Clustering Related-Tweak Characteristics: Application to MANTIS-6

Maria Eichlseder and Daniel Kales

Graz University of Technology, Graz, 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 tweakey 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 MANTIS_5 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 MANTIS_6 with probability about $2^{-67.73}$ and derive a key-recovery attack with a complexity of about $2^{55.09}$ chosen-plaintext queries and $2^{55.52}$ computations. The data-time product is $2^{110.61} \ll 2^{126}$.

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 $\tilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$. $\tilde{E}$ 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$^+$17] showed that if this is not considered in the search, the resulting minimal characteristics are often invalid, and that tighter bounds can be obtained by adapting the search model accordingly. Dobraunig et al. [DEKM16], on the other hand, take advantage of the predictable tweakey schedule to cluster several differential characteristics with nearly optimal probability for an attack on MANTIS$_5$.

Our contributions

We generalize the clustering approach from the attack on MANTIS$_5$ [DEKM16] and describe a tool (`https://github.com/dkales/clusterfk`) for efficiently finding and evaluating such clusters. Whereas the cluster for MANTIS$_5$ was found by hand and was simple enough for its probability to be evaluated on a cell-by-cell basis, we argue that such probability estimates are not sufficiently 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, like [DEKM16], 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 want to efficiently estimate the probability of the semi-truncated characteristic without enumerating all individual characteristics or round differentials, since some steps contain many active S-boxes. [DEKM16] proposed a simple estimate similar to the probability of a truncated characteristic, but for more complex characteristics, this is not sufficiently accurate. 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.

Our approach combines 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, and incurs significant costs from the linear constraints in the tweak schedule. We discuss how such semi-truncated characteristics can be applied to obtain efficient key-recovery attacks, and analyze the complexity and possible tradeoffs.

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 $2^{55.09}$ chosen-plaintext queries and $2^{55.52}$ computations. The data-time product of $2^{110.61}$ 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 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. In Section 5, we perform experiments to verify our theoretical results and discuss the results of the experiments and their impact on the 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 $M = M_0\|M_1\|\cdots\|M_{15}$ 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, where $S_0, \ldots, S_3$ 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 $\mathsf{S} \circ \mathsf{M} \circ \mathsf{S}$ (Figure 1a). 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.
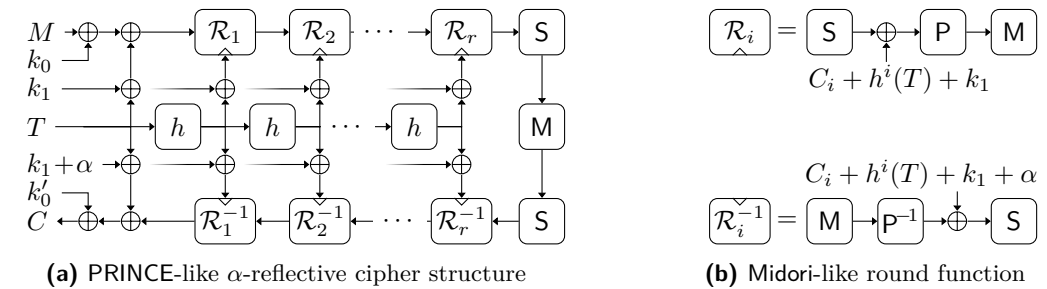

(a) PRINCE-like α-reflective cipher structure

(b) Midori-like round function

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].
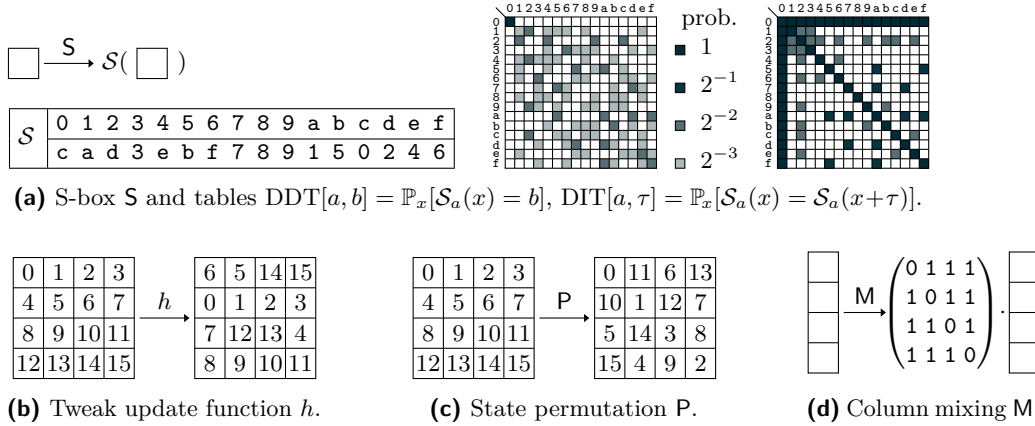
(a) S-box S and tables $\mathrm{DDT}[a,b] = \mathbb{P}_x[\mathcal{S}_a(x) = b]$, $\mathrm{DIT}[a,\tau] = \mathbb{P}_x[\mathcal{S}_a(x) = \mathcal{S}_a(x+\tau)]$.

(b) Tweak update function h.   **(c)** State permutation P.   **(d)** Column mixing M.

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 a, 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 MANTIS$_5$ for an attacker constrained to $D \leq 2^{30}$ chosen plaintexts or $D \leq 2^{40}$ known plaintexts, and for MANTIS$_7$ without further constraints besides the data-time product $D \cdot T < 2^{126}$.

Dobraunig et al. [DEKM16] refuted the claim for MANTIS$_5$ 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 near-optimal differential characteristic. Finally, they collect many more closely related 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, 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 collective 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 MANTIS$_5$, 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 cell-by-cell basis as follows. For each S-box in SubCells, they compute the average transition probability for (uniformly distributed) input differences in the input set, summed over the output differences in the output set. For MixColumns, they only consider specific transitions that can be described by equality constraints for the input differences drawn (uniformly randomly) from the identical input sets in one column, similar to truncated characteristics. A practical verification confirmed that the estimates are sufficiently accurate to practically perform the attack with the given characteristic, although the authors observed small deviations in some steps.

However, when we try to extend the MANTIS$_5$ approach 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 MANTIS$_5$ 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

Like [DEKM16], we want to cluster several characteristics that follow the same truncated characteristic, and want to describe them in terms of cell-wise sets. More specifically, we want to compactly characterize the set of all differential characteristics compatible with several constraints configured by the cryptanalyst:

- **Truncated constraints:** A truncated characteristic serves as a starting point. We chose 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 consistent differential characteristics compatible with these constraints that we can specify by listing possible cell differences for each state cell, and considering the direct product of all these sets. Without the second and third constraint, this superset would just be the truncated characteristic itself, where each cell permits either only the zero difference or any difference (except maybe for the first/last round); but with the constraints of a linear tweak schedule and a fixed tweak difference, this superset is significantly reduced (as estimated by the "degrees of freedom" in [CHP+17]), and we refer to it as a semi-truncated characteristic. We refer to all individual characteristics in this superset as "compatible", and use "consistent" or "possible" to mean characteristics with non-zero probability.

To find this semi-truncated characteristic, we start with a model of the cipher based on the initial constraints given by the truncated characteristic, tweak difference, and optionally input/output differences as discussed above. For each operation of the cipher, we define necessary conditions (constraints) for consistent input and output difference sets imposed by the differential behaviour of the operation. We then look at each target state between two operations in the cipher and apply an iterative constraint propagation to the two neighbouring states. The constraint propagation for each operation is described in the following and potentially reduces the input and output difference sets if they contain unreachable differences. Afterwards, the target state is marked as "reduced", whereas the two neighbouring states are marked "not reduced" if they were changed by the update. This is repeated until no more changes are observed during a round of propagation and all states are "reduced", resulting in the reduced semi-truncated characteristic.

Consider the state geometry and operations of MANTIS as an example. For each intermediate state $S = (S_0, \ldots, S_{15})$, let $\chi = (\chi_0, \ldots, \chi_{15})$ with $\chi_i \subseteq \mathcal{X} = \{0, \ldots, \mathtt{f}\}$ denote the set of differences specified by the semi-truncated characteristic. We now

consider an operation $f \in \{\mathsf{S}, \mathsf{A}, \mathsf{P}, \mathsf{M}\}$ 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 DDT of the S-box defines the relation between consistent input and output differences. We define the corresponding set transition function $\sigma : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$, which maps a set of input differences $X \subseteq \mathcal{X}$ to the set of reachable output differences:

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

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

$$\chi_i^{\mathsf{S}} \leftarrow \chi_i^{\mathsf{S}} \cap \sigma(\chi_i)\,, \qquad\qquad i = 0, \ldots, 15\,,$$
$$\chi_i \leftarrow \chi_i \cap \sigma(\chi_i^{\mathsf{S}})\,, \qquad\qquad i = 0, \ldots, 15\,.$$

PermuteCells (P): Let $[\mathsf{P}(\chi)]_i$ denote cell i of the permuted input characteristic χ. The output set must equal the permuted input set, so we update

$$\chi_i^{\mathsf{P}} \leftarrow \chi_i^{\mathsf{P}} \cap [\mathsf{P}(\chi)]_i\,, \qquad\qquad i = 0, \ldots, 15\,,$$
$$\chi_i \leftarrow \chi_i \cap [\mathsf{P}^{-1}(\chi^{\mathsf{P}})]_i\,, \qquad\qquad i = 0, \ldots, 15\,.$$

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

$$\chi_i^{\mathsf{A}} \leftarrow \chi_i^{\mathsf{A}} \cap (\chi_i \oplus \tau_i)\,, \qquad\qquad i = 0, \ldots, 15\,,$$
$$\chi_i \leftarrow \chi_i \cap (\chi_i^{\mathsf{A}} \oplus \tau_i)\,, \qquad\qquad i = 0, \ldots, 15\,.$$

MixColumns (M): The possible transitions are analyzed column by column. Consider the column with cell indices $I = (I_0, I_1, I_2, I_3) \in \mathcal{I} = \{(0, 4, 8, 12), \ldots, (3, 7, 11, 15)\}$ and the corresponding column characteristic $\chi_I = \chi_{I_0} \times \ldots \times \chi_{I_3}$. Since M is linear and involutive, we update

$$\chi_{I_j}^{\mathsf{M}} \leftarrow \chi_{I_j}^{\mathsf{M}} \cap \{[\mathsf{M} \cdot \delta]_j \mid \delta \in \chi_I,\ \mathsf{M} \cdot \delta \in \chi_I^{\mathsf{M}}\}\,, \qquad I \in \mathcal{I},\ j = 0, \ldots, 3\,,$$
$$\chi_{I_j} \leftarrow \chi_{I_j} \cap \{[\mathsf{M} \cdot \delta]_j \mid \delta \in \chi_I^{\mathsf{M}},\ \mathsf{M} \cdot \delta \in \chi_I\}\,, \qquad I \in \mathcal{I},\ j = 0, \ldots, 3\,,$$

where $[\mathsf{M} \cdot \delta]_j$ denotes the value of cell $j \in \{0, \ldots, 3\}$ after applying the MixColumns matrix to the column difference $\delta \in \chi_I$ (or χ_I^{M}).

For the specific case of transitions that match the branch number bound for the MANTIS matrix, these update conditions can be simplified [DEKM16]: All valid transitions must have the same difference in all 4 active input/output cells, so all active sets must be equal. Let $\chi_I^* = ([\chi_I^*]_0, \ldots, [\chi_I^*]_3)$ refer to the 4 active cells out of the 8 input/output column cells $\chi_I \| \chi_I^{\mathsf{M}}$ for column I, then we only update

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

The updates are iterated 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. In the following, "semi-truncated characteristic" always refers to a reduced characteristic. Next, we want to estimate the probability of such a reduced semi-truncated characteristic, and see how further constraints impact the resulting attacks.

Comparison with [DEKM16].   The attack on MANTIS$_5$ describes a cluster of differential characteristics which are all compatible with a fixed truncated characteristic and tweak difference, and can be described with a set of differences per cell. We target the same type of clusters. However, that cluster was found by hand by starting from a single characteristic and gradually adding other differences in individual cells that produce similar characteristics with the same probability. As a result, the cluster in [DEKM16, Figure 5] is neither reduced (in Round 10) nor fully expanded to include all characteristics compatible with the constraints identified above (in Rounds 1 and 2, mentioned as an observation in the practical verification). We describe and implement a general approach for finding good clusters of this type.

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 MANTIS$_5$ [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 make 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 MANTIS$_5$, 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 $\Delta = (\Delta_0, \ldots, \Delta_{15})$. 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:

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

Now consider an operation $f \in \{\mathsf{S}, \mathsf{A}, \mathsf{P}, \mathsf{M}\}$ 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:

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

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 $\delta \notin \chi_i^f$):

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

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

$$\tilde{\varphi}_i^\mathsf{S}(\delta) = \sum_{\alpha \in \chi_i} \varphi_i(\alpha) \cdot \mathbb{P}\big[\alpha \xrightarrow{\mathcal{S}} \delta\big], \qquad\qquad p_i^\mathsf{S} = \sum_{\delta \in \chi_i^\mathsf{S}} \tilde{\varphi}_i^\mathsf{S}(\delta),$$

$$\varphi_i^\mathsf{S}(\delta) = \mathbb{1}_{\chi_i^\mathsf{S}}(\delta) \cdot \frac{\tilde{\varphi}_i^\mathsf{S}(\delta)}{p_i^\mathsf{S}}, \qquad\qquad p^\mathsf{S} = \prod_i p_i^\mathsf{S}.$$

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 φ_I^M of column differences $\Delta_I = (\Delta_{I_0}, \ldots, \Delta_{I_3})$ and (dependent) cell distributions $\varphi_{I_j}^\mathsf{M}$:

$$\tilde{\varphi}_I^\mathsf{M}(\delta_I) = \varphi_I(\mathsf{M}^{-1} \cdot \delta_I) = \prod_j \varphi_{I_j}([\mathsf{M}^{-1} \cdot \delta_I]_j), \qquad\qquad p_I^\mathsf{M} = \sum_{\delta_I \in \chi_I^\mathsf{M}} \tilde{\varphi}_I^\mathsf{M}(\delta_I),$$

$$\varphi_I^\mathsf{M}(\delta_I) = \mathbb{1}_{\chi_I^\mathsf{M}}(\delta_I) \cdot \frac{\tilde{\varphi}_I^\mathsf{M}(\delta_I)}{p_I^\mathsf{M}} \quad \Rightarrow \quad \varphi_{I_j}^\mathsf{M}(\delta) = \sum_{[\delta_I]_j = \delta} \varphi_I^\mathsf{M}(\delta_I), \qquad p^\mathsf{M} = \prod_I p_I^\mathsf{M}.$$

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 φ_*^M. For example, in the simplest case that the input cells are also identically (independently) distributed by some φ_*, any bias in this φ_* will be "amplified" if $a > 1$:

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

The independence assumptions we made at each step will usually not be satisfied. A solution would be to keep track of and sum over the full-state distribution φ for each state (within the constraints of χ), but this is not practicable for steps with too many active cells. As a practical compromise, we consider the dependencies φ_I^M introduced by MixColumns in the next SubCells, but assume that the following PermuteCells "clears" the dependencies by reducing column-wise distributions to their cell-wise marginal distributions:

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

Comparison with related work. In contrast to the set-based approach in [DEKM16], which is mostly useful in case of equiprobable characteristics and simple MixColumns transitions, we consider the distribution of differences within a set and within a column.

This allows a broader application to general characteristics and MixColumns matrices. We will later also use it in backwards direction starting from the ciphertext for key recovery.

Related approaches have also been applied to evaluate clusters of differential characteristics in attacks on classical block ciphers without enumerating all characteristics. Canteaut et al. [CFG$^+$14] identify clusters of characteristics following structured "activity patterns" in PRINCE that can be evaluated with a multiple differential attack. For the specific iterative differential structure in PRINCE, they are able to derive a closed representation of the probability from a description of the differential behaviour using transition matrices, more specifically, a Kronecker product of the matrices describing the S-box and the linear layer. The result is a quadratic submatrix of the full differential propagation matrix, where the latter describes the transition probability of any full-state input difference to any output difference, whereas the submatrix covers only the differences included in the cluster. The product of these submatrices then describes the transition probabilities contributed by all characteristics whose intermediate differences follow the cluster. A similar approach using sparse matrix multiplication is described in more general terms by Leurent [Leu15] and Biryukov et al. [BDP15] in the analysis of the closely related ciphers LBlock-s and TWINE. This could also be applied for our characteristics as long as the number of differences per state is sufficiently low; in particular, for MANTIS$_6$, it is not feasible to apply it to the unreduced truncated characteristic, but may be feasible for the reduced semi-truncated characteristic with a careful implementation. However, to avoid storing (and summing) the exact distribution of full-state differences as an intermediate result after every operation, we choose to implement the approximation described in this section: to compress the distribution to column level (with SubCells ∘ MixColumns) and then to cell level (with PermuteCells) by taking the marginal distributions instead of the full joint distribution of the cell differences. Another, more superficial difference is that we separate the success probability p^f of and $\overline{p}^f$ up to operation f from the resulting distribution φ_f, whereas [Leu15, CFG$^+$14] consider the combined transition probabilities $\overline{p}^f \cdot \varphi_f$ from some starting difference.

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 $\chi = (\chi_0, \ldots, \chi_{15})$ with the corresponding expanded set of differences $\chi_0 \times \cdots \times \chi_{15} \subseteq \mathcal{X}^{16}$. 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

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

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

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 [BG11]: 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:

$$\mathcal{T} = T \oplus \left\langle \chi^T \right\rangle = \{T, T \oplus \delta^T\}, \qquad \mathcal{M} = M \oplus \left\langle \chi^M \right\rangle ,$$

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

$$N_F = R \cdot \frac{\rho(\chi^C)}{p} .$$

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 remainder of the 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 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 pair. 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 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.

Comparison with related work. Compared to standard key recovery based on counting and ranking [BS90, Sel08], we interleave the key recovery more closely with the characteristic. The result in our case is that we can recover the key with much less memory, but we need to re-use the probability computations to obtain useful estimates. [DEKM16] used a similar approach, but with a less accurate computation based on truncated estimates and a less efficient filter. We also provide a more general description of the initial structure.

4   Application to MANTIS$_6$

4.1   Finding a Semi-Truncated Characteristic for MANTIS$_6$

We can now apply this approach to find a semi-truncated characteristic for MANTIS$_6$. 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 (a, a), the same as for MANTIS$_5$. 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 characteristic 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 3b. 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.

Figure 3b 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 $\chi_i = \{\mathtt{a}, \mathtt{f}, \mathtt{d}, \mathtt{5}\}$. To analyze the transition probability of these three cells, consider each of the four possible differences in turn. Difference $\mathtt{a}$ will be mapped by SubCells to compatible differences in $\chi_{6,10,14}^{\mathsf{S}}$ with probability $p_{6,10,14}^{\mathsf{S}} = 1 \cdot 1 \cdot \frac{1}{4}$, and the differences in $S_6^{\mathsf{S}}, S_{10}^{\mathsf{S}}$ will be uniformly distributed (25 % each for $\mathtt{a}$, $\mathtt{f}$, $\mathtt{d}$, $\mathtt{5}$). Difference $\mathtt{f}$ has $p_{6,10,14}^{\mathsf{S}} = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{4}$, and a biased output distribution (50 % $\mathtt{a}$, 50 % $\mathtt{f}$). Differences $\mathtt{d}$ and $\mathtt{5}$ each have $p_{6,10,14}^{\mathsf{S}} = \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4}$, and produce 100 % $\mathtt{a}$. 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 $\varphi_i^{\mathsf{S}}(\mathtt{a}) = \frac{4}{8}$, $\varphi_i^{\mathsf{S}}(\mathtt{f}) = \frac{2}{8}$, $\varphi_i^{\mathsf{S}}(\mathtt{d}) = \varphi_i^{\mathsf{S}}(\mathtt{5}) = \frac{1}{8}$. The remaining cells contribute $p_{7,8} = 2^{-4}$. If we follow the distribution of cell S_{10}^{S} (◼) through MixColumns (note that the other two cells marked ◼ are uniformly distributed), we arrive at the same distribution in S_{12} at the input of SubCells in Round 3. There, it contributes $p_{12}^{\mathsf{S}} = \frac{11}{16} \approx 2^{-0.54}$ and produces $\varphi_{12}^{\mathsf{S}}(\mathtt{a}) = \frac{4}{11}$, $\varphi_{12}^{\mathsf{S}}(\mathtt{f}) = \frac{3}{11}$, $\varphi_{12}^{\mathsf{S}}(\mathtt{d}) = \varphi_{12}^{\mathsf{S}}(\mathtt{5}) = \frac{2}{11}$.

All estimated transition probabilities of the semi-truncated characteristic are indicated in Figure 3b, 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 final 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, neither this truncated characteristic nor the best single characteristic are necessarily optimal for MANTIS$_6$.

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 3b. Based on a practical evaluation of the success probability in Subsection 5.2, we want an expected number of 3 pairs following the semi-truncated characteristic, and therefore need to generate $3 \cdot 2^{67.73}$ pairs. While the trivial approach with $2 \cdot 3 \cdot 2^{67.73}$ encryption queries 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 (which coincides with the structure in [DEKM16] in two cells), we can generate $|\langle \chi^M \rangle| \cdot |\chi^M| \approx 2^{40.80}$ pairs from $2 \cdot |\langle \chi^M \rangle| \approx 2^{23}$ queries, giving a rate of $2^{17.80}$ pairs per query (Figure 3a). After repeating this for about $3 \cdot 2^{67.73}/2^{40.80} = 2^{28.51}$ different base plaintexts, we expect 3 valid pairs.

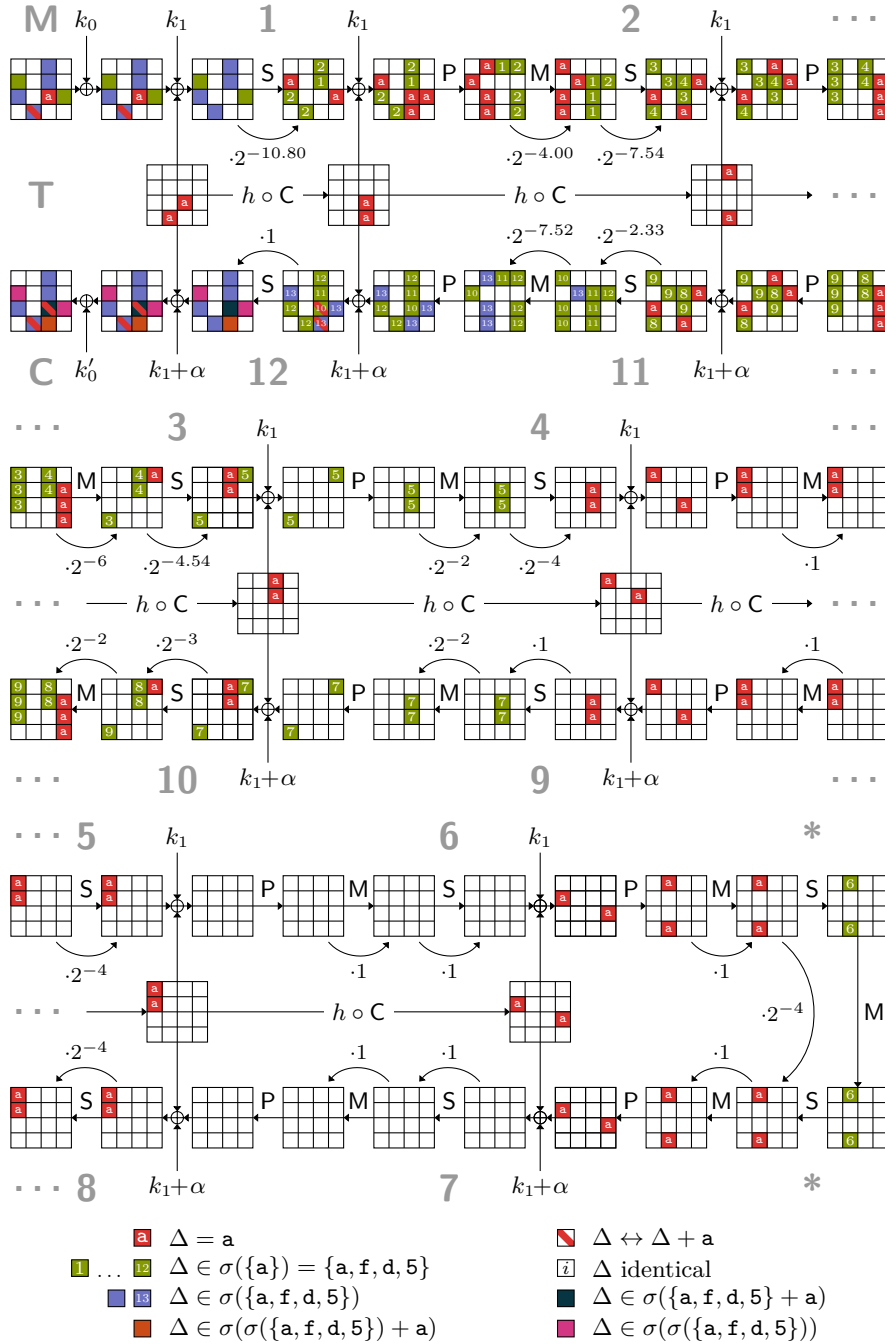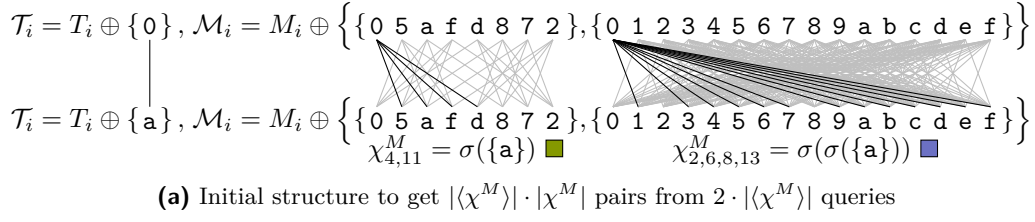$$\mathcal{T}_i = T_i \oplus \{0\}, \mathcal{M}_i = M_i \oplus \left\{ \{0\ 5\ a\ f\ d\ 8\ 7\ 2\}, \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ a\ b\ c\ d\ e\ f\} \right\}$$

$$\mathcal{T}_i = T_i \oplus \{a\}, \mathcal{M}_i = M_i \oplus \left\{ \{0\ 5\ a\ f\ d\ 8\ 7\ 2\}, \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ a\ b\ c\ d\ e\ f\} \right\}$$

$$\chi_{4,11}^{M} = \sigma(\{a\}) \quad\quad \chi_{2,6,8,13}^{M} = \sigma(\sigma(\{a\}))$$

(a) Initial structure to get $|\langle \chi^M \rangle| \cdot |\chi^M|$ pairs from $2 \cdot |\langle \chi^M \rangle|$ queries

(b) Semi-truncated differential characteristic and probability

Figure 3: Attack on MANTIS$_6$

The family of characteristics given in Figure 3b 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 $\Delta = 0$ ($\rho(\chi^C_{0,1,3,5,7,9,12,15}) = 2^{-4 \times 8} = 2^{-32}$)

(F2) Cells S_2, S_6, S_8, S_{13} have $\Delta \in \sigma(\{5, a, d, f\})[+a]$ ($\rho(\chi^C_{2,6,8,13}) = 2^{-0.299 \times 4} = 2^{-1.196}$)

(F3) Cell S_{10} has $\Delta \in \sigma(\{5, a, d, f\} + a) + a$ $\hfill (\rho(\chi^C_{10}) = 2^{-0.193})$

(F4) Cells S_4, S_{11} have $\Delta \in \sigma(\sigma(\{5, a, d, f\}))$ $\hfill (\rho(\chi^C_{4,11}) = 2^{-0.093 \times 2} = 2^{-0.186})$

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 $N_P = 3 \cdot 2^{67.73}$ to $N_F = 2^{35.73}$. 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 $N_Q = 2^{51.51}$ chosen ciphertext queries and accesses to a small data structure for finding a set of $N_F = 2^{35.73}$ pairs containing about 3 valid pairs.

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 exclude input differences with low probability and reduce the span per base plaintext accordingly, for example to the same set as $\chi^M_{4,11} = \sigma(\{a\})$ as in MANTIS$_5$ [DEKM16] with 4 differences 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. If we only need very few base plaintexts, this change would also improve the query granularity and possibly the variance in the number of solutions per repetition, but we already have a very large number of base plaintexts in this attack anyway. Thus, we keep the original characteristic of Figure 3b. An additional effect of this choice of larger sets is that the attack is more robust to differential invariance effects as observed for MANTIS$_5$ [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 the filtered pairs to narrow down the keyspace (Subsection 3.3). 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.

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 3b (Table 1).

- **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 $2^{35.73}$ pairs results in a total of roughly $2^{19.79+35.73} = 2^{55.52}$ valid subkeys, reducing the key space by a factor of $2^{5.48}$. We need to repeat this process a total of 12 times to filter out the correct 61-bit subkey. These 12 repetitions increase the overall time and data complexity by a factor of $2^{3.58}$.

To evaluate the complexity of this approach, we extend the bundle approach of [DEKM16]. For each of the conditions (C1) to (C7), we can perform independent key guesses, and the resulting list of $2^{20.49}$ key candidates for the 56-bit subkey will be structured accordingly as a direct product of sublists of 8-bit, 12-bit, ..., 4-bit subkeys based on (C1), (C2), ..., (C7). The maximum sublist size is about 2^4 from (C1). Then, for each pair, we can evaluate conditions (C8) and (C9), which have dependencies. By combining the sublists for (C2)+(C4) (for (C8)) and (C5) (for (C9)) and guessing the few additional bits, we have about $\approx 2^8$ candidates to test per pair. The result is another sublist to be combined with the existing lists. The total time to generate these lists is less than 2^{52}, based on the total number of pairs, repetitions, and maximum subkey size.

Per repetition $r \in 1, 2, \ldots, 12$, we now have a set with one bundle of sublists per pair, and need to compute the intersections. The straightforward approach is to now expand each bundle to get a full set of $2^{55.52}$ valid sub-keys for each repetition, and finally perform a set intersection of all 12 sets to calculate the correct 61-bit subkey. (One could also consider an advanced approach where only the lists of valid subkeys per pair are stored, and then only combined when intersecting different iterations. This approach can lower the memory requirements for some parts of this phase.) Using a hash-set as a data structure this can be done with a computational complexity of $2^{55.52}$ (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	$\{\mathtt{a}\}$	2^{-4}	8
	(C2)	S_2, S_8, S_{13}	2	$\{\mathtt{5,a,d,f}\}$, equal	$2^{-9.10}$	12
	(C3)	S_6	1	$\{\mathtt{a,f}\}$	$2^{-1.70}$	4
12	(C4)	S_2, S_8, S_{13}	12	$\{\mathtt{5,a,d,f}\}$, equal	$2^{-9.1}$	12
	(C5)	S_4, S_{11}, S_{14}	13	$\sigma(\{\mathtt{5,a,d,f}\})$, equal $[+\mathtt{a}]$	$2^{-8.11}$	12
	(C6)	S_6	11	$\{\mathtt{5,a,d,f}\}$	$2^{-1.7}$	4
	(C7)	S_{10}	10	$\{\mathtt{5,a,d,f}\} + \mathtt{a}$	$2^{-1.8}$	4
2 and 11	(C8)	S_7 and S_7	a	$\{\mathtt{a}\}$	2^{-4}	(C2)+(C4)+1
11	(C9)	S_5	9	$\{\mathtt{5,a,d,f}\}$	$2^{-1.7}$	(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 $12 \times 2^{35.73}$ plaintext pairs $i \in I_r$. Since the correct key misidentifies a pair as false positive with a probability of about $2^{-41.21}$, we expect that only the ≥ 12 valid pairs remain. We can now use those 12 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 4. 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 $\approx 2^{-51}$ for all 12 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 $\approx 2^{-30.48}$ 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 12 valid pairs).

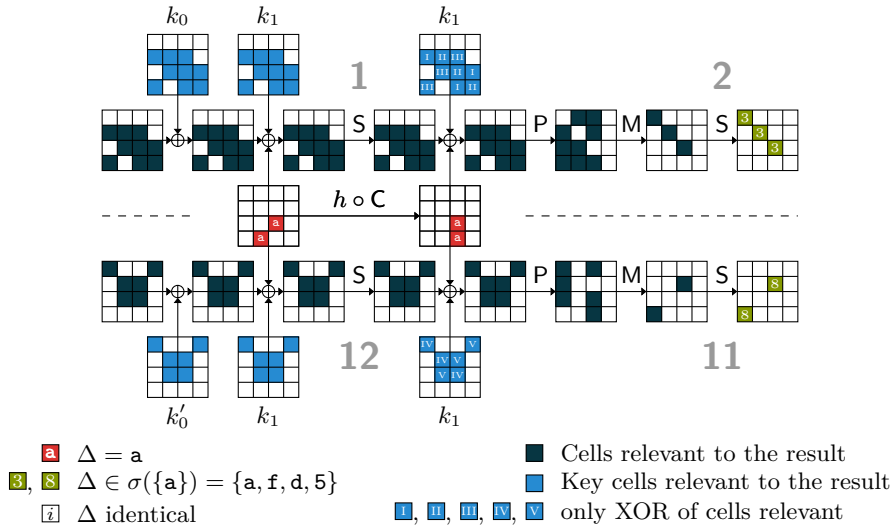| Round	Cond.	Cells		Difference Δ	Prob.	Key bits
2	(V1)	S_0, S_5, S_{10}	3	$\{\texttt{5}, \texttt{a}, \texttt{d}, \texttt{f}\}$, equal	$2^{-4.25 \times 12}$	$+29$
11	(V2)	S_6, S_{12}	8	$\{\texttt{5}, \texttt{a}, \texttt{d}, \texttt{f}\}$, equal	$2^{-2.54 \times 12}$	$+14$

Figure 4: 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 $2^{55.52}$ key candidates.

5   Experimental Verification

5.1   Experiments on the Probability of the MANTIS$_6$ Characteristic

We verified the probabilities of the semi-truncated characteristic in Figure 3b using trial encryptions with random base plaintext and tweak pairs (matching the difference in the semi-truncated characteristic). Due to the nature of the involved probabilities, we could not verify the full semi-truncated characteristic. For a single fixed key, we verified the round-reduced semi-truncated characteristic for $\approx 2^{18.5}$ base plaintexts, corresponding to $\approx 2^{59.3}$ pairs. Additionally, we observed the behaviour for $\approx 2^{16.6}$ random keys, with one base plaintext per key, corresponding to $\approx 2^{57.4}$ pairs.

Speeding up the first round. We can use an observation to speed up the experimental verification, allowing us to skip the computations up until and including the first S-Box step. For a single base plaintext we can observe the following: A uniform distribution of the base plaintext values will result in predictable distributions of the following cells after the constraints of the first S-Box step:

- Cells $S_0, S_1, S_3, S_5, S_7, S_9, S_{10}, S_{12}, S_{14}, S_{15}$ ($\square \to \square$). Since these cells are not active it is straightforward to see that a uniform distribution of the values of these cells in the input and tweak will lead to a uniform distribution after the first S-Box layer.

- Cells S_2, S_6, S_8, S_{13} ($\blacksquare \to \blacksquare$). Considering the allowed differences in the input set ($\blacksquare$), querying the span of these differences and combining all valid differences also results in a uniform distribution of all allowed value pairs in the output cells.

- Cells S_4, S_{11} ($\blacksquare \to \blacksquare$). Due to the nature of the initial structure for these cells (see Figure 3a), the resulting distribution of the cells S_4, S_{11} after the S-Box layer is not uniform over all possible values. In fact, when enumerating all possible values for these cells, we discover that there are two distinct distributions possible. Depending on the base plaintext chosen, valid pairs are either in the set $\{(1, 11), (2, 8), (6, 12), (7, 13)\}$ or $\{(0, 10), (3, 9), (4, 14), (5, 15)\}$, with uniformly distributed values inside these sets.

This allows us to skip the first round in the experimental verification. Instead of sampling the base plaintext and tweak at random, we only sample the tweak and state after the first S-Box layer. We pick one of the two distributions for cells S_4, S_{11} each, and sample the rest of the cells from a uniform distribution of all possible values. Then all possible differences are added to this state, resulting in 2^{30} pairs per base plaintext. We verified that these pairs are indeed equal to the pairs leftover after the first S-Box round when starting with an equivalent base plaintext. This process allows us to save a factor of $2^{10.8}$ computations when verifying the characteristic.

    As we can see in Table 3, the experimental results align closely with the theoretical estimates. Figure 5 shows both the observed total probability $\overline{p}_f$ up to a given operation f as well as the deviation of the observed individual transition probability p^f from the estimate in Figure 3b. The biggest deviation from the expected probability is in the S-Box layer of round 3, with a factor of $2^{0.26}$ ($2^{0.18}$) for a single (random) key(s). The large deviation at the later steps (Rounds 10+) is explained due to the small number of valid pairs that reached this step in our experiments.

5.2   Experiments on the Success Probability

The key-recovery attack in Section 4 depends on the fact that for a given set of $2^{67.73}$ plaintext-tweak pairs we expect at least one pair to follow the semi-truncated differential characteristic in Figure 3b. Again, due to the large number of computations involved,
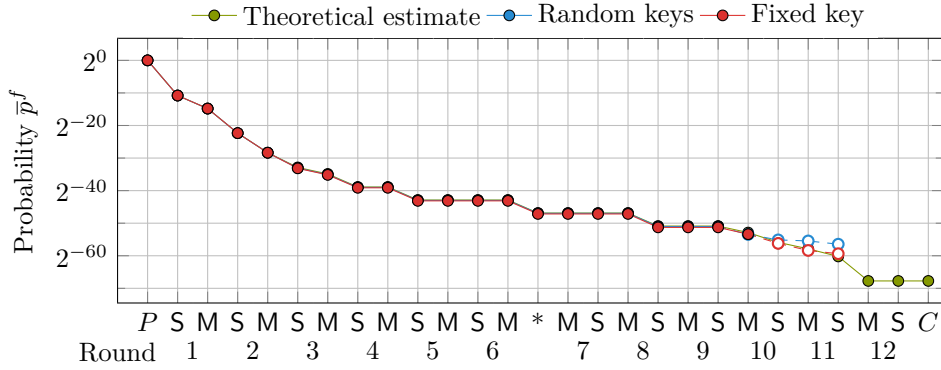
(a) Probability $\overline{p}^f$ of a pair following the semi-truncated characteristic.
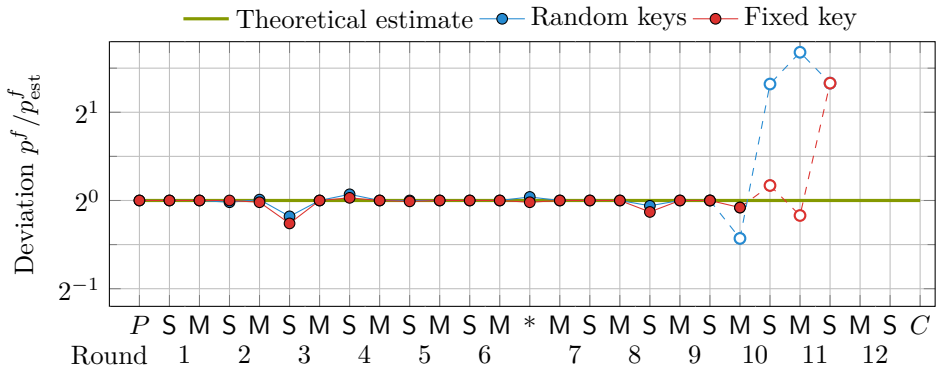
(b) Deviation of observed transition probability p^f from estimate p^f_{est}.

Figure 5: Experimental verification of estimated probability for a single fixed key ──●── and random keys ──●──. Empty marks ──○── ,──○── indicate a low sample size (< 10).
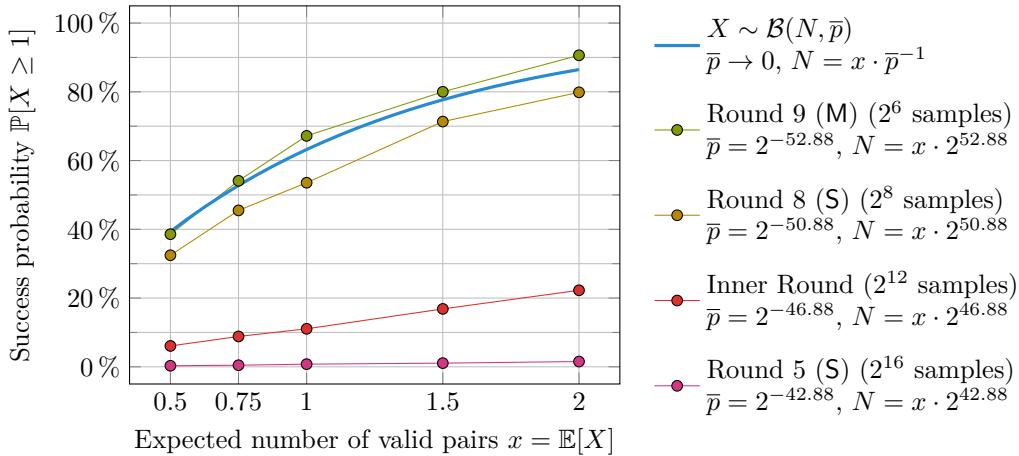

Figure 6: Experimentally observed success probability $\mathbb{P}[X \geq 1]$ of at least one valid pair within a run of N pairs, where X is the number of valid pairs up to some round in a run. The number of pairs $N \approx x \cdot \overline{p}^{-1}$ is chosen such that $x = \mathbb{E}[X] \in \{0.5, 0.75, 1, 1.5, 2\}$, based on an estimated probability $\overline{p}$ of a pair to be valid.

Table 3: Experimental verification of estimated probabilities p^f. Marks '∘' indicate a low sample size (< 10). Empty values '—' indicate that no valid pairs were found for this step.

Round	Step	Theoretical	Fixed key	Random keys
1	S	-10.80	-10.80	-10.80
	M	-4.00	-4.00	-4.00
2	S	-7.54	-7.54	-7.56
	M	-6.00	-6.02	-5.99
3	S	-4.54	-4.80	-4.72
	M	-2.00	-2.00	-2.00
4	S	-4.00	-3.97	-3.93
5	S	-4.00	-4.01	-4.00
*	S ∘ M ∘ S	-4.00	-4.02	-3.96
8	S	-4.00	-4.13	-4.06
10	M	-2.00	-2.08	-2.43 ∘
	S	-3.00	-2.83 ∘	-1.86 ∘
11	M	-2.00	-2.17 ∘	-0.32 ∘
	S	-2.33	-1.00 ∘	-1.00 ∘
12	M	-7.52	—	—

we cannot verify the success probability of the full attack. Instead we look at several round-reduced versions of the semi-truncated characteristic, removing rounds from the end. In our experiments, we choose a number of base-plaintexts that lead to an expected value of $\{0.5, 0.75, 1, 1.5, 2\}$ valid pairs per run. In Figure 6 we can observe the probability of at least one pair following the semi-truncated characteristic up until the given round. The probability that at least one valid pair is contained in a run increases when we are closer to the full semi-truncated characteristic. This is a result of the larger number of base-plaintexts needed to reach an expected value of one surviving pair, reducing the overall variance of the results (in earlier rounds the average number of valid pairs is as expected, however the valid pairs form clusters, leading to many runs without valid pairs).

As can be seen in Figure 6, the probability of a pair being valid is approaching the expected probability of a binomial distribution and is almost identical for the later rounds. Therefore we estimate the overall success probability for one repetition based on a theoretical binomial distribution. Using a number of pairs N that results in $x = \mathbb{E}[X] = 1$ expected pairs would give a success probability of $\mathbb{P} \approx 0.63$ per repetition, and an overall success probability of only $\mathbb{P}_{\text{attack}} \approx 0.016$ after 9 repetitions, which is not practical. Therefore we choose to increase the number of starting pairs so that a number of $x = \mathbb{E}[X] = 3$ pairs is expected per repetition. This results in a probability of $\mathbb{P} \approx 0.95$ per repetition, and an overall success probability of $\mathbb{P}_{\text{attack}} \approx 0.54$ after 12 repetitions. The increased number of base plaintexts increases the overall data complexity and in turn also the computational complexity of the attack by a factor of 3 when compared to the approach with one expected valid pair. The attack complexities throughout this paper already include this factor.

Acknowledgements

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.

[BDP15]   Alex Biryukov, Patrick Derbez, and Léo Perrin. Differential analysis and meet-in-the-middle attack against round-reduced TWINE. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 3–27. Springer, 2015.

[BG11]   Céline Blondeau and Benoît Gérard. Multiple differential cryptanalysis: Theory and practice. In Antoine Joux, editor, *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.

[BS90]   Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO 1990*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

[CFG+14]   Anne Canteaut, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard. Multiple differential cryptanalysis of round-reduced PRINCE. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 591–610. Springer, 2014.

[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, *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, *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 Koblitz, editor, *CRYPTO 1996*, volume 1109 of *LNCS*, pages 252–267. Springer, 1996.

[Leu15]   Gaëtan Leurent. Differential forgery attack against LAC. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 217–224. Springer, 2015.

[LRW02]   Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002.

[Sel08]   Ali Aydın Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008.