Key Recovery for MANTIS$_5$

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel

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

Abstract. MANTIS is a tweakable block cipher recently published at CRYPTO 2016. While the full version MANTIS$_7$ has 14 rounds, the authors claim security against "practical attacks" for the 10-round version, MANTIS$_5$. Here, "practical" is 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^{52} block cipher calls, which violates this claim.

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] in order to improve the latency and security. 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}, 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^{52} block cipher calls, which violates this claim. Note that the computational complexity can easily be reduced further by using a slightly more complicated key recovery approach.

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 characteristic can then be expanded to a family of trails 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).

Outline. In Sect. 2, we give a description of the tweakable block cipher MANTIS and highlight some properties relevant for our attack. In Sect. 3, we introduce a family of differential trails and a corresponding initial structure of messages for MANTIS_5 that lead to a good filter after 9 rounds. Finally, in Sect. 4, we use this initial structure and filter to mount a key recovery attack on MANTIS_5.

2 Description of MANTIS

2.1 The Tweakable Block Cipher

MANTIS is a tweakable block cipher recently 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 $(64+64)$-bit key $K = k_0 \| k_1$ and 64-bit tweak $T = t_0 \| t_1 \| \cdots \| t_{15}$. All values are mapped to 4×4 states of 4-bit cells, for example,

$$T = \begin{array}{|c|c|c|c|} \hline t_0 & t_1 & t_2 & t_3 \\ \hline t_4 & t_5 & t_6 & t_7 \\ \hline t_8 & t_9 & t_{10} & t_{11} \\ \hline t_{12} & t_{13} & t_{14} & t_{15} \\ \hline \end{array}.$$

The cipher's structure is similar to PRINCE, with r forward rounds $\mathcal{R}_i$ and r backward rounds $\mathcal{R}_i^{-1}$, 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 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 Fig. 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 sequence of transformations

$$\mathcal{R}_i = \mathsf{MixColumns} \circ \mathsf{PermuteCells} \circ \mathsf{AddRoundTweakey}_i \circ \mathsf{AddConstant}_i \circ \mathsf{SubCells},$$

or for the inverse rounds

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

as illustrated in Fig. 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].

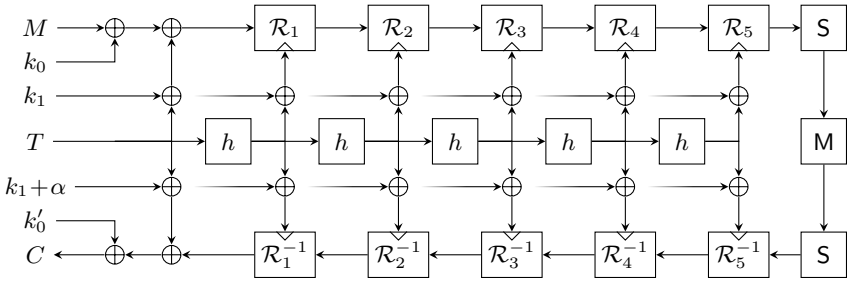


Fig. 1: PRINCE-like structure of MANTIS$_r$, illustrated for MANTIS$_5$.

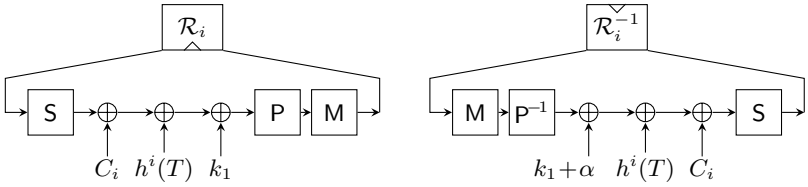


Fig. 2: The MANTIS round functions $\mathcal{R}_i$ and $\mathcal{R}_i^{-1}$.

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

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

AddRoundTweakey$_i$ (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:

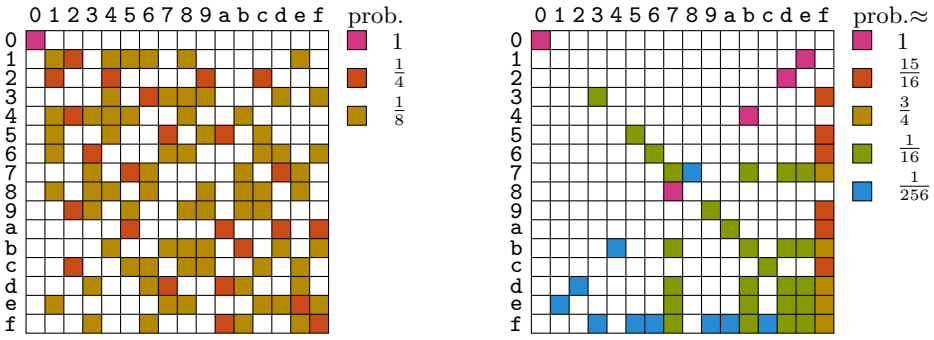
0	1	2	3		6	5	14	15
4	5	6	7	h	0	1	2	3
8	9	10	11	→	7	12	13	4
12	13	14	15		8	9	10	11

PermuteCells (P). The cells of the state are permuted as follows:

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

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 Fig. 3b:

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



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

Fig. 3: Differential properties of the MANTIS round operations.

3 Differential Trail

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 Trails

Our attack is based on a truncated differential trail for the related-tweak setting that meets this lower bound of 34 active S-boxes. The trail is illustrated in Fig. 4. Instead of considering only a single fixed input difference and characteristic, we cluster several related trails following the same truncated trail, thus obtaining a much better probability.

An optimal trail. To analyze the probability, we first construct a trail that matches the claimed optimal differential probability of $2^{-34 \cdot 2} = 2^{-68}$. Consider the differential distribution table of SubCells, given in Fig. 3a. Observe that SubCells is an involution, so the table is symmetric. There is one input/output difference, a, such that all transitions from or to difference 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 a.

Since all non-trivial MixColumns transitions of the trail in Fig. 4 match its branch number, setting all active cells to a results in a valid differential characteristic with the claimed optimal probability of $2^{-34 \cdot 2} = 2^{-68}$.

Clustering trails. We will now relax some of these constraints, and also consider trails with cell differences other than a in selected sections of the trail. Interesting candidates are all differences that can be mapped from and to a by SubCells, i.e., the differences 5, a, d, and f.

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 $\{5, \mathtt{a}, \mathtt{d}, \mathtt{f}\}$ for the two S-boxes, marked 5 in Fig. 4. Then, the trail will follow the same truncated differential, with the same probability of $2^{-4 \cdot 2}$ to transition to the all-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-a trail.

A similar observation applies for the two S-boxes S_3 and S_{12} of Round 2, marked 4 in Fig. 4. 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 a, but each to both a and 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 6 in Fig. 4. 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

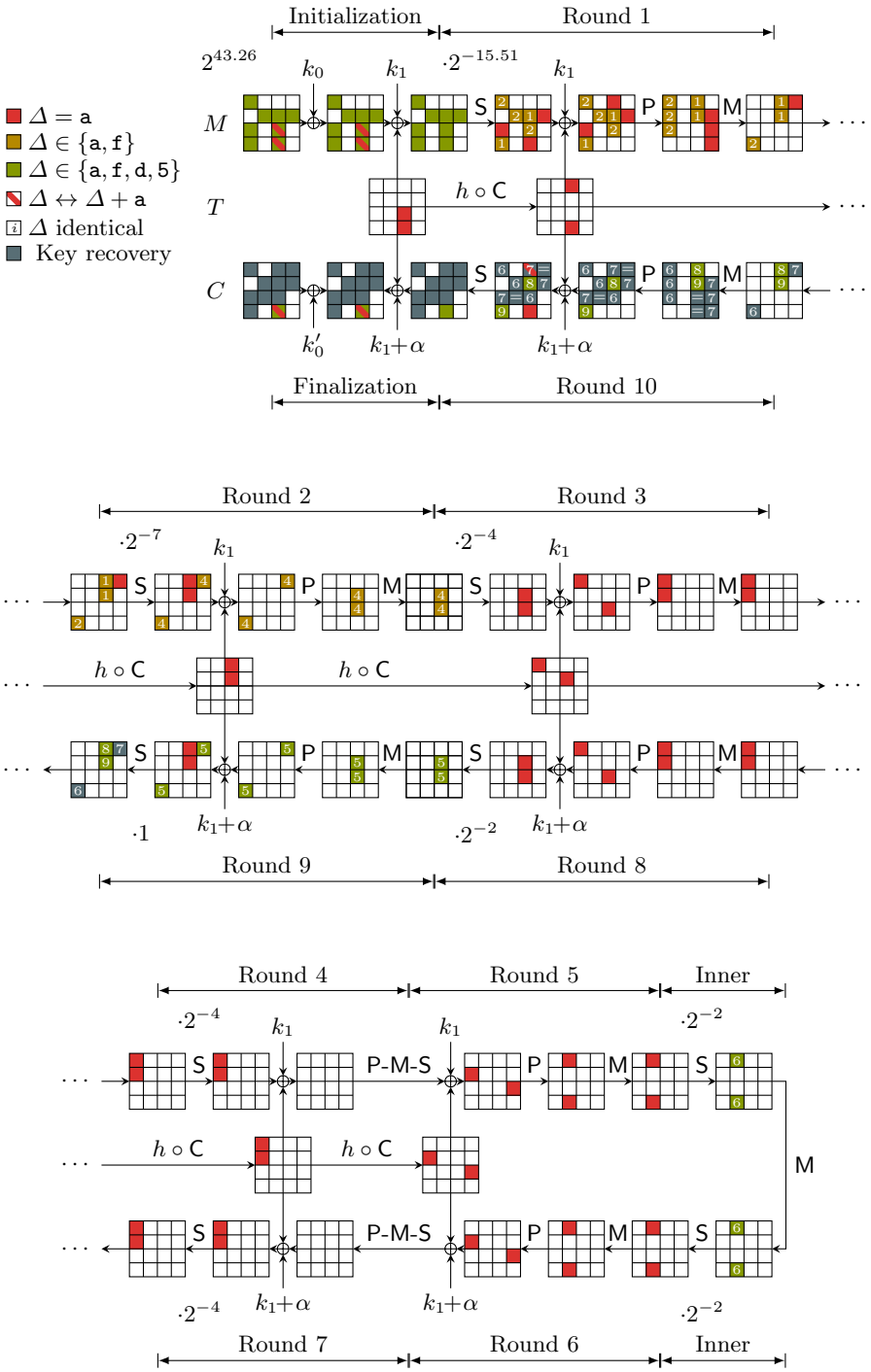


Fig. 4: Trail for MANTIS_5.

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 (1, 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 Fig. 4, we need to compensate the AddRoundTweakey 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

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

Consequently, the overall probability of the trail up to Round 9 (or more precisely, up to AddRoundTweakey 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 trail up to Round 9, the output of the AddRoundTweakey 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 $\mathtt{a}$ (2-bit filter).
- Cells S_0, S_5, S_{10} (marked 6) will have the same difference (8-bit filter), as will cells S_2, S_7, S_8 (marked 7) after compensating for the tweak difference (8-bit filter).
- Cells S_6 and S_{12} (marked 8, 9) 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 trail 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 trail'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 $\mathtt{a}$ in two cells, as specified by the trail in Fig. 4. 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.

(a) Differences $\{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$ (■). (b) Differences $\{\mathtt{0}, \mathtt{5}, \mathtt{7}, \mathtt{f}\}$ (■).

Fig. 5: Initial structure with $8 \cdot 4$ pairs from $2 \cdot 8$ queries per cell.

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 Fig. 5. Here, we exploited that $\mathtt{a} + \mathtt{5} = \mathtt{f}$, where all these three values are suitable for our trail. 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

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

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 trail family and initial structure from Sect. 3 to recover the two 64-bit secret keys k_0 and k_1.

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 trail given in Fig. 4. 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 trail 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 from the initial structure to a set of $2^{41-22} = 2^{19}$ pairs. This set of 2^{19} pairs is still expected to contain $2^{0.49} > 1$ valid pairs that follow the trail of Fig. 4.

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}:

- **1st column:** 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} (6) leads to an equal difference after an S-box application, which happens with a probability of 2^{-8} per ciphertext pair and key guess.
- **2nd column:** 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}.
- **3rd column:** For this column, cells S_2 (8) and S_6 (9) 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} (8, 9) 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.

– **4th column:** For this column, the same reasoning as for the 1st column applies, now for ciphertext and key cells S_3, S_7, S_8 (7) after compensating for the last tweak addition. Again, the probability is 2^{-8}.

If we now decrypt one ciphertext pair backwards for one round 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. If we repeat this for all 2^{19} pairs, where we expect that one of them follows the trail in Fig. 4, we expect at most $2^{14} \cdot 2^{19} = 2^{33}$ candidates for the right subkey which lead to at least one valid ciphertext pair. Hence, we effectively reduce our keyspace by 2^{-11}. So, repeating the attack a total of 4 times with fresh initial structures is sufficient to recover the correct 44 bits of $k'_0 + k_1$ with a data complexity of $4 \cdot 2^{26} = 2^{28}$ chosen plaintexts.

Note that it is not actually necessary to guess all 44 bit of the subkey at once per ciphertext pair. Instead, we can split up the key guesses column-wise in one 16-bit, two 12-bit, and one 4-bit guess, and combine this information to the 2^{14} key candidates we get for each pair.

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 so that only the valid plaintext pairs following the trail in Fig. 4 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 trail. As shown in Fig. 4, 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.

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 bits and identify the right key using trial encryptions.

4.5 Attack Complexities

Plaintext Pair Generation and Filtering. To generate the $4 \cdot 2^{41} = 2^{43}$ plaintext pairs, we need to query $4 \cdot 2^{26} = 2^{28}$ chosen plaintext with chosen tweaks, well below the complexity limit of 2^{30} for MANTIS_5. Pre-filtering costs 2^{43} state xor operations, and will reduce the relevant pairs to 2^{21} for the remaining attack.

Recovery of 44-bit $k'_0 + k_1$. To get the possible key candidates per ciphertext pair, we need $2 \cdot (2^{16} \cdot 4 + 2 \cdot 2^{12} \cdot 3 + 2^4) = 2^{19.13}$ 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 total, we get four lists, each containing 2^{33} key candidates, which dominates our memory requirements. We need to find matches between the four lists, which adds a computational complexity of roughly 2^{33} operations, depending on the implementation.

Recovery of 32-bit $k_0 + k_1$. Here, we make a 32-bit key guess for 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.

Recovery of k_0 and k_1. Finally, we have to recover the remaining 52 bits of key information to recover the original 64-bit keys k_0 and k_1. To do this, we have to make 2^{52} trial encryptions, which dominates our attack complexity.

Summarizing, we recover the full key for MANTIS_5 with a computational complexity of 2^{52} encryptions, memory requirements of 2^{33} MANTIS states, and a data complexity of 2^{28} chosen plaintexts with chosen tweaks. This violates the security claims for MANTIS_5.

The computational complexity can be further reduced by using a slightly different initial structure with an active cell S_2 in the message, or by guessing additional cells of the round key k_1 in Round 1 to add more linear equations.

Acknowledgements. We thank the MANTIS designers for verifying 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)