Dobraunig, Maria Eichlseder, Daniel Kales, and Florian Mendel

Graz University of Technology, Austria
`maria.eichlseder@iaik.tugraz.at`

Abstract. MANTIS is a lightweight tweakable block cipher recently published at CRYPTO 2016. In addition to the full 14-round version, MANTIS_7, the designers also propose an aggressive 10-round version, MANTIS_5. The security claim for MANTIS_5 is resistance against "practical attacks", defined as related-tweak attacks with data complexity 2^d less than 2^{30} chosen plaintexts (or 2^{40} known plaintexts), and computational complexity at most 2^{126-d}.
We present a key-recovery attack against MANTIS_5 with 2^{28} chosen plaintexts and a computational complexity of about 2^{38} block cipher calls, which violates this claim. Our attack is based on a family of differential characteristics and exploits several properties of the lightweight round function and tweakey schedule. To verify the validity of the attack, we also provide a practical implementation which recovers the full key in about 1 core hour using 2^{30} chosen plaintexts.

Keywords: Cryptanalysis · MANTIS · PRINCE-like ciphers

1 Introduction

MANTIS is a tweakable block cipher recently published at CRYPTO 2016 by Beierle et al. [2]. The designers' goal is to optimize this versatile building block for low-latency implementations. To this end, they use the same α-reflective structure as PRINCE by Borghoff et al. [3], but combine it with the round function of Midori by Banik et al. [1]. According to their analysis [2], this improves both the latency and the security compared to the original PRINCE, since Midori's variant of ShiftRows leads to a higher bound on the minimum number of active S-boxes. The tweak is incorporated using an adapted version of the TWEAKEY framework by Jean et al. [4].

The full version MANTIS_7 has 14 rounds, but the authors also give a reduced security claim for the 10-round version, MANTIS_5. They claim security against practical attacks, which they define as related-tweak attacks with data complexity 2^d less than 2^{30} chosen plaintexts (or 2^{40} known plaintexts), and computational complexity at most 2^{126-d} block cipher calls, similar to the PRINCE challenge. We present a key-recovery attack against MANTIS_5 with 2^{28} chosen plaintexts and a computational complexity of about 2^{38} block cipher calls, which violates this claim.

Our attack exploits the lightweight near-MDS mixing layer and certain differential properties of the involutive S-box, both inherited from Midori. These properties make it relatively easy to find a differential characteristic with the claimed optimal probability in the related-tweak setting. Using the same properties, this differential characteristic can then be expanded to a family of characteristics with a corresponding initial structure that makes efficient use of the low data complexity limit of only 2^{30} chosen plaintexts. Furthermore, the choice to keep the original Midori order of linear operations (first permute, then mix) makes the PRINCE-like middle rounds differentially less effective than the ordering used by PRINCE (first mix, then permute). Midori's order preserves a Superbox structure over 4 S-box layers in the middle rounds, instead of 2.

We verified the validity of the attack in a practical implementation. The implementation revealed an additional differential property of the Midori S-box that complicates some steps of the attack due to differentially equivalent keys. An adapted version of the attack recovers the full key in about 1 core hour using 2^{30} chosen plaintexts.

Outline. In section 2, we provide a brief description of the tweakable block cipher MANTIS and some of its cryptographic properties. In section 3, we introduce a family of differential characteristics and a corresponding initial structure of messages for MANTIS_5 that lead to a good filter after 9 rounds. In section 4, we use this initial structure and filter to mount a key recovery attack on MANTIS_5. Finally, we discuss the results of a practical implementation of the attack.

2 Description of MANTIS

2.1 The Tweakable Block Cipher

MANTIS is a tweakable block cipher published at CRYPTO 2016 by Beierle et al. [2]. The designers propose several variants MANTIS_r that differ only in the number of rounds. All variants operate on a 64-bit message block $M = M_0\|M_1\|\cdots\|M_{15}$ and work with a 64-bit tweak $T = T_0\|T_1\|\cdots\|T_{15}$ and $(64+64)$-bit key $K = (k_0, k_1)$. All 64-bit values are mapped to 4×4 states S of 4-bit cells S_j:

$$S = \begin{array}{|c|c|c|c|}\hline S_0 & S_1 & S_2 & S_3 \\ \hline S_4 & S_5 & S_6 & S_7 \\ \hline S_8 & S_9 & S_{10} & S_{11} \\ \hline S_{12} & S_{13} & S_{14} & S_{15} \\ \hline \end{array}\,.$$

The cipher's structure is similar to PRINCE, with r forward rounds $\mathcal{R}_i$ and r backward rounds $\mathcal{R}_{2r+1-i} = \mathcal{R}_i^{-1}$, separated by an involutive, unkeyed middle layer $\mathsf{S}\circ\mathsf{M}\circ\mathsf{S}$. The 64-bit subkey k_1 is used as round key for the outer forward and backward rounds, while the other 64-bit subkey k_0 and the derived $k_0' = (k_0 \ggg 1) + (k_0 \gg 63)$ serve as whitening keys. The tweak T is added together with

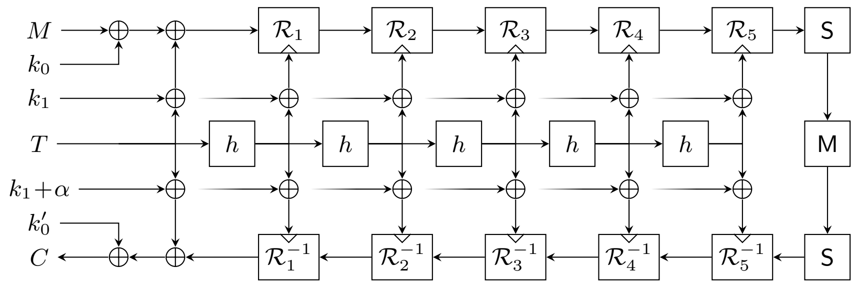


Fig. 1: PRINCE-like structure of MANTIS$_r$, illustrated for MANTIS$_5$.

k_1 in every round according to the TWEAKEY construction, with a simple cell permutation h as a tweak schedule. The construction is illustrated in Figure 1.

2.2 The Round Functions $\mathcal{R}_i$ and $\mathcal{R}_i^{-1}$

The round function $\mathcal{R}_i$ is very closely related to that of Midori [1]. It updates the 4×4 state of 4-bit cells by means of the sequences of transformations

$$\mathcal{R}_i = \mathsf{MixColumns} \circ \mathsf{PermuteCells} \circ \mathsf{AddTweakey}_i \circ \mathsf{AddConstant}_i \circ \mathsf{SubCells},$$
$$\mathcal{R}_i^{-1} = \mathsf{SubCells} \circ \mathsf{AddConstant}_i \circ \mathsf{AddTweakey}_i \circ \mathsf{PermuteCells}^{-1} \circ \mathsf{MixColumns},$$

as illustrated in Figure 2. In the following, we briefly describe the individual operations. For a more detailed description of the MANTIS family, we refer to the design paper [2].

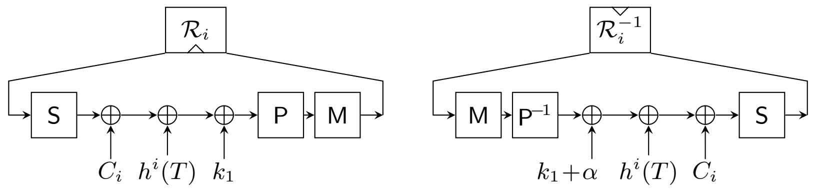


Fig. 2: The MANTIS round functions $\mathcal{R}_i$ and $\mathcal{R}_i^{-1}$.

SubCells (S). The involutive 4-bit S-box $\mathcal{S}$ given in Figure 3 is applied to each cell of the state. For our attack, we are primarily interested in the differential behaviour of $\mathcal{S}$, which is illustrated in Figure 5a.

AddTweakey$_i$ (A) and AddConstant$_i$ (C). Several round-dependent values are added to the state: The round constant C_i, the subkey k_1 (for $\mathcal{R}_i$) or $k_1 + \alpha$ (for $\mathcal{R}_i^{-1}$), and the round tweakey $h^i(T)$. The tweakey update function h simply permutes the order of cells using the permutation h, specified in Figure 4a.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\mathcal{S}(x)$	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

Fig. 3: The MANTIS S-box $\mathcal{S}$, borrowed from Midori [1].

h:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

$\xrightarrow{h}$

6	5	14	15
0	1	2	3
7	12	13	4
8	9	10	11

(a) The tweak update function h.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

$\xrightarrow{\mathsf{P}}$

0	11	6	13
10	1	12	7
5	14	3	8
15	4	9	2

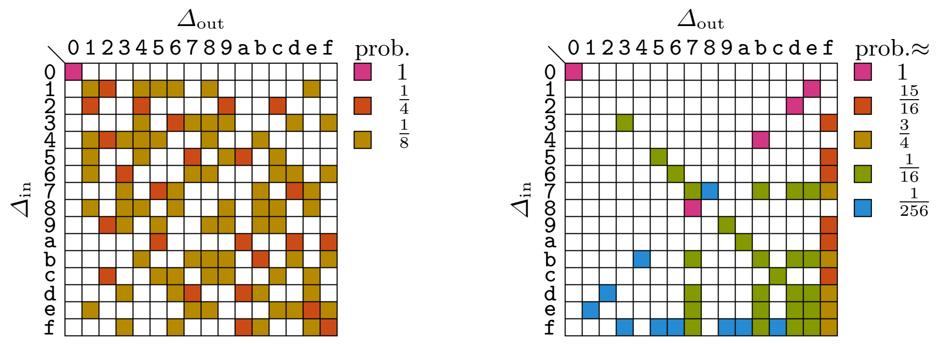
(b) The state transformation P.

Fig. 4: The MANTIS permutations h and P.

PermuteCells (P). The cells of the state are permuted by P, specified in Figure 4b.

MixColumns (M). Each column of the state is multiplied with the following involutive near-MDS matrix M over $\mathbb{F}_{2^4}$, whose truncated differential behaviour per column is illustrated in Figure 5b:

$$\mathsf{M} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} .$$



(a) DDT of SubCells. (b) Truncated DDT of MixColumns.

Fig. 5: Differential distribution tables (DDT) of the MANTIS round operations.

3 Differential Characteristic

3.1 Bounds and Security Claim

The designers of MANTIS analyze the security of the cipher against differential cryptanalysis by modelling the differential behaviour (truncated to state cells) as a mixed-integer linear program [2]. They analyzed the minimum number of active S-boxes for different round numbers, both in a fixed-tweak and a related-tweak setting. The design document provides lower bounds for full and round-reduced MANTIS.

For MANTIS_5, the minimum number of active S-boxes in the related-tweak setting is 34 (for the full MANTIS_7: 50), and the maximum differential probability of the S-box is 2^{-2}. The designers conclude that "no related tweak linear or differential distinguisher based on a characteristics is possible for MANTIS_5" [2]. In particular, they claim that MANTIS_5 is secure against "practical attacks", here defined as related-tweak attacks with data complexity 2^d at most 2^{30} chosen plaintexts (or 2^{40} known plaintexts), and computational complexity at most 2^{126-d}.

3.2 A Family of Differential Characteristics

Our attack is based on a truncated differential characteristic for the related-tweak setting that meets this lower bound of 34 active S-boxes. The truncated characteristic is illustrated in Figure 6. Instead of considering only a single fixed input difference and differential characteristic, we will cluster several related differential characteristics following the same truncated differential characteristic, thus obtaining a much better probability.

An Optimal Differential Characteristic. To analyze the probability, we first construct a differential characteristic that matches the claimed optimal differential probability of $2^{-34\cdot 2} = 2^{-68}$. Consider the differential distribution table of SubCells, given in Figure 5a. Observe that SubCells is an involution, so the table is symmetric. There is one input/output difference, $\mathtt{a}$, such that all transitions from or to difference $\mathtt{a}$ have the maximum probability of $\frac{1}{4}$. Furthermore, these possible transitions include $\mathtt{a} \mapsto \mathtt{a}$. Since MixColumns only has binary coefficients, all transitions that match the branch number of 4 for MixColumns ($1 \to 3$, $2 \to 2$, $3 \to 1$) are valid when all active cells have a fixed difference of $\mathtt{a}$.

Since all non-trivial MixColumns transitions of the truncated differential characteristic in Figure 6 match its branch number, setting all active cells to $\mathtt{a}$ results in a valid differential characteristic with the claimed optimal probability of $2^{-34\cdot 2} = 2^{-68}$.

Clustering Differential Characteristics. We will now relax some of these constraints, and also consider characteristics with cell differences other than $\mathtt{a}$ in selected sections of the characteristic. Interesting candidates include all differences that can be mapped from and to $\mathtt{a}$ by SubCells, that is, $\{\mathtt{5}, \mathtt{a}, \mathtt{d}, \mathtt{f}\}$.

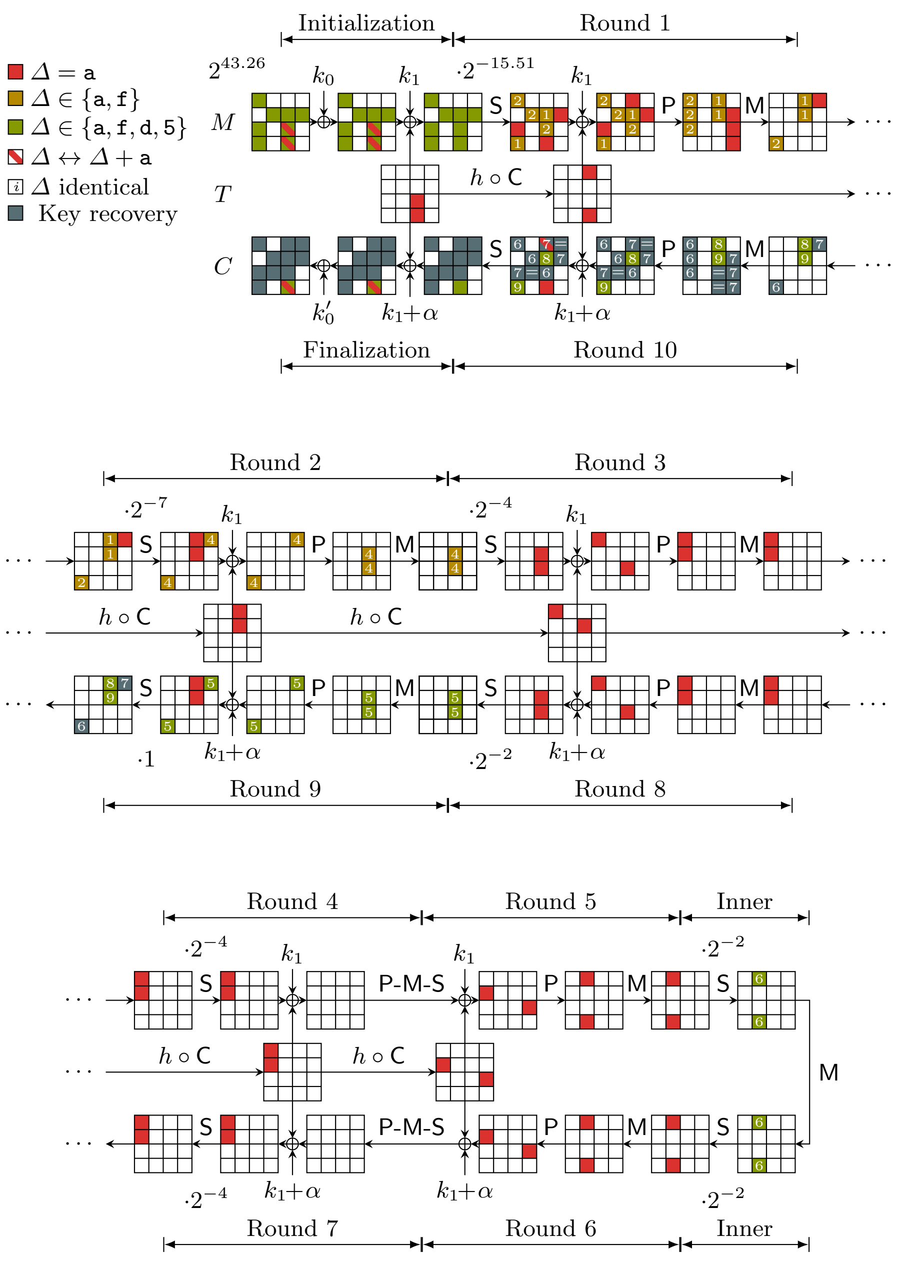


Fig. 6: Family of differential characteristics for MANTIS_5.

Rounds 9 and 2. First, consider Round 9. The SubCells layer at the end of Round 8 has 2 active S-boxes, at positions S_6 and S_{10}. Assume we allow all possible output differences $\{\mathtt{5}, \mathtt{a}, \mathtt{d}, \mathtt{f}\}$ for the two S-boxes, marked $\boxed{5}$ in Figure 6. Then, the characteristic will follow the same truncated differential, with the same probability of $2^{-4\cdot 2}$ to transition to the all-$\mathtt{a}$ state at the end of Round 9, as long as both S-boxes map to the same difference. The probability for this is 2^{-2}, instead of the original 2^{-4} of the all-$\mathtt{a}$ differential characteristic.

A similar observation applies for the two S-boxes S_3 and S_{12} of Round 2, marked $\boxed{4}$ in Figure 6. However, as we want to relax also the input differences to Round 2, we will consider only output differences $\{\mathtt{a}, \mathtt{f}\}$. These have the additional advantage of allowing transitions with probability 2^{-2} not only to $\mathtt{a}$, but each to both $\mathtt{a}$ and $\mathtt{f}$, so this relaxation can be used in multiple consecutive rounds. The probability for Round 2 improves from 2^{-8} to $2^{-2\cdot 2}\cdot 2^{-1}\cdot 2^{-2} = 2^{-7}$.

Inner Part. Second, consider the inner part. Similar as for Round 9, we can allow all 4 output differences for the first SubCells operation of the inner part, as long as both S-boxes map to the same difference, marked $\boxed{6}$ in Figure 6. This seems to improve the probability for the inner part from $2^{-4}\cdot 2^{-4}$ to $2^{-2}\cdot 2^{-4}$. However, note that there is no tweakey addition between the two SubCells layers of the inner part, so the probabilities for the S-box transitions are certainly not independent. Since there is also no PermuteCells operation, we can simply compute the exact Superbox transition probability for the entire second column of the state. This reveals that the probability for the inner part is in fact 2^{-4}.

Initialization and Round 1. Like Round 2, we relax some of the differences of Round 1 to $\{\mathtt{a}, \mathtt{f}\}$. The estimated probability for Round 2 will remain valid for the output cells cells of Round 1 ($\boxed{1}$, $\boxed{2}$, 🟥). Again, MixColumns adds several constraints for the output differences of the SubCells layer of Round 1.

Finally, we relax the input differences. In addition to $\{\mathtt{a}, \mathtt{f}\}$, we also allow $\{\mathtt{5}, \mathtt{d}\}$ in order to generate more message pairs, while retaining a reasonable differential probability. For message cells S_{10} and S_{14}, marked 🟥 in Figure 6, we need to compensate the AddTweakey operation of the initialization part by considering input differences Δ such that $\Delta + \mathtt{a} \in \{\mathtt{a}, \mathtt{f}, \mathtt{5}, \mathtt{d}\}$, or equivalently, $\Delta \in \{\mathtt{0}, \mathtt{5}, \mathtt{f}, \mathtt{7}\}$. The probability for the SubCells layer of Round 1, assuming uniformly distributed input differences, is then

$$\underbrace{2^{-3\cdot 2}}_{\substack{■\to■\\ ■\to■\\ ■\to■}} \cdot \underbrace{\left(\frac{1}{4}\cdot 2^{-3} + \frac{3}{4}\cdot 2^{-4}\right)}_{■,■\to\boxed{1},\boxed{1}} \cdot \underbrace{\left(\frac{1}{8}\cdot 2^{-5} + \frac{7}{8}\cdot 2^{-6}\right)}_{■,■,■\to\boxed{2},\boxed{2},\boxed{2}} \approx 2^{-15.51}.$$

Consequently, the overall probability of the family of characteristics up to Round 9 (or more precisely, up to AddTweakey of Round 10) is at least about

$$2^{-15.51-7-4-4-2-2-4-2} = 2^{-40.51}.$$

Round 10. If a pair followed the family of characteristics up to Round 9, the output of the AddTweakey operation of Round 10 will have several properties that can be used as a filter for key recovery.

– Cells $S_1, S_4, S_{11}, S_{13}, S_{15}$ have zero difference, which will also be immediately visible in the ciphertexts (though not useful for key recovery).
– Cell S_{14} (marked 🟥) has difference a (2-bit filter).
– Cells S_0, S_5, S_{10} (marked ▨) will have the same difference (8-bit filter), as will cells S_2, S_7, S_8 (marked ▨) after compensating for the tweak difference (8-bit filter).
– Cells S_6 and S_{12} (marked 🟩, 🟩) will have differences $\{\mathtt{a}, \mathtt{f}, \mathtt{5}, \mathtt{d}\}$, and additionally, due to the properties of MixColumns, cells S_3 and S_9 (marked ▤) will have the same difference, which is the sum of the differences of S_6, S_{12} (12-bit filter).

Overall, the family of characteristics provides a 30-bit filter with probability $2^{-40.51}$.

3.3 Initial Structure

We now want to generate enough message pairs to expect at least one valid pair, while staying well below the data complexity limit of 2^{30} chosen plaintexts. Obviously, the characteristic's probability is not good enough for a straightforward solution with 2^{29} suitable pairs. However, we can use the set $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$ of valid differences for each cell to our advantage.

We repeat the following for two random base plaintext-tweak pairs. For each of the two plaintext-tweak pairs, we query two sets of derived plaintext-tweak pairs: one for the base tweak, and one for the modified tweak with a difference of a in two cells, as specified by the truncated differential characteristic in Figure 6. The first set for the base tweak contains the following 8^8 modified messages. Each of the 8 active cells (🟩, ◩) varies over 8 values: the base plaintext plus differences $\{\mathtt{0}, \mathtt{a}, \mathtt{f}, \mathtt{5}, \mathtt{d}, \mathtt{8}, \mathtt{7}, \mathtt{2}\}$. The second set for the modified tweak contains the same 8^8 messages. In total, the number of chosen plaintext-tweak pairs we query is

$$2 \cdot 2 \cdot 8^8 = 2^{26}.$$

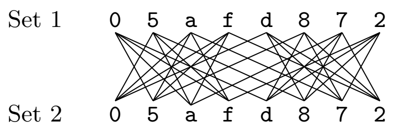
Thus, we could repeat this up to $2^4 = 16$ times and still stay below the data complexity limit.

To see how many suitable pairs we can generate from these queries, note that for each value of a cell in the first set, there are exactly 4 (out of 8) values for this cell in the second set that give a valid difference $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$ (🟩) or $\{\mathtt{0}, \mathtt{5}, \mathtt{7}, \mathtt{f}\}$ (◩), as illustrated in Figure 7. Here, we exploited that $\mathtt{a} + \mathtt{5} = \mathtt{f}$, where all these three values are suitable for our family of characteristics. Thus, the number of pairs we get is

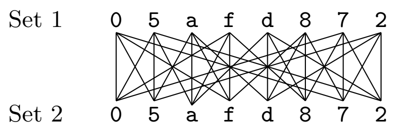
$$2 \cdot 8^8 \cdot 4^8 = 2^{41},$$

and the expected number of valid pairs is at least

$$2^{41} \cdot 2^{-40.51} = 2^{0.49} \approx 1.40\,.$$



(a) Differences $\{\mathtt{a},\mathtt{f},\mathtt{d},\mathtt{5}\}$ (🟩).



(b) Differences $\{\mathtt{0},\mathtt{5},\mathtt{7},\mathtt{f}\}$ (▨).

Fig. 7: Initial structure with $8 \cdot 4$ pairs from $2 \cdot 8$ queries per cell.

By repeating this up to 2^4 times, we can increase the expected number of valid pairs up to

$$2^{4.49} \approx 22.47\,.$$

We evaluated the initial structure practically for 1024 random keys, and found that the average number of valid pairs is significantly higher than the estimated 22.47, around $2^{6.28} \approx 78$.

4 Key Recovery

We can now use the family of characteristics and initial structure from section 3 to recover the two 64-bit secret keys k_0 and k_1. In the following, we will use 4 repetitions $r = 1, \ldots, 4$ of the initial structure. Thus, we need to query $4 \cdot 2^{26} = 2^{28}$ chosen plaintexts with chosen tweaks in order to generate the $4 \cdot 2^{41} = 2^{43}$ plaintext pairs. This is well below the complexity limit of 2^{30} chosen plaintexts for MANTIS_5.

4.1 Pre-Filtering Ciphertexts for Wrong Pairs

Before starting with the key guessing, we can filter for pairs which definitely do not follow the family of characteristics given in Figure 6. The necessary conditions for valid ciphertext pairs are that 5 cells $(S_1, S_4, S_{11}, S_{13}, S_{15})$ have a zero difference (marked □), while the difference in cell S_{14} is in $\{\mathtt{a},\mathtt{f},\mathtt{d},\mathtt{5}\}$ after removing the last tweak addition (marked ▨). The reason for the restriction of the differences in cell S_{14} lies in the tweak addition in this cell before the last S-box application.

If we assume that plaintext pairs which do not follow our family of characteristics produce a randomly distributed difference pattern for corresponding ciphertext pairs, these conditions are fulfilled with a probability of 2^{-22}. Hence, we reduce the set of 2^{41} pairs per repetition r from the initial structure to a set I_r of about $2^{41-22} = 2^{19}$ pairs. Each set I_r is still expected to contain $2^{0.49} > 1$ valid pairs that follow the family of characteristics of Figure 6.

Complexity and Optimizations. A naive implementation of generating and pre-filtering pairs costs $4 \cdot 2^{41}$ state xor operations. However, instead of enumerating all valid pairs and then filtering for matches on 5 cells, it is much more efficient to reverse the process and only generate the relevant pairs as follows. Store each plaintext-tweak-ciphertext of Set 1 in a data structure of 2^{20} partitions, partitioned according to the value of the 5 pre-filter cells $S_1, S_4, S_{11}, S_{13}, S_{15}$. The expected size of each partition is about 2^5. Then, for each plaintext-tweak-ciphertext of Set 2, iterate only over the 2^5 candidates in the correct partition, and check whether the input difference is valid and the difference of output cell S_{14} is valid. The set I_r of remaining filtered pairs is the same, but the computational complexity is reduced to less than 2^{30} state xor operations.

4.2 Recovery of 44-bit $k'_0 + k_1$

The first step of the attack is the partial recovery of 44 bits of the final whitening key $k'_0 + k_1$. We want to check our key guesses against the differential pattern we get before the last application of MixColumns in Round 10 for our filtered ciphertext pairs. The probability that a 44-bit key guess leads to this pattern before the application of MixColumns is 2^{-30}:

- **Column 1:** Here, only cell S_{12} has a difference at the input of MixColumns, while the others have none. The requirements that lead to this pattern are that a key guess on the ciphertext cells S_0, S_5, S_{10} (■) leads to an equal difference after an S-box application, which happens with a probability of 2^{-8} per ciphertext pair and key guess.
- **Column 2:** This column is inactive. The only condition we have to fulfill here is that the difference introduced in cell S_{14} (🟥) of the ciphertext is canceled by the tweak addition that happens before the S-box application of the last round (right after the last application of PermuteCells). Since our filtering ensures that only ciphertext pairs with differences $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$ in cell S_{14} (🟩) after the last SubCells are considered, this happens with a probability of 2^{-2}.
- **Column 3:** For this column, cells S_2 (🟩) and S_6 (🟩) must have a difference $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$, while cells S_{10}, S_{14} have zero difference (□). The necessary conditions for this to happen are that a key guess on cells S_3, S_6, S_9, S_{12} of the ciphertext pair leads to an input difference $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$ on cells S_6, S_{12} (🟩, 🟩) before the last SubCells (2^{-2} per cell), and that the differences in S_3, S_9 (▤) each equal the difference between S_6 and S_{12} (2^{-4} per cell). The overall probability for this is 2^{-12} per ciphertext pair and key guess.
- **Column 4:** For this column, the same reasoning as for column 1 applies, now for ciphertext and key cells S_2, S_7, S_8 (■) after compensating for the last tweak addition. Again, the probability is 2^{-8}.

If we now decrypt one ciphertext pair $i \in I_r$ backwards for one SubCells layer under $2^{11 \cdot 4} = 2^{44}$ key guesses, $2^{44-30} = 2^{14}$ key guesses remain which satisfy all these conditions for this ciphertext pair i. We expect the correct key guess to satisfy the conditions for at least one of the ciphertext pairs $i \in I_r$, which follows the family of characteristics in Figure 6. Thus, we repeat the procedure for all 2^{19} pairs and consider the union of all resulting potential key candidates. We expect at most $2^{14} \cdot 2^{19} = 2^{33}$ candidates for the right key guess, which effectively reduces our keyspace by 2^{-11}. So, repeating the attack a total of 4 times with fresh initial structures is sufficient to recover the correct value of 44 bits of $k_0' + k_1$.

Complexity and Optimizations. To get the possible key candidates per ciphertext pair, we need $2 \cdot (2^{16} \cdot 4 + 2 \cdot 2^{12} \cdot 3 + 2^4) \approx 2^{19.13}$ S-box look-ups, which corresponds roughly to $2^{11.54}$ MANTIS_5 encryptions (based on the total number of $16 \cdot 12$ S-boxes in MANTIS_5). In total, we have to generate key candidates for $4 \cdot 2^{19}$ pairs, corresponding to a total of about $2^{32.54}$ MANTIS_5 encryptions.

In a straightforward implementation, we get 4 lists, each containing 2^{33} key candidates, which dominates our memory requirements. We need to find matches between the 4 lists, which adds a computational complexity of roughly 2^{33} operations, depending on the implementation.

Note that it is not actually necessary to guess all 44 bit of the subkey at once per ciphertext pair $i \in I_r$. Instead, we can split up the key guesses column-wise into a 12-bit subkey for column 1 (with a set of valid subkey candidates of expected size $|\mathcal{C}_{0,5,10}^{(r,i)}| = 2^4$), a 4-bit subkey for column 2 ($|\mathcal{C}_{14}^{(r,i)}| = 2^2$), a 16-bit subkey for column 3 ($|\mathcal{C}_{3,6,9,12}^{(r,i)}| = 2^4$), and a 12-bit subkey for column 4 ($|\mathcal{C}_{2,7,8}^{(r,i)}| = 2^4$). The expected set of 2^{14} full key candidates per pair i is then the product set of these sub-candidates. We refer to this structured set of key candidates from repetition r and pair $i \in I_r$ as a bundle $\mathcal{B}^{(r,i)}$, where

$$\mathcal{B}^{(r,i)} = \mathcal{C}_{0,5,10}^{(r,i)} \times \mathcal{C}_{14}^{(r,i)} \times \mathcal{C}_{3,6,9,12}^{(r,i)} \times \mathcal{C}_{2,7,8}^{(r,i)}.$$

Storing all bundles requires only about $4 \cdot 2^{19} \cdot 10.25 < 2^{25}$ MANTIS states. To find the correct value of all 44 bits, we now need to compute

$$\bigcap_{r=1}^{4} \bigcup_{\substack{i \in I_r \\ |I_r| \approx 2^{19}}} \mathcal{C}_{0,5,10}^{(r,i)} \times \mathcal{C}_{14}^{(r,i)} \times \mathcal{C}_{3,6,9,12}^{(r,i)} \times \mathcal{C}_{2,7,8}^{(r,i)}.$$

The computational complexity of matching the bundles of key candidates is similar to before if the list of bundles per repetition is indexed efficiently per subkey candidate. Then, the bundles can be intersected subkey by subkey, starting with the most restrictive subkey, $\mathcal{C}_{3,6,9,12}^{(r,i)}$.

4.3 Recovery of 32-bit $k_0 + k_1$

With the help of the recovered 44 bits of $k_0' + k_1$, we can filter our plaintext pairs $i \in I_r$ so that only the valid plaintext pairs following the family of characteristics in Figure 6 remain. The probability that the right key identifies a wrong pair as correct one is 2^{-30}. Therefore, it is likely that only correct pairs (approximately 4) remain after filtering $4 \cdot 2^{19}$ pairs. We now use those 4 valid pairs to recover 32 bits of the initial whitening key $k_0 + k_1$. We guess the key bits for all plaintext cells with differences, $S_0, S_5, S_6, S_7, S_8, S_{10}, S_{12}, S_{14}$. Then we can compute forward through the SubCells layer of Round 1, and check if the resulting difference pattern matches the family of characteristics. As shown in Figure 6, a wrong key matches the pattern with a probability of $2^{-15.51}$. So, the probability that a wrong key matches for all 4 correct pairs is $2^{-62.04}$. Therefore, we expect that only the correct subkey out of the 2^{32} possible candidates remains.

Complexity. We make a 32-bit key guess for each of 4 pairs, leading to a total of $2 \cdot 4 \cdot 8 \cdot 2^{32} = 2^{38}$ S-box look-ups. This corresponds to about $2^{30.42}$ MANTIS_5 encryptions.

4.4 Recovery of k_0 and k_1

Up to this point, we have recovered 32 bits of information about $k_0 + k_1$ and 44 bits of information about $k_0' + k_1 = (k_0 \ggg 1) + (k_0 \gg 63) + k_1$. This gives us a system of 76 linearly independent linear equations for k_0 and k_1. To recover the full key, we have to guess 52 remaining bits and identify the right key using trial encryptions.

Instead of guessing all 52 bits, we can also use the SubCells layers of Rounds 2 and 3 (or 9 and 8) to first recover more bits of k_1, based on the previously recovered information. Similar to recovering $k_0 + k_1$, we can apply a guess-and-determine approach to only the 4 valid pairs, for example:

(1) Recover $S_0 + S_5 + S_{10}$ of k_1: We target cell S_{12} at the beginning of Round 2 (transition $\boxed{2} \to \boxed{4}$ in Figure 6). From our previously recovered key bits, we know the values of cells S_0, S_5, S_{10} at the beginning of Round 1 ($\boxed{2}$). Our target cell is the sum of these known values, plus an unknown cell $S_0{+}S_5{+}S_{10}$ of k_1. Checking the correct S-box transition for all 4 valid pairs is expected to eliminate all but the correct cell value (otherwise, we can additionally check the transition $\boxed{6} \leftarrow \boxed{5}$ in Round 9). This adds 1 linearly independent equation to the system.

(2) Recover $S_6 + S_{12}$ of k_1: We target cells S_2, S_6 at the beginning of Round 2 (transitions $\boxed{1} \to$ 🟥). Each of the two is the sum of the same two unknown, constant values (cell S_3 and cell S_9 after AddTweakey of Round 1), a known, variable value, and a cell of k_1 (S_6 or S_{12}, respectively). By checking the S-box transitions and then eliminating the two unknown constants, we recover the cell sum $S_6 + S_{12}$ of k_1. This adds 1 linearly independent equation to the system.

(3) Recover $S_2{+}S_7{+}S_8$ of k_1: We target cell S_3 at the end of Round 9 (transition $\boxed{7} \leftarrow \boxed{5}$). Similar to (1), the transition depends on a sum of k_1 cells, $S_2 + S_7 + S_8$. From (1), we can derive the exact target difference in $\boxed{5}$, so the transition probability is at most 2^{-2}, and we expect only the one correct cell value to remain. This adds 4 linearly independent equations to the system.

(4) Guess 1 bit: If we guess only 1 bit of k_0 now (e.g., in cell S_{12}), this will fully determine the values of cells $S_2, S_5, S_6, S_7, S_8, S_{12}$ of k_0 and k_1.

(5) Recover S_3 of k_1: We target cells S_6 and S_{10} at the beginning of Round 3 (transitions $\boxed{4} \to$ 🟥). Due to the previous MixColumns operation, the internal

difference between cells S_6 and S_{10} is equal to the internal difference between cells S_3 and S_{12} after the previous AddTweakey operation, which is known except for the addition of key cell S_3 of k_1. On the other hand, since we require that both target cells belong to the same set of 4 possible values for a valid transition, this cuts down the possible values for S_3 of k_1 to less than half. After repeating for all 4 valid pairs and, if necessary, similarly for the transition $\boxed{5}$ $\to$ 🟥 in Round 8, we expect only the correct candidate to remain. This adds 4 linearly independent equations to the system.

(6) Recover S_9 of k_1: We target cells S_2 and S_6 at the end of Round 9 (transitions $\boxed{8}$ $\to$ 🟥 and $\boxed{9}$ $\to$ 🟥). The transition depends on the values of cells S_3, S_6, S_9, S_{12} before AddTweakey of Round 10, which are all known by now except for the addition of key cell S_9 of k_1. Determining S_9 adds another 4 linearly independent equations to the system.

Complexity. The guess-and-determine approach recovers 14 of the missing bits of the original 64-bit keys k_0 and k_1. This reduces the remaining bits that need to be guessed to 38. To complete the key, we have to compute 2^{38} trial encryptions, which dominates our attack complexity.

4.5 Practical Verification

We implemented the key recovery attack in C/C++ in order to verify the probability estimates and attack complexity. A first straightforward implementation revealed some additional structural properties of MANTIS that negatively affect the success probability of the attack. For this reason, we adapted some aspects of the attack in order to obtain a good success probability in practice.

The first issue is that while the estimated number of about 1 to 10 valid pairs per repetition appears to be a reasonable estimate on average, the variance is relatively high. We observed several repetitions with no valid pairs, while other repetitions produced a dozen or more pairs. This is a problem for the 44-bit key recovery of subsection 4.2, which relies on finding at least 1 valid pair per repetition. There are several options to compensate for this. If memory requirements and higher runtime are not an issue, we can simply expand all bundles of key candidates and count the number of occurrences of each candidate, which will reveal the correct candidate with very high probability. A more practical alternative is to change the initial structures per repetition to contain more structures for different plaintexts, but with fewer queries per structure, in order to decrease the variance. For example, if we use 2^6 base plaintexts per repetition, but vary only 7 instead of 8 cells, the resulting expected number of pairs per repetition remains the same at $2^6 \cdot 8^7 \cdot 4^7 = 2^{41}$, but the data complexity increases slightly to $2 \cdot 2^6 \cdot 8^7 = 2^{28}$, or $4 \cdot 2^{28} = 2^{30}$ in total for all repetitions.

The second issue is that during the 32-bit key recovery of subsection 4.3, we always find at least 2^8 possible key candidates instead of just 1, and 2 key candidates for the 44-bit subkey. Both this and the previous issue are caused by the same structural property of the MANTIS S-box. We filter our keys by checking whether the valid pairs follow the correct differential S-box transitions in Round 1, that is, $\{\mathtt{a},\mathtt{f},\mathtt{d},\mathtt{5}\} \mapsto \{\mathtt{a},\mathtt{f}\}$ for each cell. However, it turns out that whenever a pair of cells (x, x') follows one of these transitions, then so does $(x + \mathtt{a}, x' + \mathtt{a})$. This means that for each cell k of the correct subkey, there is an equivalent value $k + \mathtt{a}$ which also satisfies all the constraints of Round 1, leading to a total of at least 2^8 candidates. This would also increase the complexity of subsection 4.4 accordingly. Instead of the expensive brute-force approach of subsection 4.4, we encoded the recovery of the remaining key information as a Boolean satisfiability (SAT) problem.

The final adapted attack successfully recovered the full key for several tested random challenges. A sample test run takes about 16 minutes to query 2^{30} plaintexts and generate the pre-filtered list of about 4×2^{21} pairs (subsection 4.1). Creating the bundles of key guesses takes 22 minutes and produces about $4 \times 2^{14.6}$ bundles in total, corresponding to about 2^{32} key candidates per repetition if fully enumerated (subsection 4.2). Intersecting these lists takes 18 minutes and produces more key candidates than expected, about 2^7. However, counting the frequency of each of these candidates across repetitions clearly identifies the correct 44-bit final whitening subkey (except for 1 bit, due to differentially equivalent keys as discussed above, which can be filtered by the SAT solver). In the sample test run, this correct key identified 14 valid pairs, slightly less than the observed average of roughly 2^5 pairs. The high number of valid pairs means that it is relatively unlikely that one repetition contains no valid pairs, and that these cases could also usually be easily fixed by reshuffling the random plaintexts between repetitions. We also observed no false positives among the valid pairs, except for several cases with differences $\{\mathtt{d},\mathtt{5}\}$ in $S_0, S_5, S_6, S_{10}, S_{12}$ after the SubCells layer of Round 1 (1, 2). We included these cases in the family of target characteristics. For the initial whitening subkey recovery (subsection 4.3), as discussed above, the recovered information is only about $32 - 9$ bits due to differentially equivalent subkeys, and takes about 3 minutes. Finally, the SAT solver takes another 1.5 minutes to successfully recover the rest of the key (subsection 4.4). Overall, the full correct key is recovered in about 1 hour on a single core, and the process is trivially parallelizable.

5 Discussion and Conclusion

We recover the full 128-bit key for MANTIS_5 with a computational complexity of about 2^{38} encryptions, memory requirements of about 2^{25} MANTIS states, and a data complexity of 2^{28} chosen plaintexts with chosen tweaks. A practical implementation recovered the correct key in about an hour based on 2^{30} chosen plaintexts. This violates the security claim for MANTIS_5: The designers claim resistance against attacks with computational complexity less than $2^{126-30} = 2^{96}$ encryptions based on this data complexity.

We did not analyze the full-round MANTIS_7 proposal. Many of the observations and methods for MANTIS_5 also apply to MANTIS_7, which certainly casts some doubt on the design's security margin. It is relatively easy to find a very similar optimal differential characteristic with probability 2^{-100} (compared to 2^{-68} for MANTIS$_5$), and to apply the same observations for clustering characteristics and improving the probability. However, a straightforward adaptation of the full key recovery attack is made more difficult by several factors. For example, it is hard to find characteristics for MANTIS$_7$ which on the one hand have a sufficiently low number of active S-boxes, and on the other hand have enough active cells at the input and output to be useful for key recovery. Also, due to the small state size, the probability must be relatively high to avoid false positives among the seemingly valid pairs.

Our attack takes advantage of several very lightweight building blocks of MANTIS, most of them inherited from the Midori block cipher. This includes the involutive S-box with its high-probability differential fixed points `a` and `f`, the lightweight near-MDS matrix with its binary coefficients, and the lightweight tweakey schedule. Throughout the analysis, the symmetries of the PRINCE-like design facilitate the repeated exploitation of these properties. Another major issue is the interaction of the Midori-inspired round function with the PRINCE-inspired inner rounds, which leads to a Superbox structure over 4 S-box layers in the inner rounds. Considering all these properties, the security margin of MANTIS may be too optimistic.

Acknowledgements. We thank the MANTIS designers for verifying a preliminary version of our results and providing useful comments.

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. LNCS, vol. 9453, pp. 411–436. Springer (2015)
2. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer (2016)
3. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE – A low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer (2012)
4. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer (2014)