

Clustering Related-Tweak Characteristics: Application to MANTIS-6

Maria Eichlseder and Daniel Kales

Graz University of Technology, Austria

maria.eichlseder@iaik.tugraz.at

daniel.kales@student.tugraz.at

Abstract. The TWEAKEY/STK construction is an increasingly popular approach for designing tweakable block ciphers that notably uses a linear tweak schedule. Several recent attacks have analyzed the implications of this approach for differential cryptanalysis and other attacks that can take advantage of related tweakeys. We generalize the clustering approach of a recent differential attack on the tweakable block cipher MANTIS₅ and describe a tool for efficiently finding and evaluating such clusters. More specifically, we consider the set of all differential characteristics compatible with a given truncated characteristic, tweak difference, and optional constraints for the differential. We refer to this set as a semi-truncated characteristic and estimate its probability by analyzing the distribution of compatible differences at each step. We apply this approach to find a semi-truncated differential characteristic for MANTIS₆ with probability about $2^{-67.73}$ and derive a key-recovery attack with a complexity of about $2^{53.94}$ chosen-plaintext queries and computations. The data-time product is $2^{107.88} \ll 2^{126}$.

Keywords: (Truncated) Differential Cryptanalysis · TWEAKEY · MANTIS

1 Introduction

Tweakable block ciphers generalize the concept of block ciphers by adding an additional public input, the tweak. This tweak plays a role similar to the nonces or initialization values of higher-level modes of operation, and provides additional variation of the instances of the cipher family. The concept was formally introduced by Liskov et al. [LRW02], who defined it as a family \tilde{E} of permutations $\tilde{E} : \{0, 1\}^k \times \{0, 1\}^t \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. \tilde{E} maps a k -bit key K , t -bit tweak T and n -bit plaintext M to an n -bit ciphertext C , such that $\tilde{E}(K, T, \cdot)$ is a permutation. The recent popularity of tweakable block ciphers, for instance in the CAESAR competition, shows that tweakable block ciphers may be more naturally suited as building blocks for higher-level modes of operation than block ciphers. A particularly relevant application area for tweakable block ciphers is memory and disk encryption, where the address of each data item defines the tweak. However, generic constructions to turn block ciphers $E(K, M)$ into secure tweakable block ciphers $\tilde{E}(K, T, M)$ are often not well-suited for such applications, since they incur a significant latency overhead compared to a plain block cipher call.

Compared to generic constructions that use some block cipher as a black box, dedicated constructions try to provide more efficient designs with full security by integrating the tweak in the core primitive design. With the TWEAKEY framework, Jean et al. [JNP14] propose to treat the tweak in almost the same way as the key in a key-alternating construction. This approach, and in particular the special case STK with its linear tweak schedule, has been adopted in several CAESAR candidates (Deoxys, Joltik, KIASU), as well as standalone tweakable block cipher designs like SKINNY and MANTIS [BJK⁺16] or QARMA [Ava17].

Regarding the cryptanalytic implications of this approach, one central aspect is the possibility of related-tweak attacks. The tweak is usually assumed to be under the attacker’s control, although in practice, the definition of the mode of operation that uses the cipher may impose some constraints. In particular, this means that the attacker can introduce differences via the key schedule, similar to related-key attacks on classical block ciphers. This increases the number of rounds necessary for security against differential cryptanalysis, as well as certain other attacks [DEM16], such as integral distinguishers or Meet-in-the-Middle attacks. For designers, this means that they must analyze bounds for the differential probability in the related-key model. Standard search approaches for finding or lower-bounding the best characteristics, such as mixed-integer linear programming (MILP), satisfiability (SAT) or constraint programming (CP) solvers, can usually be adapted to the related-tweak case.

The output of such a search is either an optimal differential characteristic or, more often, a truncated differential characteristic with the minimum number of active S-boxes, referred to as “minimal characteristic” in the following. For standard strongly aligned block ciphers in the fixed-key model, the bounds derived from such a minimal characteristic are usually both reasonably tight and reasonably reliable to estimate the security margin. However, several recent papers have discussed issues which indicate that the bounds obtained from minimal characteristics of STK-based tweakable block ciphers can be less useful. The main reason for this is the deterministic behaviour of the linear tweak schedule with respect to the input tweak difference. Cid et al. [CHP⁺17] showed that if this is not considered in the search, the resulting minimal characteristics are often invalid, and that tighter bounds can be obtained by adapting the search model accordingly. Dobraunig et al. [DEKM16], on the other hand, take advantage of the predictable tweeky schedule to cluster several differential characteristics with nearly optimal probability for an attack on MANTIS₅.

Our contributions

We generalize the clustering approach from the attack on MANTIS₅ [DEKM16] and describe a tool for efficiently finding and evaluating such clusters. Whereas the cluster for MANTIS₅ was found by hand and was simple enough for its probability to be evaluated on a round-by-round basis, we argue that such probability estimates are not accurate in general. Instead of starting with a differential characteristic and trying to find similar characteristics that can be clustered, we start from a truncated differential characteristic (plus, optionally, a fixed, compatible differential) and consider all compatible differential characteristics for a fixed tweak difference.

To represent the resulting family of individual characteristics in a compact way, we describe the set of permissible differences for each intermediate state on a cell-by-cell basis. We refer to the resulting structured cluster of characteristics as a semi-truncated characteristic. We then need to efficiently estimate the probability of the semi-truncated characteristic without enumerating all individual characteristics, somewhat similar to the probability of a truncated characteristic. Due to the influence of the tweak difference, we need to estimate the expected distribution of differences within the specified set at each step and analyze the resulting transition probabilities of each operation.

In summary, we combine advantages of classical and truncated characteristics: On the one hand, by clustering many characteristics, we improve the overall probability and generate pairs more efficiently compared to the single best differential characteristic. On the other hand, a straightforward truncated approach cannot take advantage of the high-probability transitions in the S-box, incurs significant costs from the linear constraints imposed by the tweak schedule, and does not provide a fixed output difference that can be used, for instance, for boomerang attacks. We discuss how such semi-truncated characteristics can be applied to obtain efficient key-recovery attacks, and analyze the complexity and possible tradeoffs for this approach.

We apply this approach to find a semi-truncated differential characteristic for MANTIS_6 with probability about $2^{-67.73}$ and derive a key-recovery attack with a complexity of about $2^{53.94}$ chosen-plaintext queries and computations. The data-time product of $2^{107.88}$ is below the designers' bound of 2^{126} claimed for MANTIS_5 and MANTIS_7 (with additional data limits for MANTIS_5). The designers' bound for the probability of the best characteristic is $\leq 2^{-88}$. Note that MANTIS_6 has a block size of 64 bits, so the probability of our semi-truncated characteristic is worse than the generic probability of any fixed differential, and much worse than the generic probability of $2^{-33.58}$ of its semi-truncated differential.

Outline

In Section 2, we provide a brief description of the TWEAKEY construction and the tweakable block cipher MANTIS, as well as some of its cryptographic properties. In Section 3, we introduce our approach for finding semi-truncated characteristics, estimating their probability, and deriving key-recovery attacks. In Section 4, we apply the approach to find a semi-truncated characteristic for MANTIS_6 and develop a corresponding key-recovery attack.

2 Background on MANTIS

2.1 The Tweakable Block Cipher MANTIS

MANTIS is a tweakable block cipher published at CRYPTO 2016 by Beierle et al. [BJK⁺16]. The designers propose several variants MANTIS_r that differ only in the number of rounds. All variants operate on a 64-bit message block $M = M_0 \| M_1 \| \dots \| M_{15}$ and work with a 64-bit tweak $T = T_0 \| T_1 \| \dots \| T_{15}$ 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 , where S_0, \dots, S_3 is the first row, etc.

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 $S \circ M \circ S$. 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 $k'_0 = (k_0 \ggg 1) + (k_0 \ggg 63)$ 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 Figure 1a.

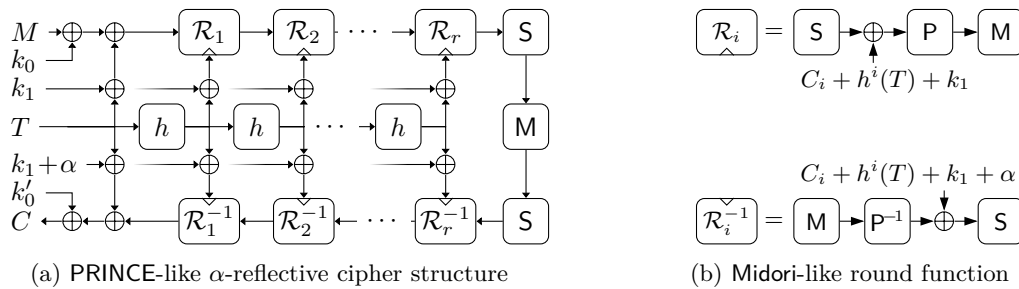
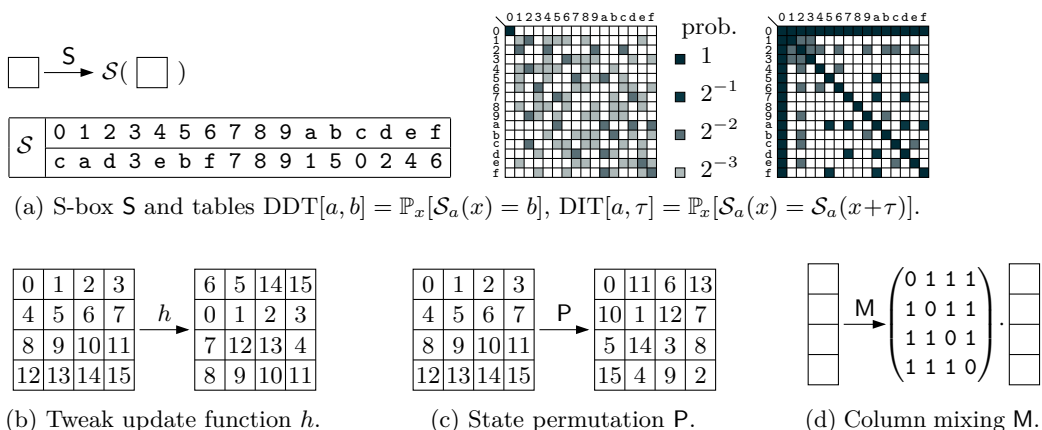


Figure 1: Design of the tweakable block cipher MANTIS_r .

The round function \mathcal{R}_i is very closely related to that of Midori [BBI⁺15]. It updates the 4×4 state of 4-bit cells by means of the sequences of transformations \mathcal{R}_i and \mathcal{R}_i^{-1} , as illustrated in Figure 1b. Its S-box layer (SubCells) and linear layer (PermuteCells, MixColumns) are directly inherited from Midori [BBI⁺15]. In the following, we briefly describe the individual operations. For a more detailed description of the MANTIS family, we refer to the design paper [BJK⁺16].

Figure 2: Nonlinear and linear transformations of the MANTIS round function \mathcal{R}_i .

- **SubCells (S)** applies the involutive 4-bit S-box \mathcal{S} given in Figure 2a to each state cell. For our attack, we are primarily interested in the differential behaviour of \mathcal{S} . The differential distribution table (DDT) in Figure 2a shows that \mathcal{S} has 24 differential transitions with a probability of 2^{-2} ; for two of the input differences, 2 and 4, each of the four possible output differences is observed with probability 2^{-2} . This is due to the algebraic properties of \mathcal{S} : only 12 of the 15 component functions have algebraic degree 3. In addition to the first derivative $\mathcal{S}_a(x) := \mathcal{S}(x) + \mathcal{S}(x+a)$ of the S-box, as tabulated in the DDT, we will also refer to some properties of the second derivative $\mathcal{S}_{a,\tau}(x) := \mathcal{S}_a(x) + \mathcal{S}_a(x+\tau)$, in particular the case $\mathcal{S}_{a,\tau} = 0$ as tabulated in the differential invariance table (DIT) in Figure 2a.
- **AddTweakey_i (A)** and **AddConstant_i (C)** add 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 $h^i(T)$ to the state. The tweakey update function h simply permutes the order of cells as specified in Figure 2b.
- **PermuteCells (P)** permutes the state cells as specified in Figure 2c.
- **MixColumns (M)** multiplies columns with involutive near-MDS matrix M in Figure 2d.

2.2 Previous Cryptanalysis Results

Security claims for MANTIS are given with respect to the data-time product limit of $D \cdot T < 2^{126}$ due to generic attacks on its FX construction [KR96], similar to the claims for PRINCE [BCG⁺12]. With a MILP model of the cipher’s differential behaviour in a related-tweak model, the designers are able to prove lower bounds of at least 34, 44, 50 active S-boxes with MDP 2^{-2} for 5, 6, 7 rounds (corresponding to 12, 14, 16 S-box layers) [BJK⁺16]. Explicit security claims are given for MANTIS₅ for an attacker constrained to $D \leq 2^{30}$ chosen plaintexts or $D \leq 2^{40}$ known plaintexts, and for MANTIS₇ without further constraints besides the data-time product $D \cdot T < 2^{126}$.

Dobraunig et al. [DEKM16] refuted the claim for MANTIS₅ with a differential attack using 2^{30} chosen plaintexts and a practical runtime of about 1 hour, or about 2^{38} cipher calls. This attack exploits the minimalistic security margin and a strong clustering effect of differential characteristics. Dobraunig et al. start from a truncated differential characteristic and show how to find a consistent optimal differential characteristic. Finally, they collect many more closely related optimal and near-optimal characteristics to obtain a cluster with higher probability, estimated as $2^{-40.51}$, and with multiple starting differences, thus reducing the data complexity with suitable initial structures.

3 Semi-Truncated Families of Differential Characteristics

In this section, we consider families of differential characteristics for tweakable block ciphers designed according to the TWEAKEY/STK approach. Considering several characteristics instead of a single one offers two primary advantages: First, by allowing several different input differences, candidate plaintext pairs can often be generated more efficiently. This is particularly useful when the data complexity is a limiting factor for the attack complexity, such as in FX designs. Second, by allowing several differences in the middle of the characteristic, the overall probability that a pair follows any of those characteristics is increased.

The classical approach to take advantage of both effects is to consider truncated differential characteristics [Knu94]. All published STK designs are strongly aligned, AES-like ciphers, so we focus on such designs. When applied to AES-like designs, the probability of (cell-wise) truncated differential characteristics is usually evaluated by considering the approximate probability of all MixColumns transitions, which depends primarily on the number of inactive output cells. For related-tweak truncated characteristics, cancellations of tweak differences need to be taken into account in a similar manner. A more fine-grained approach is to consider several individual differential characteristics, as in multiple differential cryptanalysis [BG11], and base the analysis on the knowledge of each individual characteristic’s probability.

When we try to apply the classical estimates for the probability of (aligned) truncated characteristics in the context of tweakable block ciphers with a linear tweak schedule, we notice that the estimates become very unreliable. Consider a related-tweak scenario. Once the input tweak difference for a pair is fixed, the deterministic differential tweak schedule imposes many constraints on the differences of the intermediate values in all rounds. Often, these constraints will be contradictory; in some of the remaining cases, the probability that the pair follows the truncated characteristic will be much higher than estimated. In their analysis of Deoxys and Joltik, Cid et al. [CHP⁺17] observed that indeed many truncated characteristics turn out to be impossible for all input tweak differences, and proposed additional criteria to identify and eliminate such cases. Since truncated characteristics are also used as an intermediate step to find or derive bounds on standard differential characteristics, eliminating such impossible truncated characteristics is important for faster search results and tighter bounds.

What remains unclear is if and how truncation can be used to improve actual differential attacks on tweakable designs, and how the probability of such constrained truncated characteristics should be estimated. In the attack on MANTIS₅, Dobraunig et al. [DEKM16] cluster several individual differential characteristics that follow the same truncated characteristic for a fixed tweak difference. They start from one solution and manually add other, similar characteristics, which deviate in a few S-box transitions. To describe the resulting family of characteristics, they specify a small set of possible differences in each cell of the characteristic. The probability is estimated on a round-by-round basis by considering the columnwise transition probability for MixColumns \circ SubCells, summing over all permitted output differences, and averaging over all input differences. A practical verification confirmed that the estimates are reasonably accurate.

However, when we try to extend the MANTIS₅ approach [DEKM16] to more rounds of MANTIS or to other STK designs, we face several issues. First, finding clusters manually is tedious and error-prone work, in particular if many truncated starting points are contradictory. Second, the cell-wise probability estimate only works well for MANTIS₅ due to the special structure of the characteristic, where almost all S-box transitions have either only one input difference or only one output difference with equiprobable transitions for all input differences. Third, the “optimality” of the cluster (with respect to the truncated starting point) is unclear, it might be possible to add further characteristics. In this section, we address these issues with a more generally applicable, partially automated approach.

3.1 Finding Semi-Truncated Characteristics

As a first step, we want to characterize the set of all compatible differential characteristics subject to several constraints configured by the cryptanalyst:

- **Truncated constraints:** A truncated characteristic serves as a starting point. The resulting set of characteristics will be compatible with this truncated characteristic. We chose several candidates with a close-to-minimal number of active S-boxes obtained from a MILP (or SAT) solver, i.e., optimized for differential probability. For target designs with stronger S-boxes and lower diffusion, it may be more efficient to start from truncated characteristics optimized for truncated probability instead.
- **Fixed tweak difference:** We consider only characteristics with one fixed tweak difference, for several reasons. The value of the tweak difference can completely change both the structure of the set of compatible characteristics and its probability, so for an attack, it usually only makes sense to consider the “best” difference. For many attacks, like boomerangs [CHP⁺17], only a fixed tweak difference is useful. If the number of active tweak cells in the truncated characteristics is low, or if many combinations can be excluded due to linear constraints, all possible tweak differences can be evaluated based on the quality of the resulting semi-truncated characteristic.
- **Input/output constraints:** If the attack setup requires, the differential can be additionally constrained. Adding constraints to a finished characteristic may also be useful for tradeoffs between different attack phases, as discussed in the next sections.

We want to describe a superset of the set of compatible characteristics where we specify for each state cell a list of possible differences. In absence of a tweak difference, this superset would usually be a standard truncated characteristic, where each cell permits either only the zero difference or any difference (except maybe for the last round); but with the constraints of a linear tweak schedule and a fixed tweak difference, this superset is significantly reduced, and we refer to it as a semi-truncated characteristic.

We initialize the semi-truncated characteristic based on the initial constraints and propagate these constraints across the steps of the round function. Consider the state geometry and operations of MANTIS as an example. For each intermediate state $S = (S_0, \dots, S_{15})$, let $\chi = (\chi_0, \dots, \chi_{15})$ with $\chi_i \subseteq \mathcal{X} = \{0, \dots, \mathbf{f}\}$ denote the set of differences specified by the semi-truncated characteristic. We now consider an operation $f \in \{S, A, P, M\}$ in some round of the cipher. Let S be the input state and $S^f = f(S)$ the output state of this operation. We iteratively update the sets with propagated information for each operation:

SubCells (S): The relation between consistent input and output differences is defined by the DDT of the S-box. We define the according set transition function $\sigma : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{X}}$:

$$\sigma(X) := \{y \in \mathcal{X} \mid \exists x \in X : \text{DDT}(x, y) > 0\}.$$

Since the MANTIS S-box is involutive, we can eliminate unreachable differences by

$$\begin{aligned} \chi_i^S &\leftarrow \chi_i^S \cap \sigma(\chi_i), & i = 0, \dots, 15, \\ \chi_i &\leftarrow \chi_i \cap \sigma(\chi_i^S), & i = 0, \dots, 15. \end{aligned}$$

PermuteCells (P): The output set must equal the permuted input set, so we update

$$\begin{aligned} \chi_i^P &\leftarrow \chi_i^P \cap [P(\chi)]_i, & i = 0, \dots, 15, \\ \chi_i &\leftarrow \chi_i \cap [P^{-1}(\chi^P)]_i, & i = 0, \dots, 15. \end{aligned}$$

AddTweakey (A): If τ denotes the tweak difference and $S \oplus \tau_i := \{s \oplus \tau_i \mid s \in S\}$, update

$$\begin{aligned} \chi_i^A &\leftarrow \chi_i^A \cap (\chi_i \oplus \tau_i), & i = 0, \dots, 15, \\ \chi_i &\leftarrow \chi_i \cap (\chi_i^A \oplus \tau_i), & i = 0, \dots, 15. \end{aligned}$$

MixColumns (M): The possible transitions need to be analyzed column by column. Consider the column $I = (I_0, I_1, I_2, I_3) \in \mathcal{I} = \{(0, 4, 8, 12), \dots, (3, 7, 11, 15)\}$ with its associated characteristic $\chi_I = \chi_{I_0} \times \dots \times \chi_{I_3}$. Using the linearity of M, we update

$$\begin{aligned}\chi_{I_j}^M &\leftarrow \chi_{I_j}^M \cap \{[M \cdot \delta]_j \mid \delta \in \chi_I, M \cdot \delta \in \chi_I^M\}, & I \in \mathcal{I}, j = 0, \dots, 3, \\ \chi_{I_j} &\leftarrow \chi_{I_j} \cap \{[M \cdot \delta]_j \mid \delta \in \chi_I^M, M \cdot \delta \in \chi_I\}, & I \in \mathcal{I}, j = 0, \dots, 3.\end{aligned}$$

For the specific case of transitions that match the branch number bound for the MANTIS matrix, these update conditions can be simplified: All valid transitions must have the same difference in all 4 active input/output cells, so all active sets must be equal. Let χ_I^* denote the 4 active cells of $\chi_I \parallel \chi_I^M$, then we only update

$$[\chi_I^*]_0, [\chi_I^*]_1, [\chi_I^*]_2, [\chi_I^*]_3 \leftarrow [\chi_I^*]_0 \cap [\chi_I^*]_1 \cap [\chi_I^*]_2 \cap [\chi_I^*]_3, \quad I \in \mathcal{I}.$$

Each update of a state potentially impacts the two neighbouring states, so we iterate the updates until the semi-truncated characteristic converges to a fixed-point where none of the update steps causes any more changes. The resulting reduced semi-truncated characteristic still describes a superset of all consistent differential characteristics that follow the initial constraints. We refer to all characteristics in this superset as “compatible”, and use “consistent” or “possible” to mean characteristics with non-zero probability. Next, we want to estimate the probability of such a reduced semi-truncated characteristic, and see how further constraints impact the resulting attacks.

3.2 Probability of Semi-Truncated Characteristics

The probability of a semi-truncated characteristic is defined as the sum of probabilities of all compatible differential characteristics for a fixed input difference, averaged over all compatible input differences. Similarly, the probability of a semi-truncated differential is defined as the probability that any compatible output difference is observed for a fixed input difference, averaged over all compatible input differences. As usual, we will assume that the probability of an individual differential characteristic (for the fixed target key) can be estimated based on the average probability (across all long-keys), which is in turn computed by multiplying the differential probabilities of each round for a Markov cipher.

A straightforward approach for estimating the probability of a semi-truncated characteristic is to apply the definition to each round operation, and multiply all the obtained round probabilities. This approach was applied in the attack on MANTIS₅ [DEKM16] (except for the first round). The relevant round operations for evaluating the semi-truncated probability are SubCells (as for individual characteristics) and MixColumns (as for truncated characteristics); the other operations are trivial if the semi-truncated characteristic is reduced. In this straightforward computation we however made two Markovian assumptions:

- (a) **Uniformity of values:** For each individual characteristic, we make the usual Markov assumption that the input values to SubCells are uniformly distributed; and
- (b) **Uniformity of differences:** By using the definition of the probability of a semi-truncated differential and averaging over all input differences, we make a similar uniformity and independence assumption regarding the distribution of the differences in each round among the compatible characteristics.

In the case of MANTIS, the first assumption seems reasonable except in the inner part, which features two successive SubCells layers without a key addition in between. For the specific semi-truncated characteristic used for MANTIS₅, the second assumption is also well-justified in most rounds, for example due to the uniform distribution of the message input or the most frequent transitions with 4 equiprobable differentials. However, in general – and in Round 2 in particular – this assumption does not apply.

To obtain a more accurate estimate in general, it is necessary to consider not only the set of differences at each step, but their expected distribution among all compatible, consistent differential characteristics that contribute to the probability. Consider an intermediate state S with semi-truncated characteristic χ . The difference in this state for a random compatible plaintext pair is a random variable $\Delta = (\Delta_0, \dots, \Delta_{15})$. We write $\Delta \in \chi$ for the event $\Delta_i \in \chi_i$ for all i , and $\overline{\Delta} \in \overline{\chi}$ to state that all intermediate differences in the steps up to and including S follow the semi-truncated characteristic for a particular input pair. We are interested in the distribution of Δ in case $\overline{\Delta} \in \overline{\chi}$, and specifically, in the cell-wise conditional distribution defined by the probability mass function φ_i :

$$\varphi_i : \quad \mathcal{X} \rightarrow [0, 1], \quad \delta \mapsto \mathbb{P}[\Delta_i = \delta \mid \overline{\Delta} \in \overline{\chi}].$$

Now consider an operation $f \in \{\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{M}\}$ that is applied to the input state S to produce the output state $S^f := f(S)$. We want to derive the conditional distribution φ_i^f of Δ^f and estimate the probability \overline{p}^f of the semi-truncated characteristic up to this state:

$$p^f = \mathbb{P}[\overline{\Delta}^f \in \overline{\chi}^f \mid \overline{\Delta} \in \overline{\chi}], \quad \overline{p}^f = \mathbb{P}[\overline{\Delta}^f \in \overline{\chi}^f] = p^f \cdot \mathbb{P}[\overline{\Delta} \in \overline{\chi}].$$

As an intermediate step, we consider the distribution of Δ^f without the constraints χ^f , i.e., $\tilde{\varphi}_i^f$ under the condition $\overline{\Delta} \in \overline{\chi}$ instead of φ_i^f under $\overline{\Delta}^f \in \overline{\chi}^f$ (so $\varphi_i^f(\delta) = 0$ for $\delta \notin \chi_i^f$):

$$\tilde{\varphi}_i^f : \quad \mathcal{X} \rightarrow [0, 1], \quad \delta \mapsto \mathbb{P}[\Delta_i^f = \delta \mid \overline{\Delta} \in \overline{\chi}].$$

For **AddTweakey** and **PermuteCells**, we trivially get $p^f = 1$, and $\tilde{\varphi}_i^f = \varphi_i^f$ is a permuted φ_i .

For **SubCells**, let $\mathbb{P}[\alpha \xrightarrow{\mathcal{S}} \delta]$ denote the differential probability of (α, δ) obtained from the DDT of S-box \mathcal{S} . Furthermore, let $\mathbb{1}_{\chi_i}$ denote the indicator function of χ_i : If $\delta \in \chi_i$ then $\mathbb{1}_{\chi_i}(\delta) = 1$, else $\mathbb{1}_{\chi_i}(\delta) = 0$. If we assume that the distributions φ_i are independent, then

$$\begin{aligned} \tilde{p}_i^{\mathcal{S}}(\delta) &= \sum_{\alpha \in \chi_i} \varphi_i(\alpha) \cdot \mathbb{P}[\alpha \xrightarrow{\mathcal{S}} \delta], & p_i^{\mathcal{S}} &= \sum_{\delta \in \chi_i^{\mathcal{S}}} \tilde{\varphi}_i^{\mathcal{S}}(\delta), \\ \varphi_i^{\mathcal{S}}(\delta) &= \mathbb{1}_{\chi_i^{\mathcal{S}}}(\delta) \cdot \frac{\tilde{\varphi}_i^{\mathcal{S}}(\delta)}{p_i^{\mathcal{S}}}, & p^{\mathcal{S}} &= \prod_i p_i^{\mathcal{S}}. \end{aligned}$$

For **MixColumns**, the distribution needs to be evaluated column by column for each $I \in \mathcal{I}$. Then, assuming the input distributions φ_i are independent, we get the following distribution $\varphi_I^{\mathbf{M}}$ of column differences $\Delta_I = (\Delta_{I_0}, \dots, \Delta_{I_3})$ and (dependent) cell distributions $\varphi_{I_j}^{\mathbf{M}}$:

$$\begin{aligned} \tilde{\varphi}_I^{\mathbf{M}}(\delta_I) &= \varphi_I(\mathbf{M} \cdot \delta_I) = \prod_j \varphi_{I_j}([\mathbf{M} \cdot \delta_I]_j), & p_I^{\mathbf{M}} &= \sum_{\delta_I \in \chi_I^{\mathbf{M}}} \tilde{\varphi}_I^{\mathbf{M}}(\delta_I), \\ \varphi_I^{\mathbf{M}}(\delta_I) &= \mathbb{1}_{\chi_I^{\mathbf{M}}}(\delta_I) \cdot \frac{\tilde{\varphi}_I^{\mathbf{M}}(\delta_I)}{p_I^{\mathbf{M}}} \Rightarrow \varphi_{I_j}^{\mathbf{M}}(\delta) = \sum_{[\delta_I]_j = \delta} \varphi_I^{\mathbf{M}}(\delta_I), & p^{\mathbf{M}} &= \prod_I p_I^{\mathbf{M}}. \end{aligned}$$

For the special case of meeting the branch number bound with $a \in \{1, 2, 3\}$ active input cells and $4 - a$ active output cells, all active cells share the same set χ_* . Then, all active output cells will also share an identical (dependent) distribution $\varphi_*^{\mathbf{M}}$. For example, in the simplest case that the input cells are also identically (independently) distributed by φ_* :

$$p_I^{\mathbf{M}} = \sum_{\delta \in \chi_*} [\varphi_*(\delta)]^a, \quad \varphi_*^{\mathbf{M}}(\delta) = \mathbb{1}_{\chi_*}(\delta) \cdot \frac{[\varphi_*(\delta)]^a}{p_I^{\mathbf{M}}}.$$

Clearly, the independence assumptions required at each step will usually not be satisfied. On the other hand, maintaining a full-state distribution φ for each state is not practicable. As a practical compromise, we consider the dependencies $\varphi_I^{\mathbf{M}}$ introduced by **MixColumns** in the next **SubCells**, but assume that the following **PermuteCells** “clears” the dependencies:

$$\tilde{\varphi}_I^{\mathcal{S}}(\delta_I) = \sum_{\alpha_I \in \chi_I} \varphi_I(\alpha_I) \cdot \prod_j \mathbb{P}[[\alpha_I]_j \xrightarrow{\mathcal{S}} [\delta_I]_j], \quad p_I^{\mathcal{S}} = \sum_{\delta_I \in \chi_I^{\mathcal{S}}} \tilde{\varphi}_I^{\mathcal{S}}(\delta_I).$$

3.3 Exploiting Semi-Truncated Characteristics

Data Collection

Once we have fixed a semi-truncated characteristic and determined an estimate for its probability, we need to consider how to efficiently generate message pairs with a compatible input difference, and how to evaluate the resulting output differences. All these considerations depend on the size of the semi-truncated difference set relative to the total number of possible differences. For this purpose, we identify the semi-truncated difference $\chi = (\chi_0, \dots, \chi_{15})$ with the corresponding expanded set of differences $\chi_0 \times \dots \times \chi_{15} \subseteq \mathcal{X}^{16}$. We then denote the number $|\chi|$ of differences compatible with the semi-truncated difference χ , and their ratio (or filter) $\rho(\chi)$ among all differences, by

$$|\chi| := |\chi_0 \times \dots \times \chi_{15}| = \prod_i |\chi_i| \in [1, |\mathcal{X}|^{16}]$$

$$\rho(\chi) := \frac{|\chi|}{|\mathcal{X}^{16}|} = \prod_i \rho(\chi_i) \in [2^{-16|\mathcal{X}|}, 1].$$

We consider a semi-truncated characteristic with probability p and denote its plaintext-ciphertext differential and tweak difference by (χ^M, χ^C) and χ^T , respectively. Note that the tweak difference is fixed, so $|\chi^T| = |\{\delta^T\}| = 1$, whereas $|\chi^M|, |\chi^C| \geq 1$.

Plaintext pairs can be generated efficiently with initial structures similar to the case of multiple and truncated differentials: We fix a base plaintext M and base tweak T . Then, we query the ciphertexts for the set $\mathcal{M} \times \mathcal{T}$ of plaintext-tweak combinations, where the message set \mathcal{M} and tweak set \mathcal{T} are defined as follows:

$$\mathcal{T} = T \oplus \langle \chi^T \rangle = \{T, T \oplus \delta^T\}, \quad \mathcal{M} = M \oplus \langle \chi^M \rangle,$$

where $\langle S \rangle$ denotes the linear span generated by a set S , i.e., the set of all linear combinations of elements in S . For each queried message in the first half $T \oplus \mathcal{M}$ of this set, there is a corresponding queried message in the second half $(T \oplus \delta^T) \oplus \mathcal{M}$ for any compatible difference $\delta^M \in \chi^M$. Thus, with $2 \cdot |\langle \chi^M \rangle|$ chosen-plaintext queries, we obtained $|\langle \chi^M \rangle| \cdot |\chi^M|$ compatible plaintext pairs. Among this set of compatible pairs, all message differences compatible with χ^M (and each χ_i^M) appear equally often, consistent with the uniform starting distribution we assumed in Subsection 3.2. We can repeat this procedure several more times with different base inputs M and T to generate pairs at a constant rate of $|\chi^M|/2$ pairs per query. This is independent of the structure of the sets χ_i^M and the resulting size of $\langle \chi_i^M \rangle$, except for the obtained granularity of the number of pairs.

If we want to generate enough pairs to expect R valid pairs compatible with the full semi-truncated characteristic, the necessary number of queries N_Q is

$$N_Q = R \cdot \frac{2}{|\chi^M| \cdot p}$$

in case p^{-1} is an integer multiple of $|\langle \chi^M \rangle| \cdot |\chi^M|$, or slightly more otherwise. The resulting $N_P = R/p$ ciphertext pairs can be filtered down to a much smaller number of candidates that still contains about R valid pairs based on the ciphertext difference, which must be in χ^C , resulting in a number of filtered ciphertext pairs N_F of

$$N_F = R \cdot \frac{|\chi^C|}{p}.$$

This filtering can usually be done efficiently without the need to enumerate all R/p ciphertext pairs. For example, we can select the cell positions S_i with the smallest sets χ_i^C , and repeat the following for each base input (T, M) : Store the first half of the ciphertexts

with tweak T in a hash table indexed by the values of the ciphertext cells C_i . Then, for each ciphertext in the second half with tweak $T \oplus \delta^T$, only check the relevant hash table entries according to χ_i^C for matches on the full output difference χ^C . Ideally, if there are sufficiently many cells with $|\chi_i^C| = 1$ (depending on the size $|\chi^M|$), then each filtered ciphertext pair can be identified with minimal amortized cost. In this ideal case, the total complexity is dominated either by the number of queries N_Q or the number of filtered ciphertext pairs N_F , both of which can be significantly smaller than $N_P = R/p$.

Key Recovery

Different approaches to key recovery are possible depending on the properties of the semi-truncated characteristic, such as $|\chi^M|, |\chi^C|, p$, and the cardinalities in the initial and final intermediate rounds. The details also depend heavily on the target cipher, in particular its key schedule. In the remaining paper, we focus on an approach that combines elements of classical 0-round and 1+-round key recovery using standard differential characteristics or differentials. Below, we summarize the basic approach and possible tradeoffs, but refer to [Subsection 4.3](#) for a detailed description of a practical application.

We recover the full key in three phases, where the first phase usually dominates the attack complexity. Note that in this paper, we target a cipher with key size twice as large as the block size, and also essentially more than twice as large as the key size that the attacker can brute-force, so it is not sufficient to just recover a few key bits and brute-force the rest. We assume we have generated a set of N_F filtered ciphertext pairs that contains at least one valid pair compatible with the semi-truncated characteristic, as described above.

In the first phase, we will try to identify this valid pair and recover parts of the initial and final round keys in the process. To this end, we guess parts of the initial and final round key and test for each filtered pair if the resulting intermediate values are compatible with the characteristic. We only keep round key candidates that produce valid intermediate values for at least one characteristic. To estimate how many partial key guesses produce valid intermediate values for a fixed pair, we will assume that the filtered differentials are distributed uniformly among (χ^M, χ^C) . Then, we use the same methods as in [Subsection 3.2](#) for estimating probabilities: for the initial rounds, we reuse the probability estimates for the relevant parts of the characteristics; for the final rounds, we compute estimates in essentially the same way, but based on the inverse round function. This phase reduces the space of key candidates for each cell or column, and can be repeated to (almost) uniquely determine the relevant round key values, as well as identify the valid pair.

In the second phase, we repeat a similar approach to test more conditions of the characteristic and recover more key material. Since we only need to test for one or a few valid pairs instead of all N_F filtered pairs, we can simultaneously guess larger parts of the key and thus cover more initial and final rounds. Finally, in the third phase, we brute-force the remaining key space.

As a tradeoff to balance the complexities arising from N_Q and N_F , we can consider minor adjustments of the semi-truncated characteristic. If N_F dominates the complexity, we can restrict χ^M in order to exclude the lowest-probability characteristics in the set and thus increase p . As an effect, the product $|\chi^M| \cdot p$ will slightly decrease (since we excluded several previously valid pairs), leading to a slight increase in the data complexity N_Q . Another negative effect is that the first rounds of the cipher will provide a slightly less effective filter for key recovery. On the other hand, N_F and the resulting complexity costs for key recovery will be significantly decreased.

4 Application to MANTIS₆

4.1 Finding a Semi-Truncated Characteristic for MANTIS₆

We can now apply the approach to find a semi-truncated characteristic for MANTIS₆. First, we need a truncated characteristic as a starting point. A MILP model similar to the designers' [BJK⁺16] yields characteristics with 44 active S-boxes. However, when evaluating these minimal truncated characteristics, all results show some undesirable properties that negatively influence the final probability, such as MixColumns transitions with branch number > 4 and tweak differences with more than 2 active cells. If we add extra constraints to the MILP model to forbid such properties, the minimum number of active S-boxes grows to 48. Most of the resulting solutions display the same inner structure as the existing 5-round characteristic. We use one of these for the following attack.

To develop the truncated characteristic into a useable semi-truncated characteristic, we need to fix the two active cells of the tweak difference. We can easily enumerate all 225 possible values. The most promising choice is (\mathbf{a}, \mathbf{a}) , the same as for MANTIS₅. Additionally, we can optionally add constraints on the input difference (to optimize the initial structures and average probability) and the output difference (if the intended distinguisher profits from it). For the output, we have no explicit constraints, but we can consider some modifications of the basic truncated trail in the last rounds to improve the attack, as discussed below. For the input, the unconstrained version for the input already provides a good tradeoff between the data complexity and key recovery complexity, so we do not add any constraints for the present attack.

Using the methods introduced in the previous section, the truncated characteristic is developed into the semi-truncated characteristic illustrated in Figure 3. Note that the resulting characteristic has more active S-Boxes in Round 12 than the truncated version. This was done to improve the overall probability by allowing all possible S-Box transitions from Round 11 onward for some cells, resulting in more possible differences in the Round 12. We want to strike a balance between a good probability by allowing more S-Box transitions in the later rounds, a good filtering option by keeping more cells inactive in the ciphertext, and a good key-recovery process by having more active cells in Rounds 11 and 12. We obtained the best results by having exactly half of the cells in the ciphertext active and half inactive.

Figure 3 only indicates the sets χ_i^f and the transition probability estimates p^f at each relevant step f , not the underlying distribution φ_i^f . As an example, consider the SubCells step of round 2. After the preceding MixColumns, cells S_6, S_{10}, S_{14} (■) have the same difference uniformly distributed in $\chi_i = \{\mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{5}\}$. To analyze the transition probability of these three cells, consider each of the four possible differences in turn. Difference \mathbf{a} will be mapped by SubCells to compatible differences in $\chi_{6,10,14}^S$ with probability $p_{6,10,14}^S = 1 \cdot 1 \cdot \frac{1}{4}$, and the differences in S_6^S, S_{10}^S will be uniformly distributed (25% each for $\mathbf{a}, \mathbf{f}, \mathbf{d}, \mathbf{5}$). Difference \mathbf{f} has $p_{6,10,14}^S = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4}$, and a biased output distribution (50% \mathbf{a} , 50% \mathbf{f}). Differences \mathbf{d} and $\mathbf{5}$ each have $p_{6,10,14}^S = \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4}$, and produce 100% \mathbf{a} . On average, the success probability is $\frac{1}{4} \cdot \frac{11}{32} \approx 2^{-3.54}$, and the resulting distribution for $i \in \{6, 10\}$ is $\varphi_i^S(\mathbf{a}) = \frac{4}{11}$, $\varphi_i^S(\mathbf{f}) = \frac{3}{11}$, $\varphi_i^S(\mathbf{d}) = \varphi_i^S(\mathbf{5}) = \frac{2}{11}$. The remaining cells contribute $p_{7,8}^S = 2^{-4}$.

All estimated transition probabilities of the semi-truncated characteristic are indicated in Figure 3, and the overall end-to-end probability is $2^{-67.73}$. If we compare this to the best compatible single characteristic, we get a probability of 2^{-84} (assuming 1-round key recovery, excluding the final S) or 2^{-68} (assuming 2-round key recovery, excluding the last two S, S). On the other hand, if we naively evaluate the probability of the truncated characteristic by counting the necessary cancellations, we get the much smaller probability of 2^{-100} (generously excluding initial A and final M, A, M, A, A). Of course, it should be noted that neither this truncated characteristic nor the best compatible single characteristic are necessarily optimal for MANTIS₆.

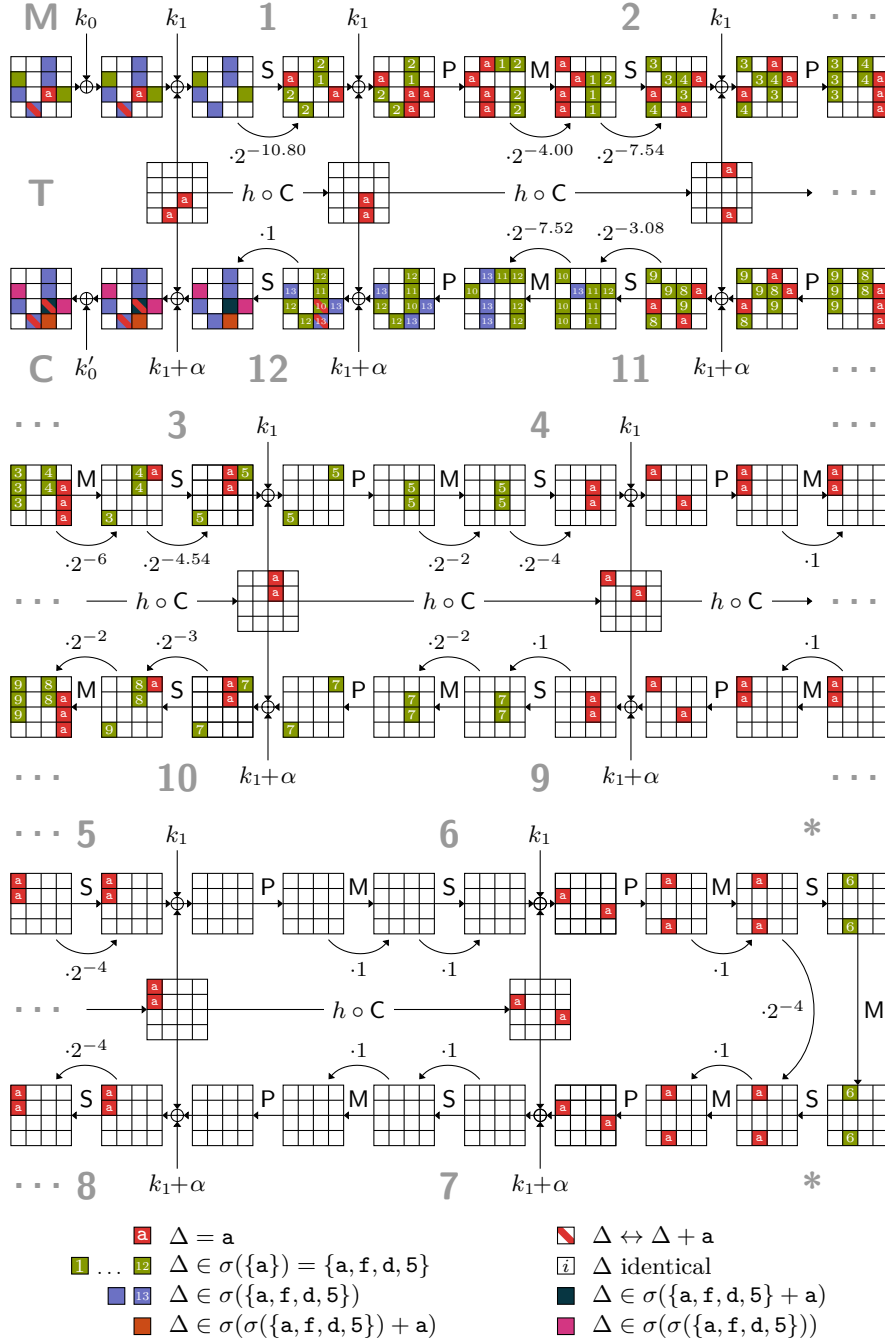


Figure 3: Family of differential characteristics for MANTIS₆.

4.2 Data Collection Phase

We need to generate about $2^{67.73}$ message pairs to have an expected number of 1 pair following the semi-truncated characteristic of Figure 3. While the trivial approach with $2 \cdot 2^{67.73}$ queries to the encryption oracle would not exhaust the codebook for this 64-bit tweakable block cipher, the resulting data-time product would exceed the attacker’s complexity bounds. Instead, we take advantage of multiple input differences.

The semi-truncated input difference covers $|\chi^M| = 4^2 \cdot 13^4 \approx 2^{18.80}$ differences and has a span of $|\langle \chi^M \rangle| = 8^2 \cdot 16^4 = 2^{22}$. Using an initial structure as described in Subsection 3.3, we can generate $|\langle \chi^M \rangle| \cdot |\chi^M| \approx 2^{40.80}$ pairs from $2 \cdot |\chi^M| \approx 2^{19.80}$ queries, giving a rate of 2^{21} pairs per query. For example, the initial structure for cells S_4, S_{11} is illustrated in Figure 4. After repeating this for about $2^{67.73} / 2^{40.80} = 2^{26.93}$ different base plaintexts, we expect 1 valid pair.

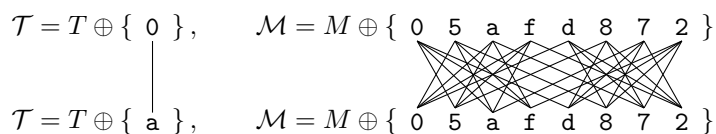


Figure 4: Initial structure for $\chi_4^M = \chi_{11}^M = \{a, f, d, 5\}$ (■) [DEKM16].

The family of characteristics given in Figure 3 has a number of conditions for valid ciphertext pairs we can use to filter generated pairs before the key guessing phase:

- (F1) Cells $S_0, S_1, S_3, S_5, S_7, S_9, S_{12}, S_{15}$ have $\Delta = 0$ ($\rho(\chi_{0,1,3,5,7,9,12,15}^C) = 2^{-4 \times 8} = 2^{-32}$)
- (F2) Cells S_2, S_6, S_8, S_{13} have $\Delta \in \sigma(\{5, a, d, f\})[+a]$ ($\rho(\chi_{2,6,8,13}^C) = 2^{-0.299 \times 4} = 2^{-1.196}$)
- (F3) Cell S_{10} has $\Delta \in \sigma(\{5, a, d, f\} + a) + a$ ($\rho(\chi_{10}^C) = 2^{-0.193}$)
- (F4) Cells S_4, S_{11} have $\Delta \in \sigma(\sigma(\{5, a, d, f\}))$ ($\rho(\chi_{4,11}^C) = 2^{-0.093 \times 2} = 2^{-0.186}$)

Combining the filtering conditions (F1), (F2), (F3), and (F4), we have a filter with probability $\rho(\chi^C) = 2^{-33.58}$ to narrow down the number of relevant pairs from $N_P = 2^{67.73}$ to $N_F = 2^{34.15}$. This step can be implemented efficiently by grouping the ciphertexts into partitions based on the values of the relevant ciphertext cells and only combining pairs in each partition. The expected number of valid pairs per base plaintext is $\ll 1$, so the overhead of generating and filtering pairs can be considered negligible. The resulting complexity is $N_Q = 2^{49.93}$ chosen ciphertext queries and accesses to a small data structure for finding a set of $N_F = 2^{34.15}$ pairs containing about 1 valid pair.

As mentioned in Subsection 3.3, we could strike a different tradeoff between N_Q and N_F by slightly modifying the input difference. One alternative possibility would be to use an input structure similar to MANTIS₅ [DEKM16] with 4 differences $\{5, a, d, f\}$ per active cell in χ^M . The resulting probability p would increase by a factor of 2^3 and decrease N_F accordingly, but the overall attack complexity also increases. Thus, we keep the original characteristic of Figure 3. An additional effect of this choice of larger sets is that the attack is more robust to differential invariance effects as observed for MANTIS₅ [DEKM16], since not all transitions are invariant with $\mathcal{S}_{a,\tau} = 0$ under a fixed τ (see Figure 2a).

4.3 Key Recovery Phase

We can now use such a set of candidate pairs to narrow down the keyspace. However, note that the end-to-end probability of $2^{-67.73}$ is smaller than the generic probability 2^{-64} of any fixed output difference, and much smaller than the generic probability of the semi-truncated difference at any step in the last rounds. Additionally, the 128-bit key is twice as large as the block size. This makes the key recovery approach somewhat more challenging. We will follow the approach sketched in Subsection 3.3.

Phase 1: Recovering 61 bits of key material from filtered pairs

The first step is the recovery of 61 total bits of key information. We check our key guesses against several constraints in the semi-truncated characteristic of Figure 3.

- **Round 1.** Guessing 24 bits of the subkey $k_0 + k_1$ allows us to compute forward until after the SubCells step in round 1, where we can check the conditions (C1), (C2), (C3) listed in Table 1. The probabilities of these conditions to hold for a filtered output pair can be evaluated with the approach of Subsection 3.2 as $2^{-14.80}$.
- **Round 12.** We can additionally guess 32 bits of the subkey $k'_0 + k_1$ and compute back before the last SubCells step in round 12, where we can check the conditions (C4), (C5), (C6), (C7) of Table 1. The total probability is $2^{-20.71}$.
- **Rounds 2 and 11.** Guessing keys for further rounds is more computationally expensive, since keyguesses depend on both keyguesses for previous rounds, as well as inactive cells in the plaintext/ciphertext, for which the key also would have to be guessed. However, due to the linear nature of the MixColumns step, we can target some cells after the SubCells step in rounds 2 and 11 with conditions (C8), (C9).










(C8) depends on the keyguesses made for (C2) and (C4), plus 4 bits $S_2 \oplus S_8 \oplus S_{13}$ of k_1 . Of these 4 bits, 3 have already been determined, so we guess only 1 bit.

(C9) depends on the keyguesses for (C5), plus 4 new bits $S_4 \oplus S_{11} \oplus S_{14}$ of k_1 .

The probability that a pre-filtered ciphertext pair follows the conditions (C1) to (C9) is estimated using the methods of Subsection 3.2, resulting in a total probability of $2^{-41.21}$. Thus, we expect that for a single ciphertext pair, out of the 2^{61} possible sub-key candidates, only $2^{61-41.21} = 2^{19.79}$ should satisfy all conditions (C1) to (C9). Repeating this process for all $2^{34.15}$ pairs results in a total of roughly $2^{19.79+34.15} = 2^{53.94}$ valid subkeys, reducing the key space by a factor of $2^{7.06}$. We need to repeat this process a total of 9 times to filter out the correct 61-bit subkey. These 9 repetitions increase the overall time and data complexity by a factor of $2^{3.17}$.

We compute the set of $2^{53.94}$ valid sub-keys for each repetition $r \in 1, 2, \dots, 9$ and finally perform a set intersection of all 9 sets to calculate the correct 61-bit subkey. Using a hash-set as a data structure this can be done with a computational complexity of $2^{53.94}$ (only the first intersection is this expensive, as the set of valid keys shrinks with each intersection performed).

Table 1: Conditions used for key recovery and their probabilities (for a pre-filtered pair).

Round	Cond.	Cells	Difference Δ	Prob.	Key bits
1	(C1)	S_4, S_{11}	 {a}	2^{-4}	8
	(C2)	S_2, S_8, S_{13}	 {5, a, d, f}, equal	$2^{-9.10}$	12
	(C3)	S_6	 {a, f}	$2^{-1.70}$	4
12	(C4)	S_2, S_8, S_{13}	 {5, a, d, f}	$2^{-9.1}$	12
	(C5)	S_4, S_{11}, S_{14}	 $\sigma(\{5, a, d, f\})$, equal [+a]	$2^{-8.11}$	12
	(C6)	S_6	 {5, a, d, f}	$2^{-1.7}$	4
	(C7)	S_{10}	 {5, a, d, f} + a	$2^{-1.8}$	4
2 and 11	(C8)	S_7 and S_7	 {a}	2^{-4}	(C2)+(C4)+1
11	(C9)	S_5	 {5, a, d, f}	$2^{-1.7}$	(C5)+4

Phase 2: Recovering 43 bits of key material from valid pairs

Using the recovered 61 bits of information about the secret key, we can further filter the $9 \times 2^{34.15}$ plaintext pairs $i \in I_r$. Since the right key misidentifies a pair as false positive with a probability of about $2^{-41.21}$, we expect that only the 9 valid pairs remain. We can now use those 9 valid pairs to recover another 43 bits of key information in two steps.

- **Round 2.** We can recover 29 bits of key material by targeting cells S_0, S_5, S_{10} of the S-Box output in Round 2, where we can verify condition (V1) listed in Table 2. These cells already depend on many key bits, as illustrated in Figure 5. Taking into account the previously recovered 61 bits, we need to guess another 29 bits of key information. Condition (V1) holds with a probability of $\approx 2^{-4.25}$, or $\approx 2^{-38.25}$ for all 9 remaining pairs, so we expect that only the correct 29-bit subkey remains.
- **Round 11.** In a similar fashion, we can recover 14 more bits by targeting cells S_6, S_{12} of the S-Box input in Round 11 and verifying condition (V2). Taking into account the previously recovered $61 + 29$ key bits, we need to guess another 14 bits. Condition (V2) holds with a probability of $\approx 2^{-2.54}$, or $\approx 2^{-22.86}$ for all 12 remaining pairs, and should uniquely determine the correct 14-bit subkey.

Table 2: Conditions used for key recovery and their probabilities (for all 9 valid pairs).

Round	Cond.	Cells	Difference Δ	Prob.	Key bits
2	(V1)	S_0, S_5, S_{10}	$\{5, a, d, f\}$, equal	$2^{9 \times 4.25}$	+29
11	(V2)	S_6, S_{12}	$\{5, a, d, f\}$, equal	$2^{9 \times 2.54}$	+14

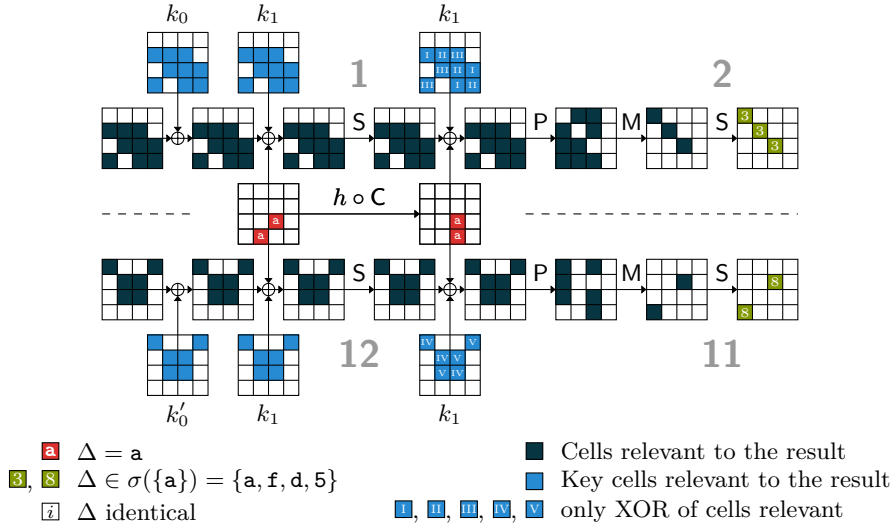


Figure 5: Cells influencing the 29-bit (top) and 14-bit (bottom) key-recovery process.

Phase 3: Recovery of k_0 and k_1 , and summary of complexities

So far, we have recovered $61 + 29 + 14$ bits of information about the key material. This results in 104 linearly independent linear equations for k_0 and k_1 . To recover the full key, we have to guess the 24 remaining bits, resulting in a complexity of 2^{24} trial encryptions.

In summary, the complexity of this attack is dominated by the first key recovery step, where we store and intersect sets of $2^{53.94}$ key candidates.

References

- [Ava17] Roberto Avanzi. The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric Even-Mansour constructions with non-involutory central rounds, and search heuristics for low-latency S-boxes. *IACR Transactions on Symmetric Cryptology*, 2017(1):4–44, 2017.
- [BBI⁺15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.
- [BCG⁺12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçın. PRINCE – A low-latency block cipher for pervasive computing applications. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 208–225. Springer, 2012.
- [BG11] Céline Blondeau and Benoît Gérard. Multiple differential cryptanalysis: Theory and practice. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *LNCS*, pages 35–54. Springer, 2011.
- [BJK⁺16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.
- [CHP⁺17] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A security analysis of Deoxys and its internal tweakable block ciphers. *IACR Transactions on Symmetric Cryptology*, 2017(3):73–107, 2017.
- [DEKM16] Christoph Dobraunig, Maria Eichlseder, Daniel Kales, and Florian Mendel. Practical key-recovery attack on MANTIS5. *IACR Transactions on Symmetric Cryptology*, 2016(2):248–260, 2016.
- [DEM16] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Square attack on 7-round Kiasu-BC. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security – ACNS 2016*, volume 9696 of *LNCS*, pages 500–517. Springer, 2016.
- [JNP14] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014.
- [Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption – FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
- [KR96] Joe Kilian and Phillip Rogaway. How to protect DES against exhaustive key search. In Neal Kobitz, editor, *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *LNCS*, pages 252–267. Springer, 1996.
- [LRW02] Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002.