

Zero-Correlation Attacks on Tweakable Block Ciphers with Linear Tweakey Expansion

Ralph Ankele^{1,4}, Christoph Dobraunig^{2,3}, Jian Guo⁴, Eran Lambooij⁵,
Gregor Leander⁶ and Yosuke Todo⁷

¹ Royal Holloway University of London, UK

² Graz University of Technology, Austria

³ Digital Security Group, Radboud University, Nijmegen, The Netherlands

⁴ Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore

⁵ University of Haifa, Israel

⁶ Ruhr-Universität Bochum, Germany

⁷ NTT Secure Platform Laboratories, Japan

ralph.ankele.2015@live.rhul.ac.uk, cdobraunig@cs.ru.nl, guojian@ntu.edu.sg,
eranlambooij@gmail.com, Gregor.Leander@rub.de, todo.yosuke@lab.ntt.co.jp

Abstract. The design and analysis of dedicated tweakable block ciphers is a quite recent and very active research field that provides an ongoing stream of new insights. For instance, results of Kranz, Leander, and Wiemer from FSE 2017 show that the addition of a tweak using a linear tweak schedule does not introduce new linear characteristics. In this paper, we consider — to the best of our knowledge — for the first time the effect of the tweak on zero-correlation linear cryptanalysis for ciphers that have a linear tweak schedule. It turns out that the tweak can often be used to get zero-correlation linear hulls covering more rounds compared to just searching zero-correlation linear hulls on the data-path of a cipher. Moreover, this also implies the existence of integral distinguishers on the same number of rounds. We have applied our technique on round reduced versions of QARMA, MANTIS, and SKINNY. As a result, we can present — to the best of our knowledge — the best attack (with respect to number of rounds) on a round-reduced variant of QARMA.

Keywords: Symmetric-key cryptography · tweakable block ciphers · zero-correlation · integral cryptanalysis · Qarma · Mantis · Skinny

1 Introduction

Tweakable block ciphers are constructions, which have — compared to traditional block ciphers — an additional input called *tweak*. Ideally, each different choice of the tweak produces a different instance of a block cipher. This concept has first been introduced by Schroeppel in the Hasty pudding cipher [Sch98] and was formally treated by Liskov, Rivest and Wagner [LRW02, LRW11]. The concept of tweakable block ciphers allows for very clean modes of operations for authenticated encryption like: ΘCB3 [KR11], or Counter-in-Tweak [PS16]. When using such a mode, one faces two choices, either use a construction that takes an ordinary block cipher as building block to build a tweakable block cipher [KR11, LST12, Men15, WGZ⁺16], or use a dedicated tweakable block cipher [JNP14, BJK⁺16, Ava17].

One can expect that designing a tweakable block cipher from scratch results in more efficient designs rather than reusing a block cipher to create a tweakable block cipher. However, when designing dedicated tweakable block ciphers, it has to be kept in mind that

the tweak is an additional publicly known input, which can potentially be influenced by an attacker. This leads to a new challenge in the analysis of such schemes, since in a chosen plaintext/related tweak model, the extra input provides additional freedom for the attacker. This freedom can be exploited in attacks. The most self-evident attack vector that is influenced by the tweak is differential cryptanalysis [BS91]. By introducing differences in the tweak, the attacker is able to introduce differences in-between rounds, which typically leads to longer differential characteristics that hold with a good probability. Naturally, this increases the number of rounds that can be covered in a key-recovery attack. Besides this, there is a constant evaluation of known attack vectors on tweakable block ciphers that exploit the tweak. There are for example: Boomerang attacks [CHP⁺17, DL17], meet-in-the-middle attacks [TAY16], impossible differential attacks [DL17, Sas18] and integral attacks [DEM16]. A positive result with respect to the security of tweakable block ciphers is that the addition of a tweak, using a linear tweak schedule, does not require additional considerations with respect to linear cryptanalysis [KLW17].

Research Gap and Contribution. Attacks on dedicated tweakable block ciphers exploit the additional freedom introduced by the tweak to extend a distinguisher in the data-path of a cipher. In this work, we follow this general idea to derive distinguishers not only on the data-path but also by considering the tweak schedule, which can be used to improve the attacks. In particular, we exploit zero-correlation linear hulls [BW12, BR14] on the data-path plus tweak. The fact that a lot of state-of-the-art tweakable block cipher constructions not only use a tweak schedule that is linear, but also have very limited diffusion in the tweak bits, becomes an advantage for an attacker. This allows us to search for zero-correlation linear hulls with the help of the miss-in-the-middle approach. In our attacks the miss (contradiction) occurs within the tweak schedule.

These zero-correlation linear hulls typically cover more rounds than zero-correlation linear hulls that only consider the data-path. Next to that, the relation between zero-correlation and integral distinguishers [SLR⁺15, BLN12] allows us to observe an integral property in the data-path. This property can then be exploited in key-recovery attacks.

In this paper, we first examine the effects of zero-correlation linear cryptanalysis on tweakable block ciphers having a linear tweak schedule. We focus on the implications on tweakable block ciphers following the *Superposition TWEAKY* (STK) constructions [JNP14]. After that, we give examples for zero-correlation linear hulls for three dedicated tweakable block ciphers QARMA [Ava17], MANTIS [BJK⁺16] and SKINNY [BJK⁺16]. As shown in Table 1, the newly acquired distinguishers allow for attacks covering more rounds compared to previous attacks in the case of round-reduced QARMA [Ava17]. QARMA is the tweakable block cipher used for pointer authentication in some ARM processors [Qua17].

Note that some of the attacks shown in Table 1 require more than 2^n data for an n -bit block size. In contrast to standard block ciphers where 2^n is the natural limit per key (i.e. the full-codebook is reached), tweakable block ciphers allow to gather the amount of 2^n data per tweak and hence, a total of 2^{n+t} data can be collected considering a t -bit tweak. Our attacks on SKINNY require data above 2^n , but we do not collect the full-codebook under one fixed tweakey. Hence, we can recover unknown tweakey-information that has not been queried in our key-recovery attacks.

Apart from the dedicated attacks, this new way of searching for integral distinguishers provides further insights in the design of tweakable block ciphers. One of the new insights is a better intuition on how the number of positions and the locations of the tweak addition influences the security of a tweakable block cipher. For instance, consider the case of a cipher where the addition of the tweak is just performed for a few rounds at the beginning and the end of the cipher, while for the rounds in the middle just the round-keys are added. Such a construction can lead to the unfortunate situation, that the zero-correlation linear hulls are independent of the number of keyed middle-rounds.

Table 1: Overview on previous and proposed key-recovery attacks on variants of QARMA-64, MANTIS, SKINNY-64/128, and SKINNY-64/192. MITM/ID/ZC/Inv. = Meet-in-the-Middle/Impossible Differentials/Zero-Correlation/Invariants

Cipher	Rounds	Attack type	Time	Data	Memory	Ref.
QARMA-64	4/4*	MITM	2^{90}	2^{16}	2^{90}	[LJ18]
QARMA-64	4/5*	MITM	2^{89}	2^{16}	2^{89}	[LJ18]
QARMA-64	4/6*	MITM	$2^{70.1}$	2^{53}	2^{116}	[ZD16]
QARMA-64	3/8*	ID	$2^{64.4}$	2^{61}	-	[ZDW18]
QARMA-64	4/7*	ID	$2^{120.4}$	2^{61}	2^{116}	[YQC18]
QARMA-64	4/8*	ZC/Integral	$2^{66.2}$	$2^{48.4}$	$2^{53.70}$	This Work
MANTIS	5/5*	Inv.	2^{56}	$2^{9.3}$	-	[Bey18]
MANTIS	6/6*	Diff.	2^{38}	2^{28}	-	[DEKM16]
MANTIS	7/7*	Diff.	$2^{53.94}$	$2^{53.94}$	-	[EK17]
MANTIS	4/8*	ZC/Integral	$2^{66.2}$	$2^{48.4}$	$2^{53.70}$	This Work
SKINNY-64/128	18	ZC	2^{126}	$2^{62.68}$	2^{64}	[SMB18]
SKINNY-64/128	19	ID	$2^{119.8}$	2^{62}	2^{110}	[YQC17]
SKINNY-64/128	20	ID	$2^{121.08}$	$2^{47.69}$	$2^{47.69}$	[TAY17]
SKINNY-64/128	20	ZC/Integral	$2^{97.5}$	$2^{68.4\dagger}$	2^{82}	This Work
SKINNY-64/128	23	ID	2^{124}	$2^{62.47}$	$2^{77.47}$	[SMB18]
SKINNY-64/128	23	ID	$2^{125.9}$	$2^{62.5}$	$2^{124.0}$	[LGS17]
SKINNY-64/128	23	ID	2^{79}	$2^{71.4\dagger}$	$2^{64.0}$	[ABC ⁺ 17]
SKINNY-64/192	21	ID	$2^{180.5}$	2^{62}	2^{170}	[YQC17]
SKINNY-64/192	22	ID	$2^{183.97}$	$2^{47.84}$	$2^{74.84}$	[TAY17]
SKINNY-64/192	23	ZC/Integral	$2^{155.6}$	$2^{73.2\dagger}$	2^{138}	This Work
SKINNY-64/192	27	Rectangle	$2^{165.5}$	$2^{63.5}$	2^{80}	[LGS17]

Related Work. The conversion [SLR⁺15] of zero-correlation linear hulls to what is commonly referred to as integral distinguishers is not the only method to find such distinguishers. Another common approach to find integral distinguishers is to exploit knowledge about upper bounds of the algebraic degree of a function as shown in higher-order differential cryptanalysis [Lai94]. Later on, methods that exploit the structure of a cipher in a more direct manner have been introduced in an attack on the block cipher Square [DKR97] which became known under the name integral cryptanalysis [KW02]. Moreover, the division property [Tod15b] and bit-based division property [TM16] provide a powerful improvement in the search for integral distinguishers that for example lead to attacks on full MISTY-1 [Tod15a, Tod17].

It is worth mentioning that **Table 1** just shows key-recovery attacks and thus, does not represent a complete list of results that provide insight into the security of QARMA, MANTIS and SKINNY. For instance Leander, Tezcan, and Wiemer [LTW18] provide results regarding the length of subspace trails for various ciphers including QARMA and SKINNY.

*We state the number of S-box layers in the inbound/outbound phase of the cipher.

[†]The attack requires more than 2^n data, where n is the block size. The full-codebook in a tweakable block cipher is exceeded by using more than 2^{n+t} data, considering a t -bit tweak.

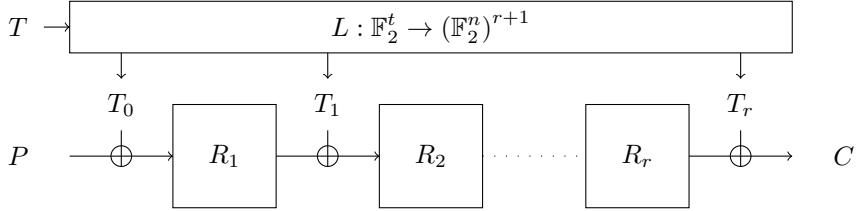


Figure 1: Key-alternating tweakable block cipher with linear tweak schedule.

Furthermore, Cid et al. [CHP⁺18] use their new tool called *Boomerang Connectivity Table* to re-evaluate existing related-tweakkey boomerang characteristics of SKINNY. Further works give more insight into the security of SKINNY against differential cryptanalysis [AK19] and impossible differential cryptanalysis [ST17] and the security of SKINNY and MANTIS against invariant attacks [BCLR17]. Eskandari et al. [EKKT19] search for integral distinguishers based on the division property for QARMA-64, MANTIS, and SKINNY-64. Furthermore, Zhang and Rijmen [ZR17] give integral distinguishers for 10 rounds of SKINNY-64 based on the division property.

The property of linear hulls under the related-key setting was also discussed by Bogdanov et al. in [BBR⁺13]. They showed that there exist linear hulls such that their bias are invariant under key difference. More concretely, when some bits in the secret-key must be inactive of a given linear hull, then there exists another linear hull with the same correlation, where the key difference is induced into the inactive bits. In comparison to our work, we review this property from zero-correlation linear hulls. By considering zero-correlation linear hulls, we can construct non-trivial distinguishers even if all bits in the secret-key/tweak are active. Therefore, our attacks are less restricted and improve over the results of Bogdanov et al. [BBR⁺13].

Outline. The paper is organized as follows. After briefly revisiting the necessary preliminaries on tweakable block ciphers, linear and zero-correlation cryptanalysis in Section 2, we explain the generic zero-correlation attack on tweakable block cipher in full detail in Section 3. Moreover, we apply the attack to QARMA, MANTIS and SKINNY in Section 4, 5 and Section 6, respectively. Finally, Section 7 concludes this work.

2 Preliminaries

2.1 Tweakable Block Cipher and TWEAKY Framework

Tweakable block ciphers were initially introduced by the Hasty pudding cipher [Sch98], and then, they were formally defined by Liskov, Rivest and Wagner [LRW11]. When the block and key lengths are n and κ bits, respectively, a conventional block cipher ($C = E_k(P)$) is defined as a function from $\mathbb{F}_2^n \times \mathbb{F}_2^\kappa \rightarrow \mathbb{F}_2^n$. A tweakable block cipher ($C = E_k(P, T) = E_k^T(P)$) has an additional input called *tweak* and it is defined as a function from $\mathbb{F}_2^n \times \mathbb{F}_2^\kappa \times \mathbb{F}_2^t \rightarrow \mathbb{F}_2^n$ when the tweak length is t bits. Responding to the high demand, many dedicated tweakable block ciphers have been proposed [JNP14, BJK⁺16, Ava17]. Throughout the paper, we consider the case of a tweakable round based block cipher with a linear tweak-scheduling $L : \mathbb{F}_2^t \rightarrow (\mathbb{F}_2^n)^{r+1}$ mapping the (master)-tweak to the sub-tweaks, as outlined in Fig. 1. Those sub-tweaks are then XORed to the current state of the cipher. The TWEAKY framework [JNP14], as illustrated in Fig. 2, is often used to design dedicated tweakable block ciphers, where the key and tweak are basically treated as one object called *tweakkey*. Moreover, each sub-tweakkey is generated by applying the same permutation

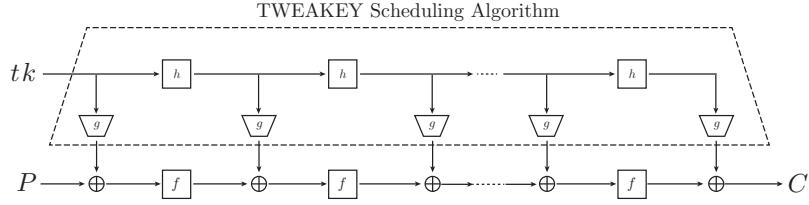


Figure 2: The TWEAKY framework.

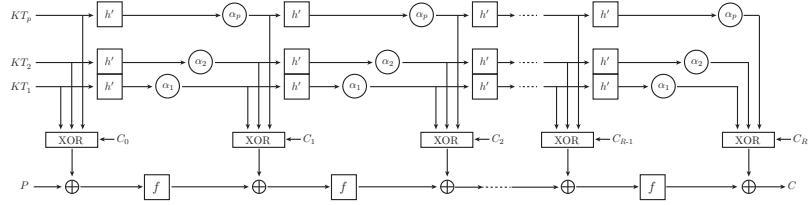


Figure 3: STK construction: Example with TK- p .

recursively. Based on this framework, there are several dedicated tweakable block ciphers such as KIASU-BC [JNP15c], DEOXYX [JNP15a], JOLTIK [JNP15b] and SKINNY [BJK⁺16]. Figure 2 shows the TWEAKY framework, where the tweakey scheduling algorithm is used instead of the key scheduling algorithm of the block cipher. The TWEAKY framework consists of a sub-tweakey extraction function g , internal update permutation f , and tweakey state update function h . A ciphertext is computed from a plaintext by applying the permutation f iteratively, and the sub-tweakey is XORed with the internal state every round. A class of tweakable block cipher denoted by $\text{TK-}p$ is introduced when the size of the tweakey is $(p \times n)$ bits. Then, $\text{TK-}1$ is suited to the simple single-key block cipher with n -bit key, and $\text{TK-}2$ is suited to the tweakable block cipher with n -bit key and n -bit tweak.

Jean et al. [JNP14] gave practical subclass of the TWEAKY framework named Superposition TWEAKY (STK), and Fig. 3 shows the construction with $\text{TK-}p$. In the STK construction, the internal state and tweakey state are partitioned into n/c and pn/c c -bit nibbles, respectively. The function h is decomposed into two functions h' and α_j , where h' is a nibble position substitution function and a non-zero coefficient α_j is multiplied with each c -bit nibble over the finite field $GF(2^c)$. The function g is a simple XOR of p n -bit states, and an additional round constant C_i is XORed. We want to highlight that the tweakey scheduling algorithm of the STK construction is fully linear.

2.2 Evaluating the Security of Dedicated Tweakable Block Ciphers

The main goal in cryptanalysis is to provide as much insight as possible into the security of symmetric cryptographic primitives. Since full versions of proposed cryptographic primitives are usually computationally hard to attack, it is common to study and evaluate the security of cryptographic primitives by analysing round-reduced versions of those primitives. The difference between the highest number of rounds that can be attacked for a round-reduced variant and the proposed number of rounds specifies the security margin of the primitive.

Another important aspect in the analysis of a scheme is the freedom an attacker has. In the case of a block cipher ($E_k(P) = C$) this actually depends on the specific use of the block cipher, e.g., in which mode of operation it is used. Thus, an attacker might only

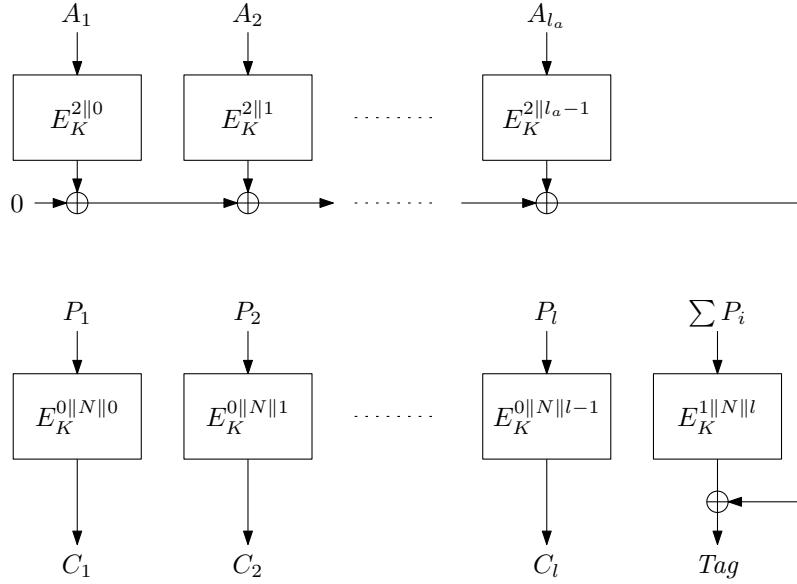


Figure 4: Sketch of the tweakable-block-cipher-based authenticated encryption scheme Deoxys-I [JNP15a].

know the ciphertext C , can make queries with plaintexts and ciphertexts of the attacker's choice [MvV96], or is even able to choose key-relations (related-key attacks [Bih94]). While it is debatable if block ciphers have to withstand powerful models like related-key attacks, it is good to know which do and which do not. However, it is usually expected that a good block cipher withstands attacks where the key is secret, but the attacker can freely choose the ciphertexts and plaintexts.

In the case of a tweakable block cipher ($E_k(P, T) = C$), we have an additional input called the tweak T . If we take a look at the existing analysis of dedicated tweakable block ciphers, e.g., [ABC⁺17, DL17, Sas18, DEKM16, EK18], we see that in most cases, an attacker is not only allowed to choose plaintexts P and ciphertexts C , but also knows and can pick the tweak T . But does this make sense in practical applications? To evaluate this, let us have a look at the authenticated encryption scheme Deoxys-I [JNP15a] in Figure 4, a CAESAR [CAE14] candidate utilizing Deoxys-BC as the underlying block cipher.

The nonce-based authenticated encryption scheme Deoxys-I takes a public nonce N , associated data A and plaintext P as input and returns a ciphertext C together with the tag Tag ($\mathcal{E}_k(N, A, P) \rightarrow (C, \text{Tag})$) and the quadruple (N, A, C, Tag) is transmitted and visible to an attacker. As indicated in Figure 4, the tweak T used in the tweakable block cipher is the concatenation of a constant, the nonce N , and a block counter and thus, is at least known to an attacker. Furthermore, if we consider that CAESAR requires an authenticated encryption algorithm to be secure, independent of the choice of the nonce (except that the nonce just be used once), we can evaluate a worst-case scenario, where an attacker has control over the nonce N and the plaintext P including the length and hence, has also control over the tweak input. In [DEM16] attacks that utilize the resulting (somewhat) chosen-tweak scenario on reduced KIASU \neq are shown, which uses a similar mode as Deoxys-I.

While in other typical use-cases for tweakable block ciphers like memory encryption [Ava17] the control of the attacker over the tweak might be more limited, designers of dedicated tweakable block ciphers usually do not restrict their claims to limited control. For instance, the designers of the tweakable block ciphers MANTIS [BJK⁺16], and QARMA [Ava17] that we examine in this paper claim security under chosen tweaks. For instance in the

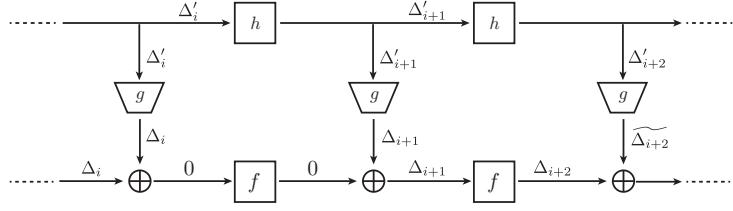


Figure 5: Differential model.

case of MANTIS: “*For MANTIS₇, we claim that any adversary who in possession of 2^n chosen plaintext/ciphertext pairs which were obtained under chosen tweaks, but with a fixed unknown key, needs at least 2^{126-n} calls to the encryption function in order to recover the secret key*” [BJK⁺16].

The attacks that we show on round-reduced versions of the tweakable block ciphers MANTIS and QARMA do not require the power of an attacker to choose the tweaks, instead they happen in a related-tweak scenario. Since our attacks make use of integral distinguishers, we require tweaks to be partially fixed to any value that does not have to be chosen by the attacker, while the other part iterates over all values for several plaintexts of the attacker’s choice. This is arguably a lighter scenario than choosing the tweaks and such a behaviour of the tweak might naturally happen in modes of operation that use a counter in the tweak to, e.g., encrypt more than one block or in memory encryption schemes that use the address as tweak input.

2.3 Differential Cryptanalysis

For all tweakable block ciphers discussed in this paper, an XOR is used to mix the sub-tweakey and internal state. This allows an attacker to cancel a difference of an internal state by XORing the same difference of a sub-tweakey to the same position. As a result, one round function is passed for free (i.e., see Fig. 5). In general, such a related-tweak setting allows for controlling differences of certain internal states. The probability of the obtained related-tweak/(twea)key differential characteristics is usually higher than that of single-key characteristics. Therefore, such attacks have been well discussed in the context of both related-key attacks on block ciphers and related-tweakey attacks on tweakable block ciphers.

2.4 Linear Cryptanalysis

Linear cryptanalysis makes use of correlations between linear combinations between input and output bits of a block cipher with a fixed key. More specifically, given a function

$$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m,$$

an input mask $\alpha \in \mathbb{F}_2^n$, and an output mask $\beta \in \mathbb{F}_2^m$ we consider

$$\text{cor}_F(\alpha, \beta) := 2 \cdot \text{Prob} (\langle \alpha, x \rangle + \langle \beta, F(x) \rangle = 0) - 1,$$

where the probability is taken over uniformly distributed inputs x . Traditionally, a high correlation is used as a distinguisher and then extended to a key-recovery attack [Mat94]. Moreover, we like to mention that for the understanding of our attacks, it might be helpful to have in mind two special cases for the propagation of masks, namely how linear masks propagate through an XOR-operation and a branching as illustrated in Fig. 6. In the formula, for the XOR operation

$$\begin{aligned} \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ X(x, y) &= x + y \end{aligned}$$

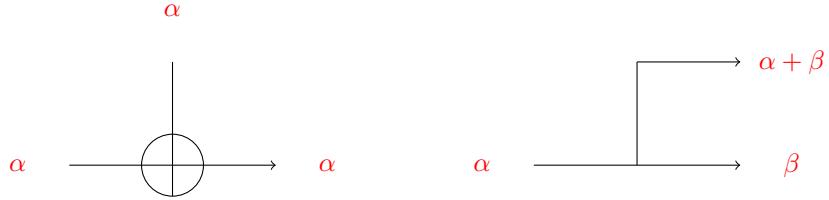


Figure 6: **Left:** Propagation of linear masks through **XOR**. **Right:** Propagation of linear masks through a branching point.

it holds that

$$\text{cor}_X((\alpha_1, \alpha_2), \beta) \neq 0 \text{ iff } \alpha_1 = \alpha_2 = \beta,$$

and for the branching operation

$$\begin{aligned} \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ B(x) &= (x, x) \end{aligned}$$

it holds that

$$\text{cor}_B(\alpha, (\beta_1, \beta_2)) \neq 0 \text{ iff } \alpha + \beta_1 + \beta_2 = 0.$$

Linear Hull

In the case of a round-based block cipher, the concept of linear hull ([Nyb95, Nyb01]) or correlations matrices ([Dae95]) are important tools to understand how the correlation is composed. Given a function F as the composition of r functions R_i , that is

$$F(x) = R_r(R_{r-1}(\dots R_1(x) \dots))$$

it is known that the correlation of F can be expressed as follows

$$\text{cor}_F(\alpha, \beta) = \sum_{\substack{\Gamma \in (\mathbb{F}_2^n)^{r-1} \\ \Gamma_0 = \alpha, \Gamma_r = \beta}} C_\Gamma$$

where C_Γ is defined as

$$C_\Gamma = \prod \text{cor}_{R_i}(\Gamma_{i-1}, \Gamma_i).$$

The value Γ , capturing all intermediated masks is what is referred to as the linear trail (or characteristics, path) and C_Γ is referred to as the trail correlation.

Zero-Correlation Linear Cryptanalysis

Zero-correlation linear cryptanalysis was introduced by Bogdanov and Rijmen [BR14]. Let α and β be the linear mask for a plaintext and ciphertext, respectively, zero-correlation attacks exploit the pair (α, β) with correlation exactly zero. One clear drawback of the basic zero-correlation linear cryptanalysis is its huge data complexity. In order to detect that the correlation is exactly zero, it is necessary to encrypt (almost) every possible message. Later, the data complexity was reduced by exploiting multiple or multidimensional zero-correlation linear approximations [BW12, BLNW12]. When there are ℓ zero-correlation linear approximations for an n -bit block cipher, the required data complexity is roughly estimated as $\mathcal{O}(2^n/\sqrt{\ell})$.

The main technique to derive zero-correlation linear approximations is very similar to deriving impossible differentials, that is a miss-in-the-middle approach. In a nutshell,

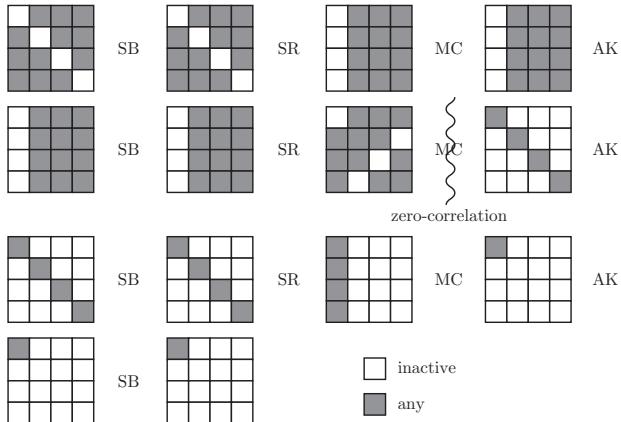


Figure 7: Zero-correlation linear hull on 4-round AES.

starting with a given input and output mask, one propagates the input mask forward and the output mask backwards through the encryption (resp. decryption) process. This propagation usually does not capture all linear trails with non-zero correlation in both direction exactly as this might easily get very difficult to handle, but rather captures an easy to describe super-set of all those trails. The fact that the linear approximation is then derived by deducing that those supersets of forward and backward linear trails have an empty intersection. As an illustration, we recall the well known zero-correlation linear hull on 4 rounds of the AES. Here, all bytes of the input mask are non-zero except for one diagonal, and the output linear mask is non-zero for only one byte. This then causes a contradiction in the second round `MixColumns` operation because of its branch number of 5.

Link From Zero-Correlation Linear to Integral

Several mathematical links among different types of cryptanalysis have been discussed, and here we focus on the link between zero-correlation linear cryptanalysis and integral cryptanalysis [BLNW12, SLR⁺15].

Theorem 1 (Link from zero-correlation linear hull to integral [SLR⁺15]). *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be a function, and A be a subspace of \mathbb{F}_2^n and $\beta \in \mathbb{F}_2^n \setminus \{0\}$. Suppose that (α, β) is a zero correlation linear approximation for any $\alpha \in A$, then for any $\lambda \in \mathbb{F}_2^n$, $\langle \beta, F(x + \lambda) \rangle$ is balanced on $A^\perp = \{x \in \mathbb{F}_2^n \mid \langle \alpha, x \rangle = 0, \alpha \in A\}$.*

Theorem 1 is proven in [SLR⁺15]. The Theorem shows that when there exists a zero-correlation linear hull, it implies an integral distinguisher. The required number of texts is 2^{n-m} , where m denotes the dimension of the subspace A . Recall the zero-correlation linear hull on 4-round AES (see Fig. 7). The zero-correlation linear hull can be converted into the integral distinguisher with 2^{32} texts, which is the exactly same as the well-known integral distinguisher of the 4-round AES [DKR97, KW02].

The known-plaintext assumption is used in the naive key-recovery of (multidimensional) zero-correlation linear cryptanalysis. If we assume a chosen-plaintext scenario, we can reduce the required data complexity by linking to the integral attack from zero-correlation linear cryptanalysis. In this work, when the key-recovery is taken into consideration, we convert the zero-correlation linear hull into integral distinguisher.

Key Recovery Technique for Integral Attacks

When N texts are required in the integral distinguisher and κ -bits of the secret-key are involved to evaluate balanced bits, the trivial key-recovery requires a time complexity of $N \times 2^\kappa$. There are two improved techniques to reduce the time complexity, i.e., the first one is the partial-sum technique [FKL⁺01] and the other is the FFT key recovery technique [TA14]. In the partial-sum technique, we first store the frequency of ciphertexts into a memory. Then, the ciphertexts are partially decrypted by guessing the part of involved keys, and the size of the memory is reduced. Since the complexity is the product of the memory size and the partially guessed key size, the attacker can reduce the whole complexity by partial decryption and compressing the data size step-by-step. The FFT key-recovery technique has a simpler description than the partial-sum technique, and thus, we can estimate the time complexity only by enumerating the involved key bits and ciphertext bits. Assuming that we need to evaluate $\bigoplus f_{k_1}(c \oplus k_2)$, where k_1 is the κ -bit round-key and c and k_2 are ℓ -bit (partial) ciphertext and the last round-key, respectively, the time complexity is estimated as $3\ell \times 2^{\kappa+\ell}$. Unfortunately, the FFT key-recovery technique cannot reduce the time complexity if some parts of the round-key are not mixed with the state. For example, in the AddRoundTweakey function of SKINNY just the two topmost rows of the tweakey are XORed with the full state. In such a case, the partial-sum technique is more efficient than the FFT key-recovery technique.

3 Zero-Correlation Linear on Tweakable Block Ciphers

In the case of a tweakable block cipher

$$E_k : \mathbb{F}_2^n \times \mathbb{F}_2^t \rightarrow \mathbb{F}_2^n,$$

we consider the tweak to be an additional input from which we can include the tweak bits into the linear combination of input bits, when considering linear approximations. More precisely, the input mask α now consists of two parts, $\alpha_1 \in \mathbb{F}_2^n$ and $\alpha_2 \in \mathbb{F}_2^t$ and we have to consider

$$\text{cor}_{E_k}((\alpha_1, \alpha_2), \beta) := 2 \cdot \text{Prob}(\langle \alpha_1, P \rangle + \langle \alpha_2, T \rangle + \langle \beta, E_k(P, T) \rangle = 0) - 1$$

where now the probability is taken over uniformly distributed inputs P and T .

Let $L : \mathbb{F}_2^t \rightarrow (\mathbb{F}_2^n)^{r+1}$ be a linear tweak-schedule, as was shown in [KLW17]. The corresponding linear hull for this setting becomes

$$\text{cor}_F((\alpha_1, \alpha_2), \beta) = \sum_{\substack{\Gamma \in (\mathbb{F}_2^n)^{r-1}, \Gamma_0 = \alpha_1, \Gamma_r = \beta \\ L^T(\Gamma) = \alpha_2}} C_\Gamma \quad (1)$$

where L^T is the adjoint linear layer of L , i.e., the unique linear mapping such that

$$\langle x, L(y) \rangle = \langle L^T(x), y \rangle$$

for all x, y . If we represent L as a matrix multiplication, then L^T is the transposed matrix. This was used in [KLW17] to argue that, in contrast to differential cryptanalysis, no new linear trails are introduced by the tweak. Thus, in order to protect against linear cryptanalysis, no fundamental new tools have to be developed. However, given the additional restriction on linear trails in the hull for tweakable ciphers, the formula actually already hints that zero-correlation might be more effective in this case.

As a first example consider the simple case of a two round tweakable cipher, where the tweak is just XORed to the state as illustrated in Fig. 8.

$$\begin{aligned} E_k : \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ E_k(x, t) &= R_2(R_1(x + t) + t) \end{aligned}$$

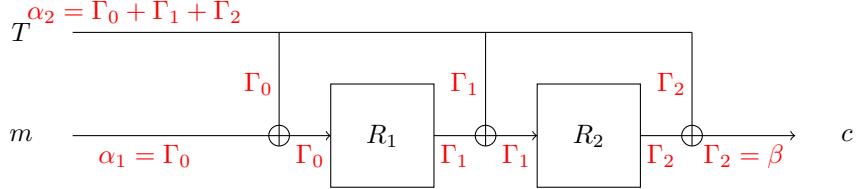


Figure 8: Propagation of masks in a simple two round tweakable block cipher.

Here, the tweak-scheduling is clearly linear and the mapping is simply

$$\begin{aligned} L : \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \\ L(t) &= (t, t, t). \end{aligned}$$

The adjoint linear layer, is the mapping

$$\begin{aligned} L^T : \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ L^T(t_1, t_2, t_3) &= t_1 + t_2 + t_3. \end{aligned}$$

Now, consider the linear hull for E_k with input mask (α_1, α_2) and output mask β . Note, that the input and output masks are independent [BBR⁺13]. Here α_1 is the mask for the data input and α_2 is the input mask for the tweak. According to Equation (1), the correlation of E_k becomes

$$\text{cor}_{E_k}((\alpha_1, \alpha_2), \beta) = \sum_{\substack{\Gamma \in (\mathbb{F}_2^n)^3, \Gamma_0 = \alpha, \Gamma_2 = \beta \\ L^T(\Gamma) = \beta}} C_\Gamma.$$

Now as

$$L^T(\Gamma) = L^T(\Gamma_0, \Gamma_1, \Gamma_2) = \Gamma_0 + \Gamma_1 + \Gamma_2$$

and $\Gamma_0 = \alpha$ as well as $\Gamma_2 = \beta$, we see that $\Gamma_1 = \alpha_1 + \alpha_2 + \beta$ and the linear hull reduces to a *single trail*, namely

$$\text{cor}_{E_k}((\alpha_1, \alpha_2), \beta) = \text{cor}_{R_1}(\alpha_1, \alpha_1 + \alpha_2 + \beta) \text{cor}_{R_2}(\alpha_1 + \alpha_2 + \beta, \beta) \quad (2)$$

Thus, for a given α_1 and β by choosing α_2 such that either $\text{cor}_{R_1}(\alpha_1, \alpha_1 + \alpha_2 + \beta)$ or $\text{cor}_{R_2}(\alpha_1 + \alpha_2 + \beta, \beta)$ equals zero, we derived a zero-correlation linear approximation. Thus, as long as there exist a zero-correlation linear approximation for R_1 (resp. R_2) the corresponding tweakable cipher has a zero-correlation linear approximation for *any choice of R_2* (resp. R_1). This is the basic observation we are going to use throughout the paper for our attacks.

In the general case, we are going to use forward and backward propagation to get a superset $\mathbb{S} \subset (\mathbb{F}_2^n)^{r+1}$ of all characteristics with non-zero correlation. Next, we check if $L(\mathbb{S}) \subset \mathbb{F}_2^\ell$ is not the full space. If so, we get a zero correlation by picking the mask for the tweak in $\mathbb{F}_2^\ell \setminus \mathbb{S}$. Note that, this becomes easier when the tweak-scheduling actually operates on single nibbles, as is the case for the tweakey setting STK as we will explain in Subsection 3.1.

From Zero-Correlation To Integral

In order to make the link between zero-correlation and integral cryptanalysis in the case of tweakable block ciphers more clear, we will demonstrate how to apply it to a simple

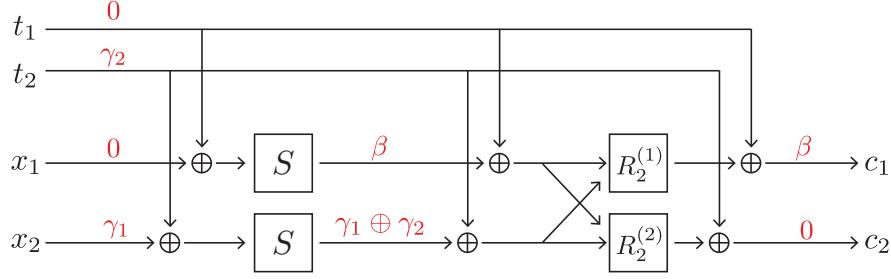


Figure 9: Propagation of masks in a simple two round tweakable block cipher with two S-boxes.

two-round tweakable block cipher as illustrated in the example in Fig. 8. For this, consider the case where R_1 consists of two parallel applications of a permutation $S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, i.e.

$$R_1(x, y) = (S(x), S(y)).$$

The entire function then becomes

$$\begin{aligned} E : (\mathbb{F}_2^m \times \mathbb{F}_2^m) \times (\mathbb{F}_2^m \times \mathbb{F}_2^m) &\rightarrow (\mathbb{F}_2^m \times \mathbb{F}_2^m) \\ E(x_1, x_2, t_1, t_2) &= (c_1, c_2) \end{aligned}$$

with x_1, x_2 (resp. (t_1, t_2)) being the two m -bit parts of the $2m$ bit message (resp. tweak). Splitting R_2 into its two components

$$\begin{aligned} R_2 : \mathbb{F}_2^m \times \mathbb{F}_2^m &\rightarrow \mathbb{F}_2^m \times \mathbb{F}_2^m \\ R_2(y_1, y_2) &= (R_2^{(1)}(y_1, y_2), R_2^{(2)}(y_1, y_2)) \end{aligned}$$

we get

$$\begin{aligned} c_1 &= R_2^{(1)}(S(x_1 + t_1) + t_1, S(x_2 + t_2) + t_2) + t_1 \\ c_2 &= R_2^{(2)}(S(x_1 + t_1) + t_1, S(x_2 + t_2) + t_2) + t_2 \end{aligned}$$

Figure 9 shows the propagation of the simple tweakable block cipher. We now fix any nonzero vector $\beta \in \mathbb{F}_2^m$ and consider the output mask $(\beta, 0)$. As input masks for the message we take $(0, \gamma_1)$ and for the tweak mask $(0, \gamma_2)$. In this case we get

$$\text{cor}_{R_1}((0, \gamma_1), (\beta, \gamma_1 \oplus \gamma_2)) = \text{cor}_S(0, \beta) \text{cor}_S(\gamma_1, \gamma_1 + \gamma_2)$$

which, as S is a permutation and β is non-zero, is zero for any choice of γ_1, γ_2 . Thus, Equation (2) implies

$$\text{cor}_E(((0, \gamma_1), (0, \gamma_2)), (\beta, 0)) = 0$$

for any choice of γ_1, γ_2 . Thus, the space of input masks with zero-correlation in this case is

$$A = \{((0, \gamma_1), (0, \gamma_2)) \mid \gamma_1, \gamma_2 \in \mathbb{F}_2^m\}$$

and its dual is

$$A^\perp = \{((x, 0), (y, 0)) \mid x, y \in \mathbb{F}_2^m\}.$$

According to Theorem 1, the function

$$(m, t) \mapsto \langle (\beta, 0), E_k(m, t) \rangle$$

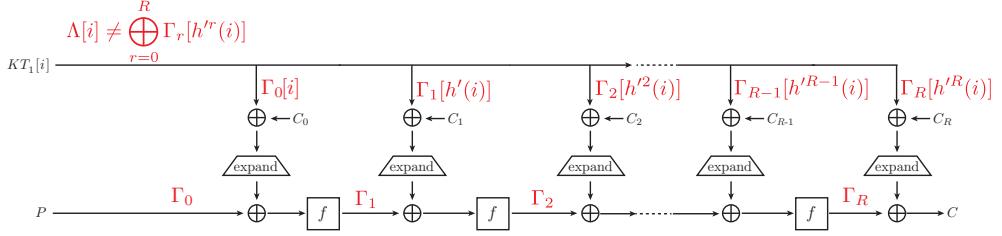


Figure 10: Zero-correlation linear hull on the STK with TK-1.

is balanced on each coset of A^\perp , that is for each $\lambda, \mu \in \mathbb{F}_2^m$ on each set

$$A_{\lambda, \mu} := \{((x, \lambda), (y, \mu)) \mid x, y \in \mathbb{F}_2^m\}$$

Thus, fixing the second half of both the message and the tweak, results in a function that is balanced (i.e. 0 and 1 appear equally often). For completeness, we note that this can also be deduced directly as follows.

$$\begin{aligned} T_{\beta, \lambda, \mu} &= \sum_{(x, y) \in A_{\lambda, \mu}} (-1)^{\langle (\beta, 0), E_k(x, y) \rangle} \\ &= \sum_{x_1, t_1 \in \mathbb{F}_2^m} (-1)^{\langle \beta, R_2^{(1)}(S(x_1 + t_1) + t_1, S(\lambda + \mu) + \lambda) + t_1 \rangle} \\ &= \sum_{x', t_1 \in \mathbb{F}_2^m} (-1)^{\langle \beta, R_2^{(1)}(S(x') + t_1, S(\lambda + \mu) + \lambda) + t_1 \rangle} \\ &= \sum_{x'', t_1 \in \mathbb{F}_2^m} (-1)^{\langle \beta, R_2^{(1)}(x'' + t_1, S(\lambda + \mu) + \lambda) + t_1 \rangle} \\ &= \sum_{x''', t_1 \in \mathbb{F}_2^m} (-1)^{\langle \beta, R_2^{(1)}(x''', S(\lambda + \mu) + \lambda) + t_1 \rangle} \\ &= \sum_{x''' \in \mathbb{F}_2^m} (-1)^{\langle \beta, R_2^{(1)}(x''', S(\lambda + \mu) + \lambda) \rangle} \left(\sum_{t_1 \in \mathbb{F}_2^m} (-1)^{\langle \beta, t_1 \rangle} \right) \\ &= 0. \end{aligned}$$

We subsequently replaced the variables and finally used that β is non-zero.

3.1 Zero-Correlation Linear Hull on STK with TK-1

When we consider the zero-correlation linear hull on general tweakable block ciphers, the domain space is expanded to $n + t$. This implies that we need to collect a huge amount of data, even if we can find a non-trivial zero-correlation linear hull. However, many dedicated tweakable block ciphers are designed based on the STK construction of the TWEAKY framework. In that case, the domain expansion of the zero-correlation linear hull is limited to a smaller size, and the number of chosen plaintexts and tweaks that we need to collect can be reduced.

Figure 10 shows the zero-correlation linear hull on the STK construction with TK-1. The tweakkey schedule of the STK construction with TK-1 consists of two functions, h and g as shown in Figure 2. The g function (denoted *expand* in Figure 10), is a subtweakkey extraction function that extracts the individual round keys from the tweakkey state and incorporates it to the internal state. The h' function is the tweakkey update function, where the nibble positions are simply permuted. Therefore, different nibbles are never mixed in

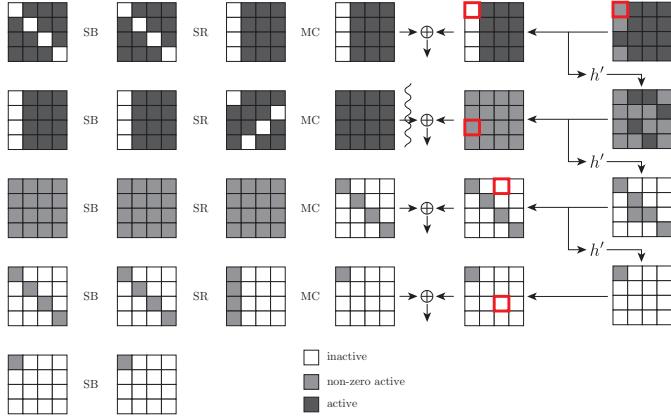


Figure 11: Zero-correlation linear hull on the toy cipher.

the tweakey scheduling algorithm, and we can focus on the i th c -bit nibble in KT_1 . Then, given a pair of input and output linear masks (Γ_0, Γ_R) , we enumerate all possible linear characteristics $(\Gamma_0, \Gamma_1, \dots, \Gamma_R)$ and evaluate a set \mathbb{S} such that

$$\mathbb{S} = \left\{ \Lambda[i] = \bigoplus_{r=0}^R \Gamma_r[h'^r(i)] \mid \forall (\Gamma_0[i], \Gamma_1[h'(i)], \dots, \Gamma_R[h'^R(i)]) \right\},$$

where $\Gamma_j[i]$ denotes the linear mask of the i th nibble in Γ_j , for $0 \leq j \leq R$. If the complement $\mathbb{F}_2^c \setminus \mathbb{S}$ is not empty, it causes a contradiction when $\Lambda[i] \in \mathbb{F}_2^c \setminus \mathbb{S}$. Note that the tweakey except for the i th c -bit nibble is independent of this linear hull, and it can be fixed to a (secret) constant. Furthermore, this implies that the domain expansion is only $n + c$ not $n + t$. Practically, we can use a miss-in-the-middle like algorithm to find such a linear hull.

Definition 1 (Γ sequence). The forward and backward propagations with probability one are evaluated from the given input linear mask Γ_0 and output linear mask Γ_r , respectively. Then, for any i , the Γ sequence is defined by the $(R+1)$ sequence, where whether $\Gamma_r[h'^r(i)]$ is active, inactive, or any is stored in the r th element.

When the Γ sequence is inactive for any i , it causes a contradiction when $\Lambda[i]$ is an active mask. Moreover, when there is one active value in the Γ sequence, it causes a contradiction when $\Lambda[i]$ is the zero mask. We use the following toy cipher to demonstrate the Γ sequence and show how to find a zero-correlation linear hull.

Example 1 (Toy Cipher). The round function is exactly the same as the AES round function. A simple tweakey scheduling algorithm is adopted instead of the AES key scheduling algorithm. The full tweak state is XORed when `AddRoundKey` is originally applied, and it uses $h' = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$, which is the same as the permutation P_T of SKINNY.

Figure 11 shows the 5-round zero-correlation linear hull, where we focus on the first byte in KT_1 . Then, the Γ sequence is $(0, 1, 0, 0)$, where 0 and 1 denotes inactive and active, respectively. Therefore, if a zero linear mask is applied to the first byte of KT_1 , it derives the zero-correlation linear hull. Moreover, we convert the zero-correlation linear hulls to the corresponding integral distinguisher, as shown in [SLR⁺15]. Zero linear masks are applied to $(32 + 8)$ bits, and any linear mask can be applied to the remaining 96 bits. Therefore, the required data complexity of the corresponding integral distinguisher is 2^{40} , and the discovered distinguisher can cover 5 rounds. We have practically verified such a distinguisher on a variant of the toy cipher with 4-bit nibbles using 2^{20} texts. More details

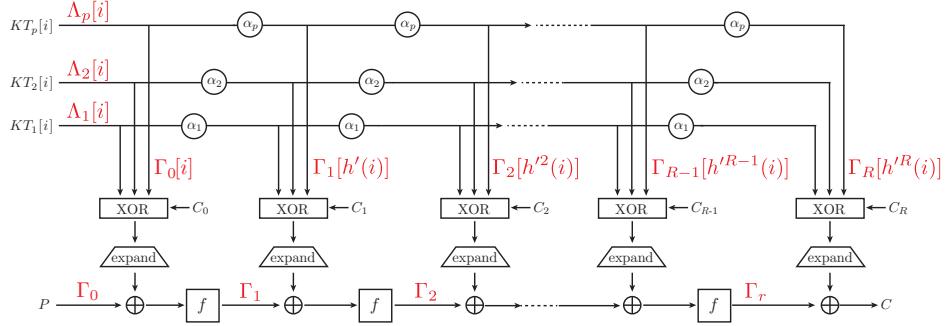


Figure 12: Zero-correlation linear hull on the STK with TK- p .

can be found in Appendix A. An interesting observation is that the second round function is independent of the zero-correlation linear hull. In other words, this distinguisher even holds if the second round function is replaced with any random permutation.

3.2 Zero-Correlation Linear Hull on TK- p

The STK construction with TK- p has p lines in the tweakey scheduling algorithm, and the same nibble position substitution function h' is applied to each line. However, a different coefficient α_j is multiplied with each c -bit nibble over $GF(2^c)$ in every line. Similarly to the case of the zero-correlation linear hull on the STK with TK-1, we can focus on the i th nibble in KT_1, KT_2, \dots, KT_p . Sub-tweakeys are generated by the XOR of p lines and all branches connected by XOR must have the same linear mask. Therefore,

$$\begin{pmatrix} \Lambda_1[i] \\ \Lambda_2[i] \\ \vdots \\ \Lambda_p[i] \end{pmatrix} = \begin{pmatrix} 1 & \alpha_1^T & (\alpha_1^T)^2 & \cdots & (\alpha_1^T)^R \\ 1 & \alpha_2^T & (\alpha_2^T)^2 & \cdots & (\alpha_2^T)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_p^T & (\alpha_p^T)^2 & \cdots & (\alpha_p^T)^R \end{pmatrix} \times \begin{pmatrix} \Gamma_0[i] \\ \Gamma_1[h'(i)] \\ \Gamma_2[h'^2(i)] \\ \vdots \\ \Gamma_R[h'^R(i)] \end{pmatrix}$$

where $\alpha_j^T : \mathbb{F}_2^c \rightarrow \mathbb{F}_2^c$ denotes the adjoint linear mapping of α_j , i.e., the mapping such that

$$\langle x, \alpha_j(y) \rangle = \langle \alpha_j^T(x), y \rangle, \quad \forall x, y \in \mathbb{F}_2^c.$$

We finally enumerate all possible linear characteristics $(\Gamma_0, \Gamma_1, \dots, \Gamma_R)$ from a given pair of input and output linear masks (Γ_0, Γ_R) . If the complement of the set of all possible $(\Lambda_1[i] \parallel \Lambda_2[i] \parallel \dots \parallel \Lambda_p[i])$ is not empty, there exists a zero-correlation linear hull. Practically, we can use the same method as in the case for TK-1. This is, choose Γ_0 and Γ_r , we evaluate the Γ sequence for any i . Then, we can show that the following proposition holds.

Proposition 1. *If there is a pair of linear masks (Γ_0, Γ_r) and the nibble position i such that the Γ sequence has at most p linearly active values, the tweakable block cipher has a non-trivial zero-correlation linear hull.*

Proof. We consider two cases: the Γ sequence is either inactive or active. The first case is trivial, and active linear mask $\Lambda_j[i]$ causes a contradiction. In the second case, we exploit the structure of the $p \times (R + 1)$ matrix. In the STK construction, α_j is chosen such that

the matrix

$$\begin{pmatrix} 1 & \alpha_1 & (\alpha_1)^2 & \cdots & (\alpha_1)^R \\ 1 & \alpha_2 & (\alpha_2)^2 & \cdots & (\alpha_2)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_p & (\alpha_p)^2 & \cdots & (\alpha_p)^R \end{pmatrix}$$

becomes MDS. Then, the matrix

$$\begin{pmatrix} 1 & \alpha_1^T & (\alpha_1^T)^2 & \cdots & (\alpha_1^T)^R \\ 1 & \alpha_2^T & (\alpha_2^T)^2 & \cdots & (\alpha_2^T)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_p^T & (\alpha_p^T)^2 & \cdots & (\alpha_p^T)^R \end{pmatrix}$$

also becomes MDS, as — using a suitable choice for the inner product — the adjoint linear mapping α_j^T is identical to α_j , and thus the matrix is unchanged. Therefore, in order to satisfy $\Lambda_j[i] = 0$ for all $j \in \{1, 2, \dots, p\}$, the Γ sequence must have at least $p + 1$ linearly active nibbles. In other words, if there are at most p linearly active nibbles in the Γ sequence, it causes a contradiction when all $\Lambda_j[i] = 0$. \square

Proposition 1 implies that the condition to find non-trivial zero-correlation linear hull is relaxed if there are more lines in the TWEAKEY construction (i.e., if p becomes larger in TK- p). More details can be found in Appendix A. For example, we can find a 5-round zero-correlation linear hull on a TK-1 construction, but we can further extend the number of rounds to a 6-round zero-correlation linear hull if a TK-2 construction is used.

4 Application to QARMA

We apply our technique to the QARMA family of lightweight tweakable block ciphers [Ava17]. QARMA has a block size of 64 or 128 bits, a key length of 128 or 256 bits, and a tweak length of 64 or 128 bits, respectively. We can successfully attack QARMA-64 whose numbers of forward and backward rounds are reduced to 4 and 8, respectively, under the related-tweak and chosen plaintext setting. More accurately, only 1 out of 16 cells of the tweak is active, while the other 15 cells take a known constant value. Our attack is currently the best known attack with respect to the number of total rounds.

4.1 Description of QARMA

An encryption of QARMA consists of forward round functions, a central construction, and backward round functions. In the specifications, the designer defines QARMA _{r} as QARMA whose numbers of forward and backward rounds are $r + 1$. In this paper however, for simplicity, we use a different notation denoted by QARMA _{r_1, r_2} , where the numbers of forward and backward rounds are r_1 and r_2 , respectively. Thus, QARMA _{r} corresponds to QARMA _{$r+1, r+1$} .

The state of QARMA is represented as a 4×4 matrix, where each index is defined as

$$\begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix}.$$

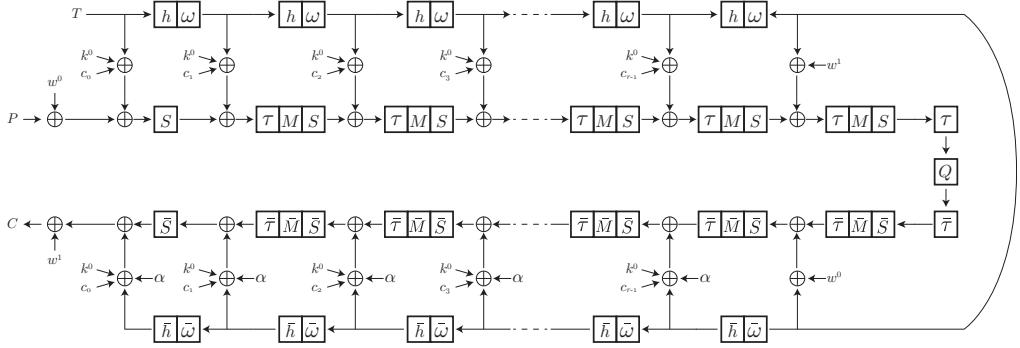


Figure 13: Structure of QARMA.

Every cell takes a 4 or 8-bit value in QARMA-64 and QARMA-128, respectively. In the state denoted by X , let $X[i_1, i_2, \dots, i_m]$ be $(s_{i_1}, s_{i_2}, \dots, s_{i_m})$ of X .

One round of QARMA consists of the following round operations (illustrated in Fig. 13):

- **SubCells (S):** substitutes each cell x by an involutory S-box. The following involutory 4-bit S-box σ_1 is directly applied for QARMA-64, and the 8-bit S-box in QARMA-128 is constructed by placing two σ_1 in parallel.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\sigma_1(x)$	a	d	e	6	f	7	3	5	9	8	0	c	b	1	2	4

- **AddRoundTweakey :** adds the (full) round-tweakey to the internal state.
- **ShuffleCells (τ):** applies the cell permutation of MIDORI as given below:

$$\tau = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2].$$

- **MixColumns (M):** multiplies each column of the state by the binary matrix from MIDORI M as shown below:

$$M = circ(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix} 0 & \rho^a & \rho^b & \rho^c \\ \rho^c & 0 & \rho^a & \rho^b \\ \rho^b & \rho^c & 0 & \rho^a \\ \rho^a & \rho^b & \rho^c & 0 \end{pmatrix},$$

For QARMA-64 and QARMA-128, $M = circ(0, \rho, \rho^2, \rho)$ and $M = circ(0, \rho, \rho^4, \rho^5)$ are used, respectively.

The tweak schedule consists of two functions, h and ω , where the h function is defined as simple permutation $h = [6, 5, 14, 15, 0, 1, 2, 3, 7, 12, 13, 4, 8, 9, 10, 11]$. Moreover, the ω function is a bit-based LFSR. The LFSR is however irrelevant for our attack, as we only consider cell-based linear masks that are either *inactive*, *active* or *any*. Moreover, in this paper we focus on QARMA-64.

4.2 Zero-Correlation Linear Hull on QARMA_{4,5}

Figure 14 shows two zero-correlation linear hulls on QARMA_{4,5}. Any linear masks are applied to 6 cells of the state, i.e., $(s_2, s_7, s_8, s_{12}, s_{13}, s_{15})$. Moreover, active linear masks

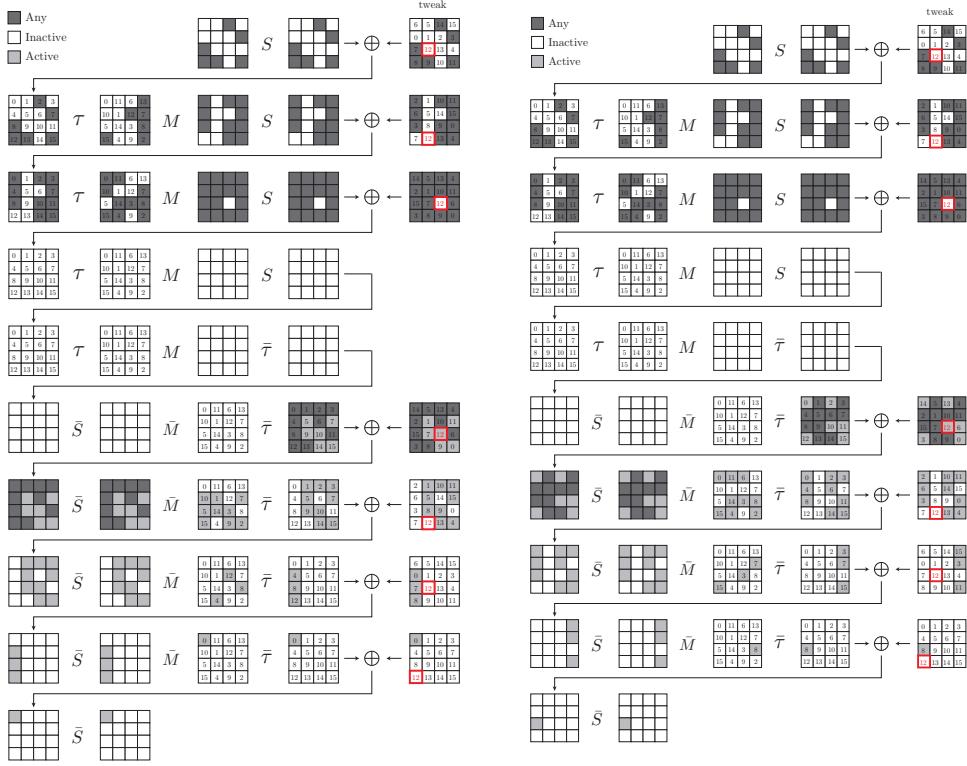


Figure 14: Two zero-correlation linear hulls on QARMA_{4,5}.

are applied to s_0 and s_8 of the output, as shown in Figure 14. Then, we focus on the tweak cell labelled 12. As illustrated in cells highlighted by red frames, the Γ sequence has just one active cell. Therefore, applying an inactive mask to the tweak cell labelled 12 causes a contradiction due to Proposition 1. Note that we do not need to activate any of the other 15 cells in the tweak and they can take any fixed value. Thus, the domain space of the zero-correlation linear hull becomes at most 17 (= 16 + 1) cells.

The attack assumption of the naive algorithm using zero-correlation linear hull is the known-plaintext and tweak setting, but it usually requires a huge data complexity. If we assume a chosen-plaintext and related-tweak setting, the required data complexity can be reduced by linking to integral distinguishers as described in Section 3. Any linear masks are applied to six cells in the two zero-correlation linear hulls, and inactive linear masks are applied to the other 11 (= 10 + 1) cells. Therefore, the corresponding related-tweak integral distinguisher requires $2^{10 \times 4} = 2^{40}$ chosen plaintexts over 2^4 related tweaks, and the total data complexity is $2^{40+4} = 2^{44}$. Here, the relation of the tweak is defined in such a way that the 4-bit cell labelled 12 takes all values. Both zero-correlation linear hulls outlined in Fig. 14 share the same input linear mask, and the output in position s_0 and s_8 is balanced at the same time[‡].

4.3 Key-Recovery Attacks on QARMA_{4,8}

In the key recovery, we first add pre-whitening before the integral distinguisher. Let P and T denote the states of plaintext and tweak, respectively. We first prepare a set of plaintexts and tweaks, where 10 cells at position $P[0, 1, 3, 4, 5, 6, 9, 10, 11, 14]$ and 1 cell at

[‡] s_{10} is also balanced at the same time.

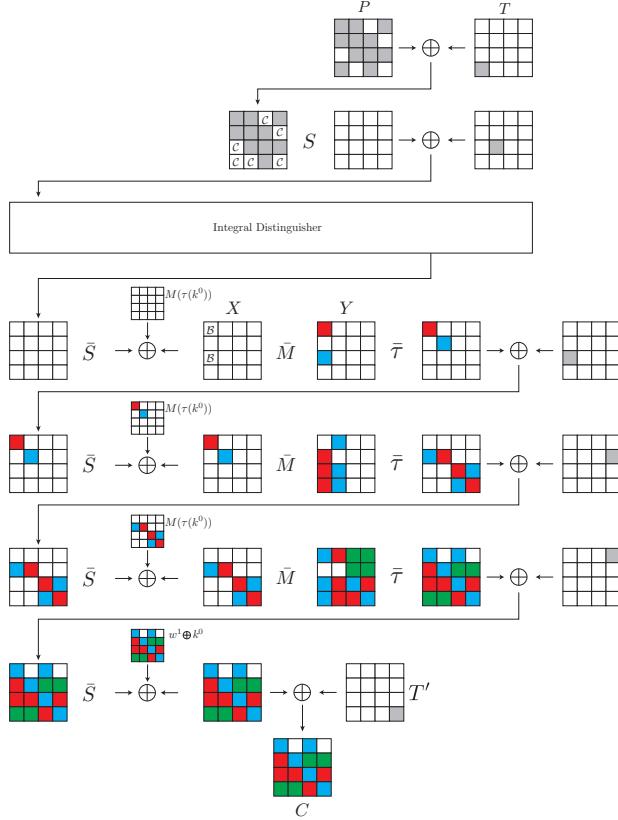


Figure 15: Key-recovery attacks on QARMA_{4,8}.

position $T[12]$ are active. Moreover, we choose the input such that $P[12] = T[12]$, and then, $(P \oplus T)[12]$ becomes constant, and further coincides to the input of the integral distinguisher.

We can append three rounds after the integral distinguisher. Figure 15 shows the key-recovery, and let X and Y be the states defined in Fig. 15. Due to the integral distinguisher, both s_0 and s_8 in X are balanced at the same time. Then

$$\begin{pmatrix} X_0 \\ X_4 \\ X_8 \\ X_{12} \end{pmatrix} = \begin{pmatrix} 0 & \rho & \rho^2 & \rho \\ \rho & 0 & \rho & \rho^2 \\ \rho^2 & \rho & 0 & \rho \\ \rho & \rho^2 & \rho & 0 \end{pmatrix} \begin{pmatrix} Y_0 \\ Y_4 \\ Y_8 \\ Y_{12} \end{pmatrix} = \begin{pmatrix} \rho Y_4 + \rho^2 Y_8 + \rho Y_{12} \\ \rho Y_0 + \rho Y_8 + \rho^2 Y_{12} \\ \rho^2 Y_0 + \rho Y_4 + \rho Y_{12} \\ \rho Y_0 + \rho^2 Y_4 + \rho Y_8 \end{pmatrix},$$

and

$$\begin{aligned} \bigoplus(X_0 + X_8) &= \bigoplus(\rho Y_4 + \rho^2 Y_8 + \rho Y_{12}) \oplus (\rho^2 Y_0 + \rho Y_4 + \rho Y_{12}) \\ &= \bigoplus(\rho^2 Y_0 + \rho^2 Y_8) = 0. \end{aligned}$$

Moreover ρ^2 is the rotation function, $\bigoplus(Y_0 + Y_8) = 0$.

We use the meet-in-the-middle technique for the integral attacks [SW13], where $\bigoplus Y_0$ and $\bigoplus Y_8$ are evaluated independently, and round keys satisfying $\bigoplus Y_0 = \bigoplus Y_8$ are recovered. The size of the involved secret-key is 56($= 14 \times 4$) bits in $w^1 \oplus k^0$ and 28($= (1+6) \times 4$) bits in $M(\tau(k^0))$, which gives a total of 84 key-bits that can be recovered. Since one structure

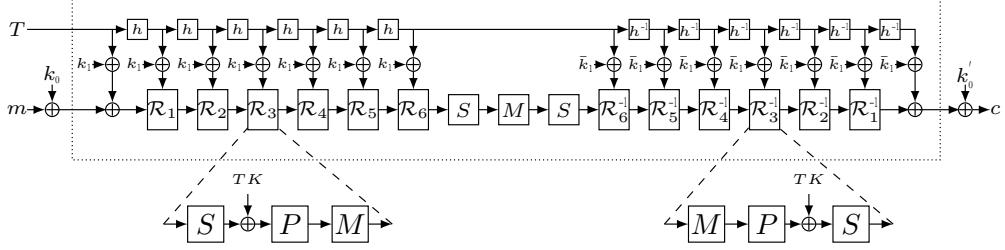


Figure 16: Illustration of the tweakable block cipher MANTIS.

removes incorrect secret-key bits by a factor of 2^{-4} , we need $84/4 = 21$ structures to uniquely determine the secret-key.

To compute $\bigoplus Y_0$, cells labelled red and green are involved, where 36-bit of ciphertexts, 36-bit of $w^1 \oplus k^0$, and 16-bit of $M(\tau(k^0))$ are involved. We first store the frequency of 36-bit ($C[4, 6, 7, 8, 9, 11, 12, 13, 14]$) into memory. Then, we use the FFT key-recovery technique [TA14], and the time complexity can be evaluated as

$$21 \times 4 \times (2^{16} \times (3 \times 36 \times 2^{36})) \approx 2^{65.2}.$$

As a result, we generate a list whose size is $2^{36+16} = 2^{52}$ and each value takes $((21 \times 4) + 16 + 4) = 104$ bits, where (21×4) -bit are the concatenation of $\bigoplus X_0$ in 21 structures, 16-bit are $(w^1 \oplus k^0)[6, 7, 12, 13]$, and 4-bit are $M(\tau(k^0))[5]$.

Similarly, cells labelled blue and green are involved to compute $\bigoplus X_8$, where 36-bit of ciphertexts, 36-bit of $w^1 \oplus k^0$, and 16-bit of $M(\tau(k^0))$ are involved. Then, we store the frequency of 36-bit ($C[0, 2, 5, 6, 7, 10, 12, 13] \parallel C[15] \oplus T'[15]$) into memory. Again, the FFT key-recovery technique enables us to compute the sum with $2^{65.2}$ computations, and we generate a similar list as in the case of $\bigoplus Y_0$. Finally, we compare these lists to find a match, and the time complexity is 2^{52} .

Thus, the total time complexity is $2 \times 2^{65.2} + 2^{52} \approx 2^{66.2}$, and the required data complexity is $21 \times 2^{44} \approx 2^{48.4}$. The memory complexity is determined by storing our two lists and is $2 \times 104/64 \times 2^{52} = 2^{53.70}$ 64-bit blocks. We already recover 56-bit $w^1 \oplus k^0$ and 28-bit $M(\tau(k^0))$, and there are still 44 bits of the secret-key, remaining. Finally, we exhaustively guess these bits, but the complexity, i.e., 2^{44} , is negligible compared with $2^{66.2}$. The security of QARMA-64 is claimed as $2^{128-d-2}$ where 2^d chosen or known {plaintext, ciphertext, tweak} triples, i.e., $2^{128-48.46-2} = 2^{77.54}$ in our case. Therefore, our attack against QARMA_{4,8} is valid.

5 Application to MANTIS

In this section, we apply the attack to a reduced-round version of MANTIS₈, where the number of forward and backward rounds are reduced to 4 and 8, respectively. Our attack assumption is the same as the case of QARMA, where only 1 cell in the tweak is activated and the other 15 cells can take any known constant.

5.1 Description of MANTIS

MANTIS is a family of lightweight tweakable block ciphers proposed together with SKINNY by Beierle *et al.* [BJK⁺16]. MANTIS has a block size of 64 bits, a key length of 128 bits, and a tweak length of 64 bits, respectively. The structure of MANTIS follows the design of PRINCE [BCG⁺12] and is aimed to achieve low-latency. While it is rather easy to turn the cipher into a tweakable cipher by using the TWEAKY framework, the designers reused

components of MIDORI [BBI⁺15] to achieve low-latency. One round of MANTIS consists of the following round operations (illustrated in Fig. 16):

- **SubCells** (SC): substitutes each nibble x by the involutory MIDORI S-box $Sb_0(x)$ which is given below:

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

- **AddConstant** (AC): adds a round constant RC_i to the state. The constants are similarly generated as in PRINCE.
- **AddRoundTweakey** (ART): adds the (full) round tweakey to the internal state.
- **PermuteCells** (PC): applies the cell permutation of MIDORI as given below:

$$P = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2].$$

- **MixColumns** (MC): multiplies each column of the state by the binary matrix from MIDORI M as shown below:

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

The state is represented as a 4×4 matrix, where each index is defined as

$$\begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix}.$$

In the state denoted by X , let $X[i_1, i_2, \dots, i_m]$ be $(s_{i_1}, s_{i_2}, \dots, s_{i_m})$ of X . The encryption of MANTIS consists of a forward round function, a central construction, and backward round function, similar as in QARMA. The designers of MANTIS defines MANTIS_r as MANTIS whose numbers of forward and backward rounds are r . For simplicity, we use a different notation denoted by MANTIS_{r₁,r₂}, where the numbers of forward and backward rounds are r_1 and r_2 , respectively.

5.2 Zero-Correlation Linear Hull on MANTIS_{4,5}

The zero-correlation linear hulls and the consequential integral distinguishers for MANTIS_{4,5} are identical to the distinguishers on QARMA_{4,5}. This is because, we can re-arrange the components of the round function in MANTIS so that the overall structure of MANTIS is the same as for QARMA. We can define MANTIS_r \sim QARMA_r by changing the applications of the round components from

$$\text{MixColumns} \circ \text{PermuteCells} \circ \text{AddTweakey}_{tk} \circ \text{AddConstanct}_i \circ \text{SubCells}$$

to

$$\text{SubCells} \circ \text{MixColumns} \circ \text{PermuteCells} \circ \text{AddTweakey}_{tk} \circ \text{AddConstanct}_i$$

Table 2: Comparison between MANTIS and QARMA.

	QARMA	MANTIS
Round function		
S-box	$\sigma_1 = [a, d, e, 6, f, 7, 3, 5, 9, 8, 0, c, b, 1, 2, 4]$ $\tau = [0, b, 6, d, a, 1, c, 7, 5, e, 3, 8, f, 4, 9, 2]$	$Sb_0 = [c, a, d, 3, e, b, f, 7, 8, 9, 1, 5, 0, 2, 4, 6]$ $P = [0, b, 6, d, a, 1, c, 7, 5, e, 3, 8, f, 4, 9, 2]$
Linear Layer	$M = \begin{pmatrix} 0 & \rho & \rho^2 & \rho \\ \rho & 0 & \rho & \rho^2 \\ \rho^2 & \rho & 0 & \rho \\ \rho & \rho^2 & \rho & 0 \end{pmatrix}$	$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

that is equivalent to the round structure of QARMA. Moreover, as the first and last round of QARMA are partial rounds (omitting `ShuffleCells` and `MixColumns`), this works for the beginning of the forward/backward rounds. Furthermore, as QARMA employs one forward and one backward round in the central construction, `ShuffleCells` and `MixColumns` can be added from MANTIS to complete the last round of the forward/backward rounds. The remaining S-box of QARMA is then equivalent to the application of the S-box in the middle construction of MANTIS.

Since our attack is of a general nature, and the components of MANTIS and QARMA are very similar (see the differences in Table 2), the distinguishers of QARMA can be re-used. All operations of both MANTIS and QARMA are in a nibble-by-nibble fashion, and the alignment of the state words is the same. Moreover, in the search for the zero-correlation linear hulls, we consider the m -bit S-box as an arbitrary S-box and do not consider the structure of a particular S-box. Similar the linear layer of MANTIS and QARMA just differ by some entries of the `MixColumns` matrix M , but again as we consider nibble-by-nibble operations and the matrices have the same structure (with an all zero-diagonal), so again there are no differences in the distinguisher. Finally, the additional application of an LFSR ω in the tweak-update function of QARMA also does not change the distinguisher, with a similar argument as for the differences in the `MixColumns` matrix M .

Figure 24 in Appendix C explicitly shows the zero-correlation linear hull for MANTIS_{4,5}, where cells s_0 and s_8 after `MixColumns` are linearly active, respectively.

5.3 Key-Recovery Attacks on MANTIS_{4,8}

Since our attacks are general against the TWEAKEY framework, and we can reuse the distinguishers of QARMA on MANTIS, we can further reuse the key-recovery for MANTIS. QARMA uses a 128-bit master key K that is initially partitioned as $w^0||k^0$, where w^i are the whitening keys and k^i are the core keys, respectively, for $i \in \{0, 1\}$. For encryption, $k^0 = k^1$ and $w^1 = (w^0 \ggg 1) \oplus (w^0 \gg (64 - 1))$.

MANTIS uses a 128-bit master key K that is split into $k_0||k_1$ that is then further extended to the 192-bit key

$$(k_0||k'_0||k_1) = (k_0||(k_0 \gg 1) \oplus (k_0 \gg 63)||k_1)$$

where k_0, k'_0 are the whitening keys and k_1 is the round key for all rounds in MANTIS.

In the key-recovery of QARMA we recover 56-bit of $w^1 \oplus k^0$ and 28-bit of $M(\tau(k^0))$. Since $w_{\text{QARMA}}^1 = k'_{0\text{MANTIS}}$ and $k_{\text{QARMA}}^0 = \bar{k}_{1\text{MANTIS}}$ we can recover the same key information as in QARMA (i.e., we can recover 56-bit of $k'_0 \oplus \bar{k}^1$ and 28-bit of $M(P(\bar{k}^1))$). Equally, as in the attack on QARMA_{4,8} the complexities to attack MANTIS_{4,8} are $2^{66.2}$ for time complexity, $2^{48.4}$ for data complexity, and $2^{53.64}$ 64-bit blocks for the memory complexity. Figure 25 in

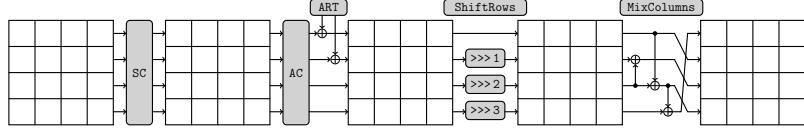


Figure 17: Round function of the tweakable block cipher SKINNY.

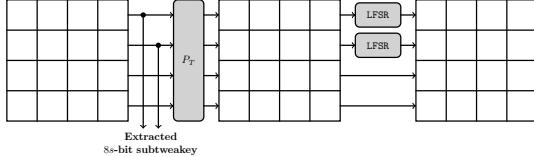


Figure 18: TWEAKEY schedule of SKINNY.

Appendix C explicitly shows the key-recovery for MANTIS_{4,8}.

6 Application to SKINNY

In this section, we apply the attack to reduced-round versions of SKINNY-64/128 and SKINNY-64/192. SKINNY [BJK⁺16] is designed according to the STK construction with TK- p , where $p \in \{1, 2, 3\}$. We show attacks on 20 rounds of SKINNY-64/128 and 23 rounds of SKINNY-64/192.

6.1 Description of SKINNY

SKINNY is a family of lightweight tweakable block ciphers introduced by Beierle *et al.* [BJK⁺16]. SKINNY has a block size n of 64 or 128 bits, and a tweakey size of $n/2n/3n$, where the tweakey can be both tweak and key. The aim of SKINNY is to achieve the performance of the NSA ciphers SIMON and SPECK [BSS⁺13], while still offering strong security bounds against differential/linear cryptanalysis. One round of SKINNY consists of the following round operations (illustrated in Fig. 17):

- **SubCells (SC):** substitutes each nibble x by the S-box $S(x)$ which is given below:

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

- **AddRoundConstants (AC):** adds LFSR-based round constants to cells 0, 4, and 8 of the state.
- **AddRoundTweakey (ART):** adds the round tweakey to the first two rows of the state.
- **ShiftRows (SR):** rotates the i^{th} row, for $i = 0 \leq i \leq 3$, by i positions to the right.
- **MixColumns (MC):** multiplies each column of the state by the binary matrix M :

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

The state is represented as a 4×4 matrix, where each index is defined as

$$\begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix}.$$

In the state denoted by X , let $X[i_1, i_2, \dots, i_m]$ be $(s_{i_1}, s_{i_2}, \dots, s_{i_m})$ of X . In this paper, we just consider SKINNY-64, however, the attack should easily be applicable to SKINNY-128, as we just consider cell-based operations and consider an arbitrary cell-size S-box. For two tweakey words (i.e., TK-2) the designers of SKINNY recommend 36 rounds, for three tweakey words (i.e., TK-3) the designers recommend 40 rounds.

6.2 Zero-Correlation Linear Hull on SKINNY-64/128

We searched the zero-correlation linear hull by using the miss-in-the-middle like algorithm. As a result, we found a 13-round zero-correlation linear hull for SKINNY-64/128. Here, active linear masks are applied to two cells (s_0, s_3) at the input, and active linear masks are applied to cells s_7 and s_{11} in the state before `MixColumns` at the output, as shown in Fig. 19. Then, we focus on the tweak cell labelled 9, where the Γ sequence is depicted by using a red frame. Since the Γ sequence has just two active cells and SKINNY-64/128 is based on TK-2, applying an inactive mask to the before mentioned tweak cell causes a contradiction due to Proposition 1. Note that the remaining 15×2 cells in the tweakey can take any constant, and the domain of our zero-correlation linear hull is $64 + 8 = 72$ bits. We can link the zero-correlation linear hull to a related-tweakey integral distinguisher. We apply any linear mask to the two cells (s_0, s_3) in the zero-correlation linear hulls as illustrated in Fig. 19 and apply inactive linear masks to the remaining 14 cells. Moreover, we apply inactive linear masks to the $2 \times 4 = 8$ -bit tweak cell labelled 9. Therefore, the corresponding related-tweakey integral distinguisher requires $2^{14 \times 4} = 2^{56}$ chosen plaintexts over 2^8 related tweakeys, and the total data complexity is $2^{56+8} = 2^{64}$. Here, the relation of the tweakey is defined in such a way that the $2 \times 4 = 8$ -bit cell labelled 9 takes all values. The integral distinguishers share the same input linear masks Γ_0 , and the output in cell s_{11} in the state after `MixColumns` is balanced.

6.3 Key-Recovery Attacks on SKINNY-64/128

Our attack model is a related-tweakey attack, where 2^8 related tweakeys are exploited. Then, there exist generic key-recovery attack with the time complexity of $2^{128-8} = 2^{120}$ [BMV11]. Therefore, the time complexity of a non-trivial key-recovery attack must be at most 2^{120} . In the key-recovery, we can prepend 1 round and append 6 rounds to the integral distinguisher. In total the attack reaches 20 rounds. Figure 20 shows the key-recovery, and let X_i , Y_i , and Z_i be the states defined in Fig. 20. Let P and T be the states of plaintext and tweak, respectively. We first prepare a set of chosen Z_1 , where 14 cells are active, i.e., $Z_1[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$. Moreover, we need the tweak cell $T[1]$ in the two lines to be active, as it will propagate to cell $T[9]$ after 1 round, which coincides with the beginning of both integral distinguishers as shown in Fig. 19. Note that the consistent set of chosen plaintexts and tweaks is computed from Z_1 and $T[1]$ without guessing any bits, since SKINNY does not have a whitening-key addition at the beginning.

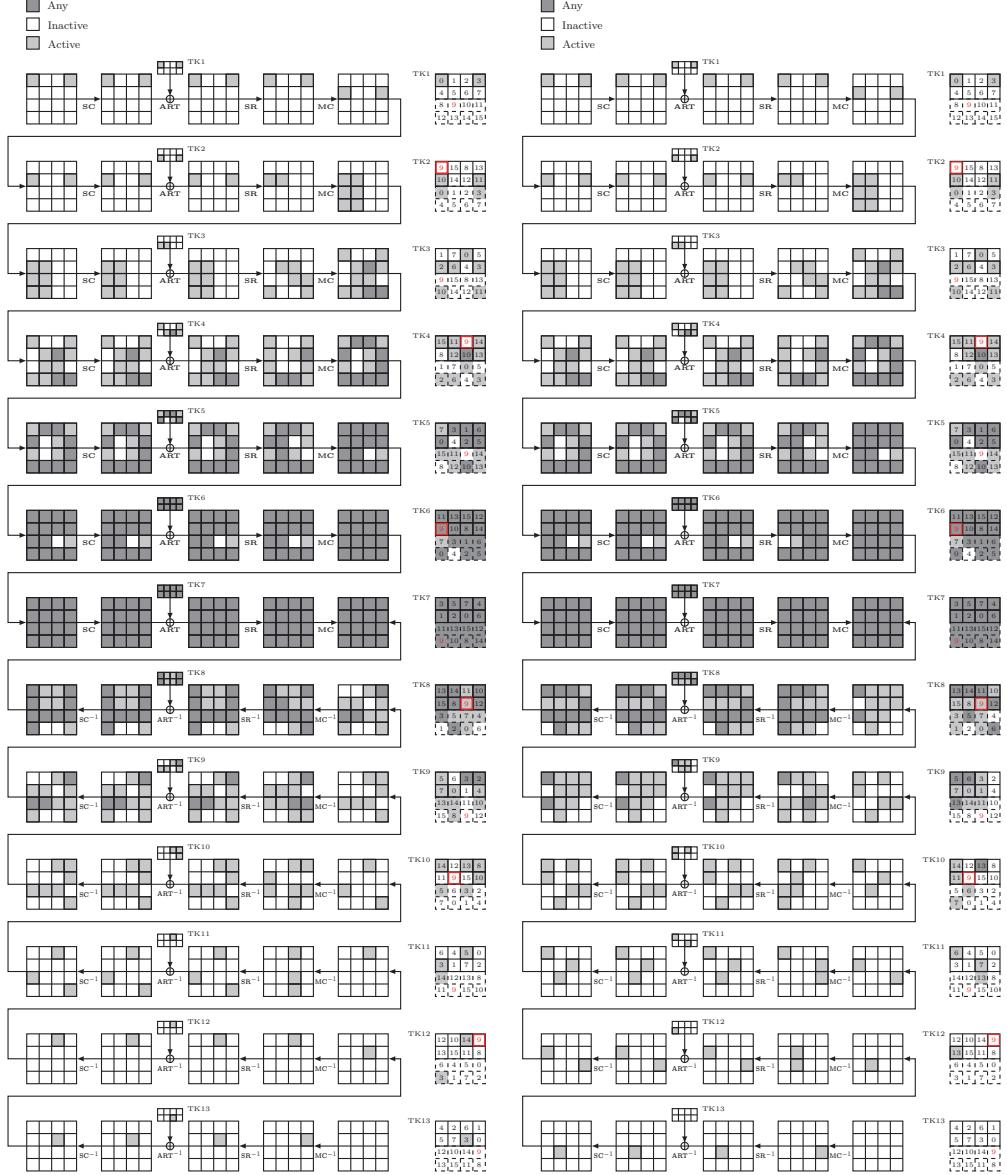


Figure 19: Two 13-round zero-correlation linear hulls for SKINNY-64/128.

Due to the integral distinguisher, both s_7 and s_{11} in Y_{14} are balanced. Then

$$\begin{pmatrix} Z_{14}[3] \\ Z_{14}[7] \\ Z_{14}[11] \\ Z_{14}[15] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} Y_{14}[3] \\ Y_{14}[7] \\ Y_{14}[11] \\ Y_{14}[15] \end{pmatrix} = \begin{pmatrix} Y_{14}[3] + Y_{14}[11] + Y_{14}[15] \\ Y_{14}[3] \\ Y_{14}[7] + Y_{14}[11] \\ Y_{14}[3] + Y_{14}[11] \end{pmatrix},$$

and

$$\bigoplus Z_{14}[11] = \bigoplus (Y_{14}[7] \oplus Y_{14}[11]) = 0$$

In SKINNY, the full tweakkey is not XORed with the internal state (i.e., just the top two rows of the tweakkey are XORed to the state), and then, the FFT key-recovery technique

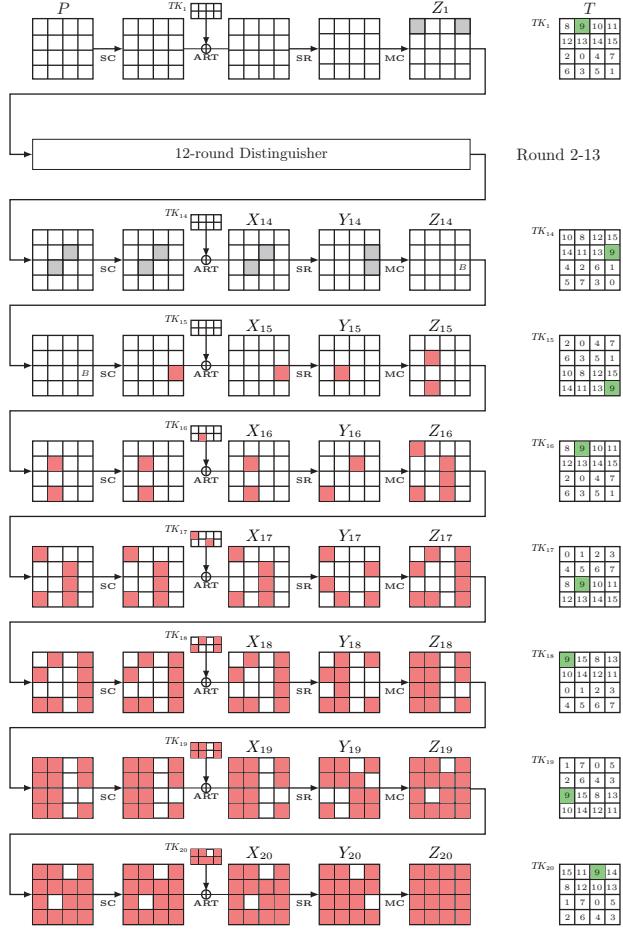


Figure 20: Key-recovery attack on 20 rounds of SKINNY-64/128.

is less efficient [TA14]. Therefore, we estimate the time complexity to recover round keys satisfying $\bigoplus Y_{11} = 0$ in detail by using the partial-sum technique [FKL⁺01]. The size of the involved secret key is $(7 + 6 + 4 + 2 + 1 + 0) \times 4 = 80$ bits, and one structure filters incorrect secret-key guesses by a factor of 2^{-4} . Therefore, we need about $80/4 = 20$ structures to uniquely determine the secret key.

Table 3 summarizes the procedure of the partial-sum technique, where the time complexity can be computed as

$$3 \times 2^{64} + 2^{72} + 2^{76} + 2 \times 2^{80} + 2^{84} + 2 \times 2^{92} + 4 \times 2^{88} \approx 2^{93.2}$$

We need to repeat this procedure 20 times to recover the secret-key. Thus, the total time complexity is $2^{97.5}$, the data complexity is $20 \times 2^{64} \approx 2^{68.4}$, and the memory complexity is $1/64 \cdot 2^{88} = 2^{82}$ 64-bit blocks. Note that our attack requires a data complexity above 2^{64} , however, we do not need to collect the full-codebook under a fixed tweakkey.

6.4 Zero-Correlation Linear Hull on SKINNY-64/192

We can reuse parts of the zero-correlation linear hull for SKINNY-64/128 in the TK-2 setting for that of SKINNY-64/192 in the TK-3 setting. Therefore, we apply any linear mask to cells (s_0, s_3) in the input mask Γ_0 . In contrast to the case for SKINNY-64/128, we

Table 3: Procedure for the key-recovery on SKINNY-64/128

Step	Guessed key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{56}	$X_{20}[0,1,3,4,5,6,7,8,10,11,12,13,14,15]$	2^{56}	2^{64}
1	$TK_{20}[1,5]$	2^{52}	$X_{20}[0,3,4,6,7,8,10,11,12,14,15], Y_{19}[1,5]$	$2^{52+8} = 2^{60}$	$2^{56+8} = 2^{64}$
2	$TK_{20}[6]$	2^{52}	$X_{20}[0,3,4,7,8,11,12,15], Y_{19}[1,5,6,10,14]$	$2^{52+12} = 2^{64}$	$2^{52+12} = 2^{64}$
3	$TK_{20}[0,4]$	2^{48}	$X_{20}[3,7,11,15], Y_{19}[0,1,4,5,6,10,12,14]$	$2^{48+20} = 2^{68}$	$2^{52+20} = 2^{72}$
4	$TK_{20}[3,7]$	2^{44}	$Y_{19}[0,1,3,4,5,6,10,11,12,14,15]$	$2^{44+28} = 2^{72}$	$2^{48+28} = 2^{76}$
-		2^{44}	$X_{19}[0,1,3,4,5,7,8,9,12,13,15]$		
5	$TK_{19}[0,4]$	2^{36}	$X_{19}[1,3,5,7,9,13,15], Y_{18}[4,12]$	$2^{36+36} = 2^{72}$	$2^{44+36} = 2^{80}$
6	$TK_{19}[3,7]$	2^{32}	$X_{19}[1,5,9,13], Y_{18}[3,4,12,15]$	$2^{32+44} = 2^{76}$	$2^{36+44} = 2^{80}$
7	$TK_{19}[1,5]$	2^{32}	$Y_{18}[1,3,4,5,9,12,13,15]$	$2^{32+52} = 2^{84}$	$2^{32+52} = 2^{84}$
-		2^{32}	$X_{18}[1,3,4,7,11,12,13,15]$		
8	$TK_{18}[3,7]$	2^{24}	$X_{18}[1,4,12,13], Y_{17}[7,15]$	$2^{24+60} = 2^{84}$	$2^{32+60} = 2^{92}$
9	$TK_{18}[1]$	2^{20}	$X_{18}[4,12], Y_{17}[7,13,15]$	$2^{20+64} = 2^{84}$	$2^{24+64} = 2^{88}$
10	$TK_{18}[4]$	2^{20}	$Y_{17}[0,7,8,13,15]$	$2^{20+68} = 2^{88}$	$2^{20+68} = 2^{88}$
-		2^{20}	$X_{17}[0,6,10,12,14]$		
11	$TK_{17}[6]$	2^{12}	$X_{17}[0,12], Y_{16}[6]$	$2^{12+72} = 2^{84}$	$2^{20+72} = 2^{92}$
12	$TK_{17}[0]$	2^8	$Y_{16}[6,12]$	$2^{8+76} = 2^{84}$	$2^{12+76} = 2^{88}$
12	$TK_{16}[5]$	2^4	$Z_{14}[11]$	$2^{4+80} = 2^{84}$	$2^{8+80} = 2^{88}$

now apply active linear masks to only cell s_9 in the state before `MixColumns`, as shown in Fig. 21. Then, we focus on the tweakey cell labelled 7, and the Γ sequence has now three active cells. Again, by using Proposition 1 and applying an inactive mask to the before mentioned tweakey cell, this causes a contradiction. Note that the remaining 15×3 cells in the tweakey can take any constant, and the domain of our zero-correlation linear hull is $64 + 12 = 76$ bits.

Again, we link the zero-correlation linear hull to a related-tweakey integral distinguisher. We apply any linear mask to the two cells (s_0, s_3) in the zero-correlation linear hulls as illustrated in Fig. 21 and apply inactive linear masks to the remaining 14 cells. Moreover, we apply inactive linear masks to the $3 \times 4 = 12$ -bit tweak cell labelled 7. Therefore, the corresponding related-tweakey integral distinguisher requires $2^{14 \times 4} = 2^{56}$ chosen plaintexts over 2^{12} related tweakeys, and the total data complexity is $2^{56+12} = 2^{68}$. Here, the relation of the tweakey is defined in such a way that the $3 \times 4 = 12$ -bit tweak cell labelled 7 takes all values. The cell s_9 before `MixColumns` is balanced because any linear mask is applied to the cell.

6.5 Key-Recovery Attacks on SKINNY-64/192

Our integral distinguisher uses 2^{12} related tweakeys, and then, there exist generic key-recovery attack with a time complexity of $2^{192-12} = 2^{180}$ [BMV11]. Therefore, the time complexity of a non-trivial key-recovery attack must be at most 2^{180} .

In the key-recovery, we can prepend 1 round and append 8 rounds to the integral distinguisher. In total the attack reaches 23 rounds. Figure 22 shows the key-recovery, and let X_i , Y_i , and Z_i be the states as defined in Fig. 20. Let P and T be the states of plaintext and tweak, respectively. We first prepare a set of chosen Z_1 , where 14 cells are active, i.e., $Z_1[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$. Moreover, we need the tweak cell $T[11]$ in all three lines of the tweak-schedule to be active, as it will propagate to cell $T[7]$ after

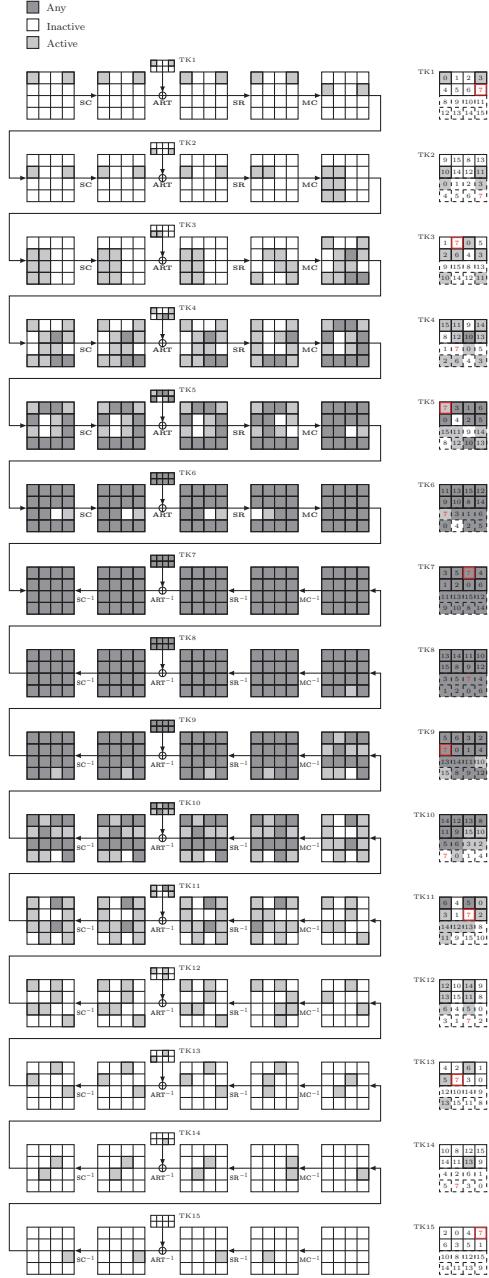


Figure 21: Zero-correlation linear hull for SKINNY-64/192.

one round, which coincides with the beginning of both integral distinguishers as shown in Fig. 21. Note that the consistent set of chosen plaintexts is computed from Z_1 without guessing any bits.

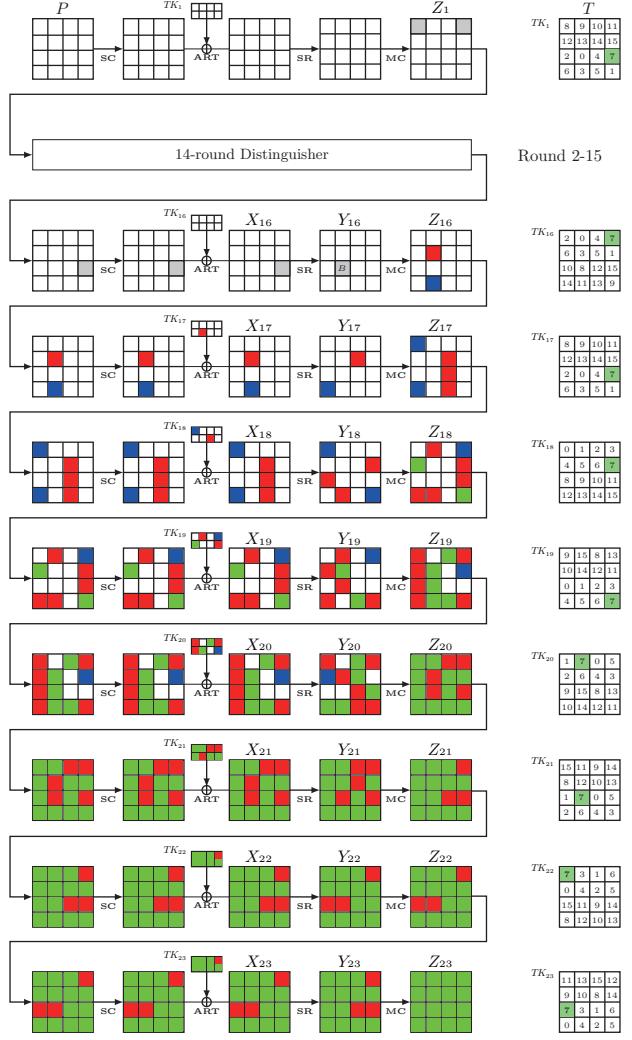


Figure 22: Key-recovery attack on 23-rounds of SKINNY-64/192.

Due to the integral distinguisher s_9 in Y_{16} is balanced. Then

$$\begin{pmatrix} Z_{16}[1] \\ Z_{16}[5] \\ Z_{16}[9] \\ Z_{16}[13] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} Y_{16}[1] \\ Y_{16}[5] \\ Y_{16}[9] \\ Y_{16}[13] \end{pmatrix} = \begin{pmatrix} Y_{16}[1] + Y_{16}[9] + Y_{16}[13] \\ Y_{16}[1] \\ Y_{16}[5] + Y_{16}[9] \\ Y_{16}[1] + Y_{16}[9] \end{pmatrix},$$

and

$$\bigoplus Z_{16}[5] + Z_{16}[13] = \bigoplus Y_{16}[9] = 0$$

Similarly to the attack against QARMA, we use the meet-in-the-middle technique for the integral attack [SW13], where $\bigoplus Z_{16}[5]$ and $\bigoplus Z_{16}[13]$ are independently evaluated, and round-tweakeys satisfying $\bigoplus Z_{16}[5] = \bigoplus Z_{16}[13]$ are recovered. The size of involved secret-tweakey is 148($= 37 \times 4$) bits. Since one structure removes incorrect secret-key guesses by a factor of 2^{-4} , we need $148/4 = 37$ structures to uniquely determine the secret-key.

Table 4: Procedure for the key-recovery of $\bigoplus Z_{16}[5]$ on SKINNY-64/192.

Step	Guessed key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{68}	$X_{23}[0, 1, 2, \dots, 15], \Delta TK_{22}[0]$	2^{68}	2^{68}
1	$TK_{23}[0, \dots, 7]$	2^{68}	$X_{22}[0, 1, 2, \dots, 15], \Delta TK_{22}[0]$	$2^{68+32} = 2^{100}$	$2^{68+32} = 2^{100}$
2	$TK_{22}[0, \dots, 7]$	2^{64}	$X_{21}[0, 1, 2, \dots, 15]$	$2^{64+64} = 2^{128}$	$2^{68+64} = 2^{132}$
3	$TK_{21}[0, 4]$	2^{56}	$X_{21}[1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15], Y_{20}[0, 12]$	$2^{56+72} = 2^{128}$	$2^{64+72} = 2^{136}$
4	$TK_{21}[1, 5]$	2^{48}	$X_{21}[2, 3, 6, 7, 10, 11, 14, 15], Y_{20}[0, 5, 12, 13]$	$2^{48+80} = 2^{128}$	$2^{56+80} = 2^{136}$
5	$TK_{21}[2, 6]$	2^{48}	$X_{21}[3, 7, 11, 15], Y_{20}[0, 2, 5, 6, 10, 12, 13, 14]$	$2^{48+88} = 2^{136}$	$2^{48+88} = 2^{136}$
6	$TK_{21}[3, 7]$	2^{44}	$Y_{20}[0, 2, 3, 5, 6, 10, 11, 12, 13, 14, 15]$	$2^{44+96} = 2^{140}$	$2^{48+96} = 2^{144}$
-		2^{44}	$X_{20}[0, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15]$		
7	$TK_{20}[0, 4]$	2^{36}	$X_{20}[2, 3, 5, 9, 13, 14, 15], Y_{19}[4, 12]$	$2^{36+104} = 2^{140}$	$2^{44+104} = 2^{148}$
8	$TK_{20}[5]$	2^{36}	$X_{20}[2, 3, 14, 15], Y_{19}[1, 4, 5, 9, 12]$	$2^{36+108} = 2^{144}$	$2^{36+108} = 2^{144}$
9	$TK_{20}[2]$	2^{32}	$X_{20}[3, 15], Y_{19}[1, 4, 5, 9, 12, 14]$	$2^{32+112} = 2^{140}$	$2^{36+112} = 2^{144}$
10	$TK_{20}[3]$	2^{28}	$Y_{19}[1, 4, 5, 9, 12, 14, 15]$	$2^{28+116} = 2^{144}$	$2^{32+116} = 2^{148}$
-		2^{28}	$X_{19}[1, 4, 7, 11, 12, 13, 15]$		
11	$TK_{19}[4]$	2^{24}	$X_{19}[1, 7, 11, 13, 15], Y_{18}[8]$	$2^{24+120} = 2^{144}$	$2^{28+120} = 2^{148}$
12	$TK_{19}[1]$	2^{20}	$X_{19}[7, 11, 15], Y_{18}[8, 13]$	$2^{20+124} = 2^{144}$	$2^{24+124} = 2^{148}$
13	$TK_{19}[7]$	2^{12}	$Y_{18}[7, 8, 13]$	$2^{12+128} = 2^{140}$	$2^{20+128} = 2^{148}$
-		2^{12}	$X_{18}[6, 10, 14]$		
14	$TK_{18}[6]$	2^4	$Y_{17}[6]$	$2^{4+132} = 2^{140}$	$2^{12+132} = 2^{144}$
-		2^4	$X_{17}[5]$		
15	$TK_{17}[5]$	2^4	$Z_{16}[5]$	$2^{4+136} = 2^{140}$	$2^{4+136} = 2^{140}$

To compute $\bigoplus Z_{16}[5]$, all cells labelled red and green are involved. We use the partial-sum technique to recover $\bigoplus Z_{16}[5]$, and the procedure is summarized in Table 4. Here, $\Delta TK_{22}[0]$ is computed from the relation of tweakeys. Then, the time complexity can be computed by

$$2^{68} + 2^{100} + 2^{132} + 3 \times 2^{136} + 2^{140} + 4 \times 2^{144} + 5 \times 2^{148} \approx 2^{150.4}$$

We also need to compute $\bigoplus Z_{16}[13]$, but the time complexity is clearly negligible compared with that for $\bigoplus Z_{16}[5]$. Nevertheless, the procedure to recover $\bigoplus Z_{16}[13]$ is shown in Table 5, where we consider all cells labelled blue and green. Again, we use the partial-sum technique to recover $\bigoplus Z_{16}[13]$, and the time complexity can be computed by

$$2^{68} + 2^{84} + 2^{112} + 3 \times 2^{108} + 5 \times 2^{104} + 2 \times 2^{100} \approx 2^{112.3}$$

We need to repeat these two procedures for 37 times to recover the secret-key. Thus, the total time complexity can be computed by

$$37 \times (2^{150.4} + 2^{112.3}) \approx 2^{155.6}$$

the data complexity is $37 \times 2^{68} \approx 2^{73.2}$, and the memory complexity is $1/64 \cdot 2^{144} = 2^{138}$ 64-bit blocks. Finally, we compare these lists and find a match. Similarly to the attack against SKINNY-64/128, our attack requires a data complexity above 2^{64} , however, we do not need to collect the full-codebook under a fixed tweakey.

Table 5: Procedure for the key-recovery of $\bigoplus Z_{16}[13]$ on SKINNY-64/192.

Step	Guessed key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{56}	$X_{23}[0, 1, 2, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15], \Delta TK_{22}[0]$	2^{56}	2^{68}
1	$TK_{23}[0, 1, 2, 4, 5, 6, 7]$	2^{56}	$X_{22}[0, 1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15], \Delta TK_{22}[0]$	$2^{56+28} = 2^{84}$	$2^{56+28} = 2^{84}$
2	$TK_{22}[0, 1, 2, 4, 5, 6, 7]$	2^{44}	$X_{21}[0, 1, 4, 6, 7, 8, 10, 12, 13, 14, 15]$	$2^{44+56} = 2^{100}$	$2^{56+56} = 2^{112}$
3	$TK_{21}[0, 4]$	2^{36}	$X_{21}[1, 6, 7, 10, 13, 14, 15], Y_{20}[4, 12]$	$2^{36+64} = 2^{100}$	$2^{44+64} = 2^{108}$
4	$TK_{21}[1]$	2^{32}	$X_{21}[6, 7, 10, 14, 15], Y_{20}[4, 12, 13]$	$2^{32+68} = 2^{100}$	$2^{36+68} = 2^{104}$
5	$TK_{21}[6]$	2^{28}	$X_{21}[7, 15], Y_{20}[2, 4, 6, 12, 13]$	$2^{28+72} = 2^{100}$	$2^{32+72} = 2^{104}$
6	$TK_{21}[7]$	2^{24}	$Y_{20}[2, 4, 6, 11, 12, 13]$	$2^{24+76} = 2^{100}$	$2^{28+76} = 2^{104}$
-		2^{24}	$X_{20}[2, 5, 7, 9, 13, 14]$		
7	$TK_{20}[5]$	2^{16}	$X_{20}[2, 7, 14], Y_{19}[5]$	$2^{16+80} = 2^{96}$	$2^{24+80} = 2^{104}$
8	$TK_{20}[2]$	2^{12}	$X_{20}[7], Y_{19}[5, 14]$	$2^{12+84} = 2^{96}$	$2^{16+84} = 2^{100}$
9	$TK_{20}[7]$	2^{12}	$Y_{19}[3, 5, 14]$	$2^{12+88} = 2^{100}$	$2^{12+88} = 2^{100}$
-		2^{12}	$X_{19}[3, 4, 15]$		
10	$TK_{19}[4]$	2^{12}	$X_{19}[3, 15], Y_{18}[0]$	$2^{12+92} = 2^{104}$	$2^{12+92} = 2^{104}$
11	$TK_{19}[3]$	2^8	$Y_{18}[0, 15]$	$2^{8+96} = 2^{104}$	$2^{12+96} = 2^{108}$
-		2^8	$X_{18}[0, 12]$		
12	$TK_{18}[0]$	2^4	$Y_{17}[12] = X_{17}[13] = Z_{16}[13]$	$2^{4+100} = 2^{104}$	$2^{8+100} = 2^{108}$

7 Conclusion

In this paper, we study zero-correlation attacks on tweakable block ciphers and consider for the first time the effect of the tweak. Kranz, Leander and Wiemer [LTW18] showed that the addition of the tweak, that is updated by a linear key schedule, does not introduce new linear characteristics, which is quite different to the differential model. However, the given additional restrictions from the linear tweak schedule allow us to efficiently find zero-correlation linear hulls for tweakable block ciphers.

Turning the zero-correlation distinguisher into integral distinguishers allows us to show new attacks on round-reduced variants of QARMA, MANTIS and SKINNY, where the attack on QARMA is currently the best attack (with respect to number of rounds) on a round-reduced variant of QARMA. This new way of searching for distinguishers on tweakable block ciphers does not only allow attackers to find longer distinguishers, but also provides designers of tweakable block ciphers with new insights. For example in tweakable reflection ciphers like MANTIS or QARMA, where the tweak is added just in the forward and backward rounds, while in the middle rounds just round-keys are added, the additional middle rounds do not provide extra security with respect to our attacks. This is because the zero-correlation linear hulls over the tweaks are independent of the number of keyed middle rounds.

Acknowledgments. The research leading to the presented results started during the Flexible Symmetric Cryptography workshop at the Lorentz Center in Leiden, the Netherlands. Christoph Dobraunig is supported by the Austrian Science Fund (FWF): J 4277-N38. This research is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its Strategic Capability Research Centres Funding Initiative, Singapore Ministry of Education under Research Grant M4012049, and Nanyang Technological University under Grant M4082123. Ralph Ankele is supported by the European Union’s Horizon 2020 research and innovation programme under grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET.

References

- [ABC⁺17] Ralph Ankele, Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. Related-key impossible-differential attack on reduced-round skinny. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 208–228, Kanazawa, Japan, July 10–12, 2017. Springer, Heidelberg, Germany.
- [AK19] Ralph Ankele and Stefan Kölbl. Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In Carlos Cid and Michael J. Jacobson Jr., editors, *SAC 2018: 25th Annual International Workshop on Selected Areas in Cryptography*, volume 11349 of *Lecture Notes in Computer Science*, pages 163–190, Calgary, AB, Canada, August 15–17, 2019. Springer, Heidelberg, Germany.
- [Ava17] Roberto Avanzi. The QARMA block cipher family. *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, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.
- [BBR⁺13] Andrey Bogdanov, Christina Boura, Vincent Rijmen, Meiqin Wang, Long Wen, and Jingyuan Zhao. Key difference invariant bias in block ciphers. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 357–376, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.
- [BCG⁺12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, 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 - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.
- [BCLR17] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving resistance against invariant attacks: How to choose the round constants. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 647–678, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.
- [Bey18] Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 3–31, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.
- [Bih94] Eli Biham. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, 7(4):229–246, December 1994.

- [BJK⁺16] Christof Beierle, Jérémie 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, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.
- [BLNW12] Andrey Bogdanov, Gregor Leander, Kaisa Nyberg, and Meiqin Wang. Integral and multidimensional linear distinguishers with correlation zero. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 244–261, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.
- [BMV11] Asli Bay, Atefeh Mashatan, and Serge Vaudenay. A related-key attack against multiple encryption based on fixed points. In Mohammad S. Obaidat, José Luis Sevillano, and Joaquim Filipe, editors, *ICETE 2011*, volume 314 of *Communications in Computer and Information Science*, pages 264–280. Springer, 2011.
- [BR14] Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Designs, Codes and Cryptography*, 70(3):369–383, Mar 2014.
- [BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO’90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany.
- [BSS⁺13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. <http://eprint.iacr.org/2013/404>.
- [BW12] Andrey Bogdanov and Meiqin Wang. Zero correlation linear cryptanalysis with reduced data complexity. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 29–48, Washington, DC, USA, March 19–21, 2012. Springer, Heidelberg, Germany.
- [CAE14] CAESAR committee. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness, 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.
- [CHP⁺18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 683–714, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.
- [Dae95] Joan Daemen. *Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis*. K.U.Leuven, 1995. <http://jda.noekeon.org/>.

- [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. <http://tosc.iacr.org/index.php/ToSC/article/view/573>.
- [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 16: 14th International Conference on Applied Cryptography and Network Security*, volume 9696 of *Lecture Notes in Computer Science*, pages 500–517, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany.
- [DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *Fast Software Encryption – FSE’97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Haifa, Israel, January 20–22, 1997. Springer, Heidelberg, Germany.
- [DL17] Christoph Dobraunig and Eik List. Impossible-differential and boomerang cryptanalysis of round-reduced kiasu-BC. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 207–222, San Francisco, CA, USA, February 14–17, 2017. Springer, Heidelberg, Germany.
- [EK17] Maria Eichlseder and Daniel Kales. Clustering related-tweak characteristics: Application to mantis-6. Cryptology ePrint Archive, Report 2017/1136, 2017. <https://eprint.iacr.org/2017/1136>.
- [EK18] Maria Eichlseder and Daniel Kales. Clustering related-tweak characteristics: Application to MANTIS-6. *IACR Transactions on Symmetric Cryptology*, 2018(2):111–132, 2018.
- [EKKT19] Zahra Eskandari, Andreas Brasen Kidmose, Stefan Kölbl, and Tyge Tiessen. Finding integral distinguishers with ease. In Carlos Cid and Michael J. Jacobson Jr., editors, *SAC 2018: 25th Annual International Workshop on Selected Areas in Cryptography*, volume 11349 of *Lecture Notes in Computer Science*, pages 115–138, Calgary, AB, Canada, August 15–17, 2019. Springer, Heidelberg, Germany.
- [FKL⁺01] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230, New York, NY, USA, April 10–12, 2001. Springer, Heidelberg, Germany.
- [JNP14] Jérémie Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.
- [JNP15a] Jérémie Jean, Ivica Nikolic, and Thomas Peyrin. Deoxys v1.3. <https://competitions.cr.yp.to/round2/deoxysv13.pdf>, 2015.
- [JNP15b] Jérémie Jean, Ivica Nikolic, and Thomas Peyrin. Joltik v1.3. <https://competitions.cr.yp.to/round2/joltikv13.pdf>, 2015.
- [JNP15c] Jérémie Jean, Ivica Nikolic, and Thomas Peyrin. Kiasu v1. <https://competitions.cr.yp.to/round1/kiasuv1.pdf>, 2015.

- [KLW17] Thorsten Kranz, Gregor Leander, and Friedrich Wiemer. Linear cryptanalysis: Key schedules and tweakable block ciphers. *IACR Transactions on Symmetric Cryptology*, 2017(1):474–505, 2017.
- [KR11] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327, Lyngby, Denmark, February 13–16, 2011. Springer, Heidelberg, Germany.
- [KW02] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption – FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127, Leuven, Belgium, February 4–6, 2002. Springer, Heidelberg, Germany.
- [Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Richard E. Blahut, Daniel J. Costello Jr., Ueli Maurer, and Thomas Mittelholzer, editors, *Communications and Cryptography: Two Sides of One Tapestry*, volume 276 of *International Series in Engineering and Computer Science*, pages 227–233. Kluwer Academic Publishers, 1994.
- [LGS17] Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakkey settings (long paper). *IACR Transactions on Symmetric Cryptology*, 2017(3):37–72, 2017.
- [LJ18] Rongjia Li and Chenhui Jin. Meet-in-the-middle attacks on reduced-round QARMA-64/128. *Comput. J.*, 61(8):1158–1165, 2018.
- [LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.
- [LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, July 2011.
- [LST12] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable blockciphers with beyond birthday-bound security. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 14–30, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.
- [LTW18] Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. Searching for subspace trails and truncated differentials. *IACR Transactions on Symmetric Cryptology*, 2018(1):74–100, 2018.
- [Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT’93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Lofthus, Norway, May 23–27, 1994. Springer, Heidelberg, Germany.
- [Men15] Bart Mennink. Optimally secure tweakable blockciphers. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 428–448, Istanbul, Turkey, March 8–11, 2015. Springer, Heidelberg, Germany.
- [MvV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

- [Nyb95] Kaisa Nyberg. Linear approximation of block ciphers (rump session). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT’94*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany.
- [Nyb01] Kaisa Nyberg. Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, 111(1-2):177–188, 2001.
- [PS16] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 33–63, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.
- [Qua17] Qualcomm Product Security. Pointer authentication on ARMv8.3 – design and analysis of the new software security instructions., January 2017. <https://www.qualcomm.com/documents/whitepaper-pointer-authentication-armv83>.
- [Sas18] Yu Sasaki. Improved related-tweakey boomerang attacks on deoxys-BC. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18: 10th International Conference on Cryptology in Africa*, volume 10831 of *Lecture Notes in Computer Science*, pages 87–106, Marrakesh, Morocco, May 7–9, 2018. Springer, Heidelberg, Germany.
- [Sch98] Rich Schroeppel. Hasty pudding cipher, 1998.
- [SLR⁺15] Bing Sun, Zhiqiang Liu, Vincent Rijmen, Ruilin Li, Lei Cheng, Qingju Wang, Hoda AlKhzaimi, and Chao Li. Links among impossible differential, integral and zero correlation linear cryptanalysis. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 95–115, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.
- [SMB18] Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR Transactions on Symmetric Cryptology*, 2018(3):124–162, 2018.
- [ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.
- [SW13] Yu Sasaki and Lei Wang. Meet-in-the-middle technique for integral attacks against Feistel ciphers. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 234–251, Windsor, Ontario, Canada, August 15–16, 2013. Springer, Heidelberg, Germany.
- [TA14] Yosuke Todo and Kazumaro Aoki. FFT key recovery for integral attack. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 14: 13th International Conference on Cryptology and Network Security*, volume 8813 of *Lecture Notes in Computer Science*, pages 64–81, Heraklion, Crete, Greece, October 22–24, 2014. Springer, Heidelberg, Germany.

- [TAY16] Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. A meet in the middle attack on reduced round kiasu-bc. *IEICE Transactions*, 99-A(10):1888–1890, 2016.
- [TAY17] Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. Impossible differential cryptanalysis of reduced-round SKINNY. In Marc Joye and Abderrahmane Nitaj, editors, *AFRICACRYPT 17: 9th International Conference on Cryptology in Africa*, volume 10239 of *Lecture Notes in Computer Science*, pages 117–134, Dakar, Senegal, May 24–26, 2017. Springer, Heidelberg, Germany.
- [TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *Fast Software Encryption – FSE 2016*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377, Bochum, Germany, March 20–23, 2016. Springer, Heidelberg, Germany.
- [Tod15a] Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 413–432, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.
- [Tod15b] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
- [Tod17] Yosuke Todo. Integral cryptanalysis on full MISTY1. *Journal of Cryptology*, 30(3):920–959, July 2017.
- [WGZ⁺16] Lei Wang, Jian Guo, Guoyan Zhang, Jingyuan Zhao, and Dawu Gu. How to build fully secure tweakable blockciphers from classical blockciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 455–483, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
- [YQC17] Dong Yang, Wen-Feng Qi, and Hua-Jin Chen. Impossible differential attacks on the SKINNY family of block ciphers. *IET Information Security*, 11(6):377–385, 2017.
- [YQC18] Dong Yang, Wen-Feng Qi, and Hua-Jin Chen. Impossible differential attack on qarma family of block ciphers. Cryptology ePrint Archive, Report 2018/334, 2018. <https://eprint.iacr.org/2018/334>.
- [ZD16] Rui Zong and Xiaoyang Dong. Meet-in-the-middle attack on QARMA block cipher. Cryptology ePrint Archive, Report 2016/1160, 2016. <http://eprint.iacr.org/2016/1160>.
- [ZDW18] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Milp-aided related-tweak/key impossible differential attack and its applications to qarma, joltik-bc. Cryptology ePrint Archive, Report 2018/142, 2018. <https://eprint.iacr.org/2018/142>.
- [ZR17] Wenyng Zhang and Vincent Rijmen. Division cryptanalysis of block ciphers with a binary diffusion layer. Cryptology ePrint Archive, Report 2017/188, 2017. <http://eprint.iacr.org/2017/188>.

A Experimental Verification of Our Distinguishers

For the TK-1, we used the toy cipher described in Example 1, but the size of S-box is shrunk to 4 bits and the 4-bit S-box of QARMA is used. As depicted in Fig. 11, any linear mask is applied to cells except for the first diagonal elements. Then, the Γ sequence of the first nibble of the tweak has 1 active nibble. Therefore, the corresponding related-tweak integral distinguisher requires $2^{4 \times 4} = 2^{16}$ chosen plaintexts under 2^4 related tweaks. Then, the first nibble in the output is balanced. We implemented and verified the distinguisher.

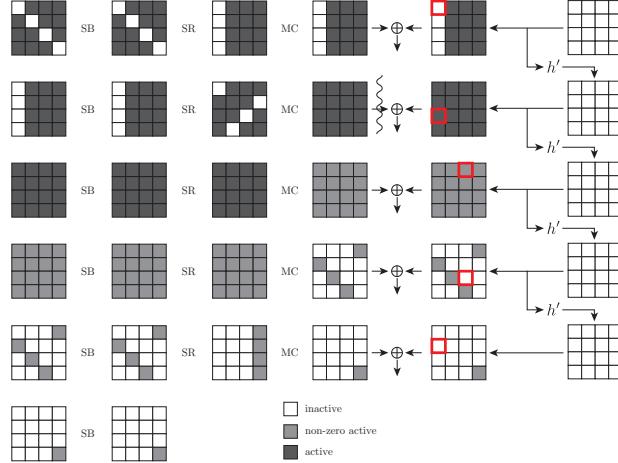


Figure 23: Verification of our distinguisher for TK-2

Similarly to the case of TK-1, we also verified the case of the TK-2 by using the modified toy cipher. We added additional 1-line tweakey schedule in the toy cipher above, where the multiplication by 2 over $GF(2^4)$ is applied to every cell. Figure 23 shows the zero-correlation linear hull. The Γ sequence of the first nibble of the tweak has at most 2 active nibbles. Therefore, the corresponding related-tweak integral distinguisher requires $2^{4 \times 4} = 2^{16}$ chosen plaintexts under 2^8 related tweaks. Then, the last nibble in the output is balanced, and we implemented and verified the distinguisher too.

B Source code for the distinguishers of the Toy cipher

```

1 #include<stdio.h>
2 #include<iostream>
3 #include<vector>
4 #include<time.h>
5
6 using namespace std;
7
8 int gmul2(int in) {
9     int out = (in << 1) & 0xF;
10    out ^= ((in >> 3) * 0x3);
11    return out;
12 }
13
14 int gmul3(int in) {
15     return (in ^ gmul2(in));
16 }
17
18 vector<vector<int>> subByte(vector<vector<int>> in, vector<vector<int>> rk1,
19                                vector<vector<int>> rk2) {

```

```

19     vector<int> sbox = { 10, 13, 14, 6, 15, 7, 3, 5, 9, 8, 0, 12, 11, 1, 2, 4
20         };
21     vector<vector<int>> out = in;
22     for (int i = 0; i < 4; i++) {
23         for (int j = 0; j < 4; j++) {
24             out[i][j] = sbox[in[i][j]] ^ rk1[i][j] ^ rk2[i][j];
25         }
26     }
27     return out;
28 }
29 vector<vector<int>> subByte(vector<vector<int>> in, vector<vector<int>> rk)
30 {
31     vector<int> sbox = { 0x1, 0x0, 0x5, 0x3, 0xe, 0x2, 0xf, 0x7, 0xd, 0xa, 0x9
32         , 0xb, 0xc, 0x8, 0x4, 0x6 };
33     vector<vector<int>> out = in;
34     for (int i = 0; i < 4; i++) {
35         for (int j = 0; j < 4; j++) {
36             out[i][j] = sbox[in[i][j]] ^ rk[i][j];
37         }
38     }
39     return out;
40 }
41 vector<vector<int>> subByte(vector<vector<int>> in) {
42     vector<int> sbox = { 0x1, 0x0, 0x5, 0x3, 0xe, 0x2, 0xf, 0x7, 0xd, 0xa, 0x9
43         , 0xb, 0xc, 0x8, 0x4, 0x6 };
44     vector<vector<int>> out = in;
45     for (int i = 0; i < 4; i++) {
46         for (int j = 0; j < 4; j++) {
47             out[i][j] = sbox[in[i][j]];
48         }
49     }
50     return out;
51 }
52 //AES-ShiftRows
53 vector<vector<int>> shiftRow(vector<vector<int>> in) {
54     vector<vector<int>> out = in;
55     for (int i = 0; i < 4; i++) {
56         for (int j = 0; j < 4; j++) {
57             out[i][j] = in[i][(i+j)%4];
58         }
59     }
60     return out;
61 }
62 //AES-MixColumns
63 vector<vector<int>> mixColumn(vector<vector<int>> in) {
64     vector<vector<int>> out = in;
65     for (int j = 0; j < 4; j++) {
66         out[0][j] = gmul2(in[0][j]) ^ gmul3(in[1][j]) ^ in[2][j] ^ in[3][j];
67         out[1][j] = gmul2(in[1][j]) ^ gmul3(in[2][j]) ^ in[3][j] ^ in[0][j];
68         out[2][j] = gmul2(in[2][j]) ^ gmul3(in[3][j]) ^ in[0][j] ^ in[1][j];
69         out[3][j] = gmul2(in[3][j]) ^ gmul3(in[0][j]) ^ in[1][j] ^ in[2][j];
70     }
71     return out;
72 }
73 //Tweakey schedule for TK1
74 vector<vector<int>> keySchedule(vector<vector<int>> in) {
75     //SKINNY-permutation
76     vector<int> ks = { 9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7 };
77     vector<vector<int>> out(4, vector<int>(4, 0));
78     for (int i = 0; i < 16; i++) {
79         out[i / 4][(i % 4)] = in[ks[i] / 4][ks[i] % 4];
80     }

```

```

81     }
82     return out;
83 }
84
85 //Tweakey schedule for TK2
86 vector<vector<int>> keySchedule2(vector<vector<int>> in) {
87     //SKINNY-permutation
88     vector<int> ks = { 9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7 };
89     vector<vector<int>> out(4, vector<int>(4, 0));
90     for (int i = 0; i < 16; i++) {
91         out[i / 4][(i % 4)] = gmul2(in[ks[i] / 4][ks[i] % 4]);
92     }
93     return out;
94 }
95
96 //Test distinguisher for tweakable BC with one tweakey line
97 int testTK1(void){
98     // generate all keys at random
99     strand(time(NULL));
100    vector<vector<int>> key(4, vector<int>(4, 0));
101    for (int row = 0; row < 4; row++) {
102        for (int col = 0; col < 4; col++) {
103            key[row][col] = rand() & 0xF;
104        }
105    }
106
107    int evalNumRounds = 5;
108    printf(" Number of rounds : %d\n", evalNumRounds);
109
110    vector<int> counter(16, 0);
111    for (int i1 = 0; i1 < 16; i1++) {
112        for (int i2 = 0; i2 < 16; i2++) {
113            for (int i3 = 0; i3 < 16; i3++) {
114                for (int i4 = 0; i4 < 16; i4++) {
115                    for (int i5 = 0; i5 < 16; i5++) {
116                        vector<vector<int>> in(4, vector<int>(4, 0));
117                        vector<vector<int>> tk = key;
118                        in[0][0] = i1;
119                        in[1][1] = i2;
120                        in[2][2] = i3;
121                        in[3][3] = i4;
122                        tk[0][0] = i5;
123
124                        // encryption
125                        in = subByte(in);
126                        for (int r = 0; r < (evalNumRounds - 1); r++) {
127                            in = shiftRow(in);
128                            in = mixColumn(in);
129                            in = subByte(in, tk);
130                            tk = keySchedule(tk);
131                        }
132                        counter[in[0][0]]++;
133                    }
134                }
135            }
136        }
137    }
138
139    for (int i = 0; i < 16; i++) {
140        printf("%X : frequency of appearance = %5d, mod 2 = %d \n", i,
141               counter[i], counter[i] % 2);
142    }
143    cout << endl;
144 }
145 //Test distinguisher for tweakable BC with two tweakey lines

```

```

146 int testTK2(void){
147     // generate all keys at random
148     srand(time(NULL));
149     vector<vector<int>> key1(4, vector<int>(4, 0));
150     vector<vector<int>> key2(4, vector<int>(4, 0));
151     for (int row = 0; row < 4; row++) {
152         for (int col = 0; col < 4; col++) {
153             key1[row][col] = rand() & 0xF; key2[row][col] = rand() & 0xF;
154         }
155     }
156
157     int evalNumRounds = 6;
158     printf(" Number of rounds : %d\n", evalNumRounds);
159
160     vector<int> counter(16, 0);
161     for (int i1 = 0; i1 < 16; i1++) {
162         for (int i2 = 0; i2 < 16; i2++) {
163             for (int i3 = 0; i3 < 16; i3++) {
164                 for (int i4 = 0; i4 < 16; i4++) {
165                     for (int i5 = 0; i5 < 16; i5++) {
166                         for (int i6 = 0; i6 < 16; i6++) {
167                             vector<vector<int>> in(4, vector<int>(4, 0));
168                             vector<vector<int>> tk1 = key1;
169                             vector<vector<int>> tk2 = key2;
170                             in[0][0] = i1;
171                             in[1][1] = i2;
172                             in[2][2] = i3;
173                             in[3][3] = i4;
174                             tk1[0][0] = i5;
175                             tk2[0][0] = i6;
176
177                             // encryption
178                             in = subByte(in);
179                             for (int r = 0; r < (evalNumRounds - 1); r++) {
180                                 in = shiftRow(in);
181                                 in = mixColumn(in);
182                                 in = subByte(in, tk1, tk2);
183                                 tk1 = keySchedule(tk1);
184                                 tk2 = keySchedule2(tk2);
185                             }
186                             counter[in[3][3]]++;
187                         }
188                     }
189                 }
190             }
191         }
192     }
193
194     for (int i = 0; i < 16; i++) {
195         printf("%X : frequency of appearance = %5d, mod 2 = %d \n", i,
196               counter[i], counter[i] % 2);
197     }
198     cout << endl;
199 }
200
201 int main(void) {
202     printf("Experimental verification of distinguisher on TK1.\n");
203     testTK1();
204
205     printf("Experimental verification of distinguisher on TK2.\n");
206     testTK2();
207     return 0;
208 }
```

C Figures about Application to MANTIS

C.1 Zero-Correlation Linear Hull on MANTIS_{4,5}

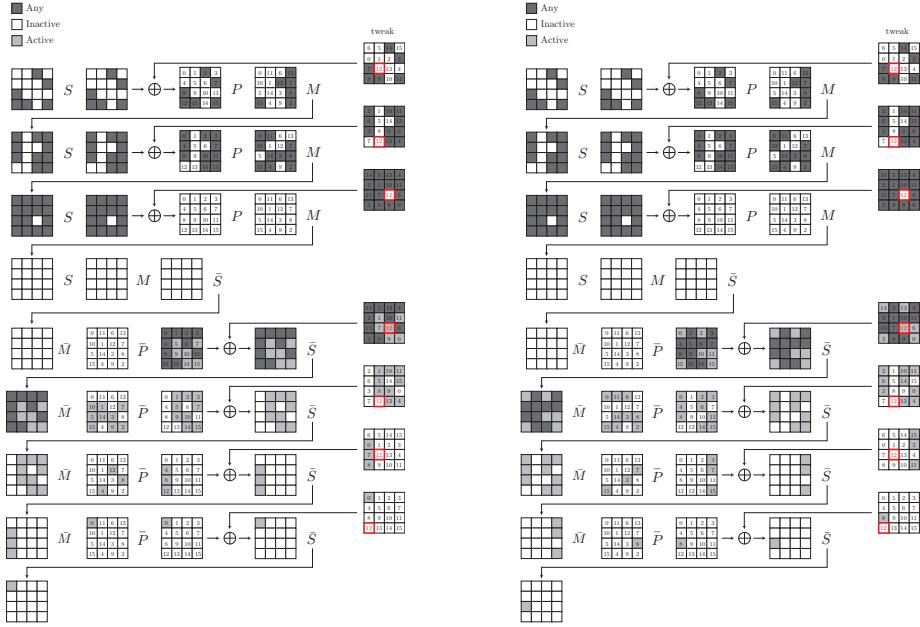


Figure 24: Two zero-correlation linear hulls on MANTIS_{4,5}.

C.2 Key Recovery Attacks on MANTIS_{4,8}

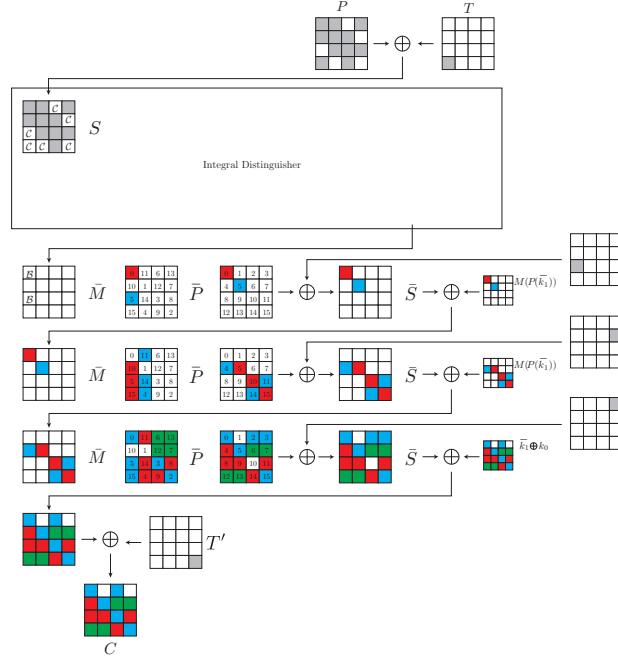


Figure 25: Key-recovery attack on MANTIS_{4,8}.