Exploiting Strong Key Bridges: Full-Fledged Automatic Rectangle Attacks on Deoxys-BC and SKINNY

Ling Song[1], Yincen Chen[1,6], Qianqian Yang[2,5,7✉], Huimin Liu[1], Lei Wang[3✉], Lei Hu[2,7], and Jian Weng[4]

[1] College of Cyber Security, Jinan University, Guangzhou, China
[2] State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[4] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
[5] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[6] Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France
[7] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
songling.qs@gmail.com,yincenc@gmail.com,yangqianqian@iie.ac.cn, liuhuimin301@gmail.com,wanglei_hb@sjtu.edu.cn, hulei@iie.ac.cn,cryptjweng@gmail.com

Abstract. The TWEAKEY framework provides a generic construction for designing tweakable block ciphers. Prominent instances are Deoxys-BC and SKINNY, which have been standardized by ISO/IEC. In this paper, we analyze the tweakey schedules of these ciphers and identify strong dependencies between certain subtweakeys, which we call *strong key bridges*. We then exploit these dependencies in rectangle attacks under the related-tweakey setting. Moreover, we develop a comprehensive constraint programming model to search for rectangle attacks. Our model not only unifies the distinguisher and the key-recovery part while permitting arbitrary key-guessing strategies, but also integrates three new components, *i.e.*, the state-test technique, explicit last-step computation, and the strong key bridges. As a result, we obtain significantly improved cryptanalytic results on both Deoxys-BC and SKINNY. For Deoxys-BC-384 and Deoxys-BC-256, we reduce the time complexity by a factor of 2^{40} and 2^{32}, respectively, yielding the best attacks to date. Moreover, we extend the longest existing attacks on the authenticated encryption schemes, Deoxys-I-256 and Deoxys-I-128, by one round. For SKINNY, we improve upon prior best attacks by one more round for SKINNY-128-384 and by two more rounds for SKINNY-64-192.

Keywords: TWEAKEY framework, Rectangle attack, Automatic search, Deoxys-BC, SKINNY

1 Introduction

Tweakable block cipher formalized by Liskov *et al.* [27] is well suited for building modes of operation to achieve higher security and to achieve multiple functionalities simultaneously (*i.e.*, authenticated encryption). Hence, it has become a significant cornerstone for designing symmetric-key cryptographic schemes. Examples include highly secure message authentication codes [17,23,32] and various authenticated encryption schemes in different setting and with flexible security goals [21, 22, 33, 34, 40]. The trends can also be witnessed by the candidates submitted to the CAESAR Competition [1] and NIST lightweight cryptography competition [2].

Specifically, the tweakable block cipher introduces an additional input *tweak* to the classical block cipher and behaves as a family of blockciphers indexed by the tweak. Natural and important challenges arise: how to effectively insert a tweak into a block cipher to obtain a tweakable block cipher, and *how to accurately analyze the security bound of practical instantiations of tweakable block cipher?*

The tweakable block cipher designs can be categorized into modular and dedicated ad-hoc constructions. For the former category, it is either building a tweakable block cipher from a traditional block cipher in a black-box fashion, such as LRW1 and LRW2 [27], or tweaking generic constructions of block cipher, such as tweaking Luby-Rackoff cipher [16] and tweaking Key-Alternating cipher [12]. Typically, these modular constructions obtain concrete security proofs, assuming that the underlying primitives are ideally secure. However, their security bounds and efficiency are rather unsatisfactory and therefore in continuous development [26, 29, 36, 46]. For the latter category, it is designing a tweakable block cipher from scratch, that is, integrating the design of a dedicated block cipher and the design of the tweak injection method, such as Hasty Pudding Cipher [38] and Threefish [15]. Dedicated designs are highly promising for meeting the performance requirements of practical applications. On the other hand, their security must be thoroughly scrutinized.

Jean *et al.* proposed the `TWEAKEY` framework [24] for building dedicated tweakable block ciphers with arbitrary tweak and key sizes from a public round function such as the `AES` round function. A notable `TWEAKEY` specialization is the `STK` (Superposition TWEAKEY) construction. Prominent instances are `Deoxys-BC` [25] and `SKINNY` [6], which have been standardized by ISO/IEC [3]. `Deoxys-BC` consists of `Deoxys-BC-384` and `Deoxys-BC-256`, and serves as the underlying primitive of the Authenticated Encryption scheme `Deoxys` [25], where `Deoxys-I` was selected as a third-round candidate of the CAESAR competition and `Deoxys-II` is a finalist. `SKINNY` is a family of lightweight tweakable block ciphers. Depending on the block size, it has two instances: `SKINNY`-64 and `SKINNY`-128.

The security of `Deoxys-BC` has been extensively studied, particularly using rectangle, boomerang, and impossible differential attacks [5,10,31,37,51]. In [10], Cid *et al.* introduced a Mixed Integer Linear Programming (MILP) model to search for efficient rectangle distinguishers on `Deoxys-BC`, and found effective

attacks on 13 rounds of Deoxys-BC-384 and 10 rounds of Deoxys-BC-256. Subsequently, the *Boomerang Connectivity Table* (BCT) [11] and its variants, such as *Boomerang Difference Table* (BDT) [45], were introduced to better capture the dependency between the two differential trails in a boomerang distinguisher. Incorporating the BDT into the MILP model of [10], later works [48, 50] identified new boomerang distinguishers on Deoxys-BC, which worked more effectively for key recovery attacks. As a result, the rectangle attacks were extended to 14 rounds of Deoxys-BC-384 and 11 rounds of Deoxys-BC-256, respectively. Further improvements were achieved by using refined rectangle key-recovery algorithms, leading to reducing the time complexities of attacks on both versions [14, 44]. Most recently, Song *et al.* proposed a one-step framework that integrates the boomerang distinguisher and the key recovery phases into a single optimization model [43]. This unified approach enabled the rectangle attack on 15-round Deoxys-BC-384, leaving with only one round of security margin.

Similarly, SKINNY has also been subject to extensive cryptanalysis [14, 19, 20, 41]. In the single-key setting, the integral and MITM attacks are the most effective, while the rectangle attacks reach the largest number of attacked rounds in the related-tweakey setting so far. The first rectangle attacks on SKINNY were proposed in [28] that worked on 27 rounds. Delaune *et al.* [13] developed an MILP model capable of handling the dependencies in the middle rounds and hence obtained better boomerang distinguishers. After that, Dong *et al.* [14] combined this MILP model with a new rectangle key recovery algorithm and then built an MILP model that unifies the distinguisher search and the key recovery process, yielding rectangle attacks on 31-round SKINNY-64-192 and 32-round SKINNY-128-384. Later, Song *et al.* [44] generalized the rectangle key recovery algorithm, reducing the time complexities of some SKINNY variants.

Motivations. Firstly, we highlight the following features of the tweakey schedule, which have received little attention and, to our best knowledge, have not been exploited in prior attacks:

- Deoxys-BC and SKINNY-128 employ the same 8-bit LFSRs to update tweakey cells. These extremely lightweight LFSRs induce deterministic dependencies between subtweakeys separated by specific round offsets.
- SKINNY-64 uses 4-bit LFSRs, whose period causes subtweakeys to reappear after 30 rounds.

We refer to these dependencies as *Strong Key Bridges*.

Secondly, with respect to automatic search models for rectangle attacks, the existing approaches [14, 43] unify the rectangle distinguisher and key recovery into a single framework to enjoy global optimality. These models also incorporated the key guessing strategy. However, these models neglect the time required for extracting unguessed key bits (typically denoted by ϵ), assuming it to be negligible. In practice, ϵ can be significant (*e.g.*, 2^{24} as in [43]), and ignoring it may result in overlooking optimal attacks.

Thirdly, the state-test technique [9], which allows guessing selected internal state cells instead of a large set of key bits and has proven very effective in

rectangle attacks [43], has not yet been integrated into existing automatic search models.

Our contributions. We conduct a deeper scrutiny of the tweakey schedule of `Deoxys-BC` and `SKINNY`. For schedules using 8-bit LFSRs, although the LFSR period is 30, we identify strong dependencies between subtweakeys separated by 15 rounds; when such correlated subtweakeys are involved, these dependencies can be exploited to accelerate the key recovery attack. For 4-bit LFSRs used in `SKINNY-64`, subtweakeys literally reappear after 30 rounds, a phenomenon that has been largely overlooked. We also observe structural properties of the cell-position permutation for `Deoxys-BC-384`: one can obtain multiple 5-round differential patterns with only 5 active S-boxes and 6-round differential patterns with 10 active S-boxes in the related-tweakey setting. Such short differential patterns with few active S-boxes are especially suitable as building blocks for long boomerang distinguishers and provide useful heuristics when searching for overall attacks.

On the tooling side, we develop a full-fledged automatic Constraint Programming (CP) model to search for rectangle attacks. Our CP model treats the distinguisher and the key-recovery phase jointly and permits arbitrary key-guessing strategies (*i.e.*, any number of key bits may be pre-guessed). It goes beyond prior work in three major ways:

- It incorporates the **state-test** technique and allows a state cell to be tested either together with the pre-guessed tweakey cells or in the step of processing quartets and extracting the unguessed tweakey.
- For the first time, the model **automatically computes** ϵ–the concrete time cost of processing quartets and extracting unguessed key information–supporting both the guess-and-filter and precomputation (hash table) approaches. This removes the usual ad-hoc assumption that ϵ is negligible and enables true single-step optimization for concrete attack cost.
- By extending classical key-bridging ideas, the model incorporates the **strong key bridges** identified above, which in some cases yield substantial improvements to rectangle attacks.

By combining the CP model with the key-schedule properties, we significantly advance the state-of-the-art attacks against `Deoxys` and `SKINNY`. Compared to the previous best rectangle attacks on `Deoxys-BC-384` and `Deoxys-BC-256`, we reduce the time complexity by a factor of 2^{40} and 2^{32}, and also lower the data and memory requirements for `Deoxys-BC-256`, achieving the best attacks to date. In addition, we extend the best-known attacks on Deoxys instances, `Deoxys-I-256` and `Deoxys-I-128`, by one round. For `SKINNY`, our results improve the prior best attacks by one round for `SKINNY-128-384` and two rounds for `SKINNY-64-192`.

Finally, we note that if the linear operations in the tweakey schedule are implemented with the original finite-field multiplications, the strong key bridges arising from the 8-bit LFSRs disappear. Moreover, the model for explicit ϵ computation is not limited to rectangle attacks and can be applied to future one-step automatic search models for other differential-style attacks. We provide

the source code for our model and attacks at https://github.com/Icsnow/full-fledged-model.

Table 1: Summary of the cryptanalytic results. Rect./ID/Int.= Rectangle/ impossible differential/Integral attack. RTK/STK=related-tweakey/single tweakey.

Cipher	#R/#F	Data	Mem.	Time	Att.	Setting	$\|K\|$	Ref.
Deoxys-BC-384	15/16	$2^{115.7}$	2^{128}	$2^{371.7}$	Rect.	RTK	384	[43]
	15/16	$2^{115.7}$	2^{128}	$\mathbf{2^{297.7}}$	Rect.	RTK	320	Sect. 5.2
	15/16	$2^{115.7}$	2^{128}	$\mathbf{2^{331.7}}$	Rect.	RTK	384	Sect. 5.2
Deoxys-BC-256	11/14	$2^{126.78}$	2^{128}	$2^{222.49}$	Rect.	RTK	256	[47]
	11/14	$\mathbf{2^{119}}$	$\mathbf{2^{119}}$	$\mathbf{2^{189.8}}$	Rect.	RTK	256	Sect. C.1
Deoxys-I-256	13/16	$2^{125.2}$	2^{136}	$2^{186.7}$	Rect.	RTK	256	[49]
	14/16	$2^{115.1}$	2^{121}	$2^{235.1}$	Rect.	RTK	256	Sect. C.4
Deoxys-I-128	9/14	2^{117}	2^{117}	2^{118}	Rect.	RTK	128	[10]
	10/14	2^{91}	2^{91}	$2^{121.8}$	Rect.	RTK	128	Sect. C.4
SKINNY-128-384	26/56	2^{122}	2^{328}	2^{331}	Int.	STK	360	[19]
	27/56	$2^{124.99}$	2^{344}	$2^{362.61}$	ID	RTK	384	[20]
	32/56	$2^{123.54}$	$2^{129.54}$	$2^{344.78}$	Rect.	RTK	384	[47]
	33/56	$2^{123.54}$	$2^{123.54}$	$2^{359.54}$	Rect.	RTK	384	Sect. C.3
SKINNY-64-192	26/40	2^{62}	2^{164}	2^{166}	Int.	STK	180	[19]
	27/40	$2^{63.64}$	2^{172}	$2^{183.26}$	ID	RTK	192	[20]
	31/40	$2^{62.78}$	$2^{62.79}$	$2^{182.07}$	Rect.	RTK	192	[14]
	31/40	$2^{62.78}$	$2^{62.79}$	$\mathbf{2^{154.07}}$	Rect.	RTK	192	Sect. C.2
	32/40	$2^{62.78}$	$2^{62.79}$	$\mathbf{2^{158.07}}$	Rect.	RTK	192	Sect. C.2
	33/40	$2^{63.78}$	$2^{64.78}$	$2^{175.56}$	Rect.	RTK	192	Sect. C.2

2 Preliminaries

2.1 The STK Framework

The TWEAKEY framework [24] builds an n-bit tweakable block cipher with variable-size key K and tweak T. The concatenation of T and K, $T||K$, is called the tweakey state TK of tk bits. The tweakable block cipher is composed of r consecutive rounds, and each round consists of three parts: a subtweakey extraction function g from the tweakey state, an internal state update function f, and a tweakey state update function u.

Jean *et al.* identify a subclass of TWEAKEY, called the Superposition Tweakey (STK) framework [24], for handling large-size tweakey. In the STK framework, the n-bit internal state is split into n/c c-bit cells. The master tweakey is divided into p n/c square arrays of c-bit cells, where $p = tk/n$, as $(TK^1, \cdots, TK^p)$. Following the STK framework, several tweakable block ciphers have been proposed, including Deoxys-BC and SKINNY. As shown in Figure 1, the functions g, f and u for Deoxys and SKINNY are defined as follows:

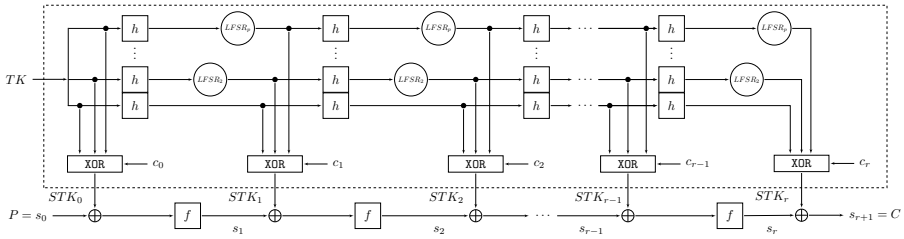

Figure 1: The `STK` framework [24].

- the function g is a simple XOR operation with all the p n-bit words of the tweakey state and the internal state, *i.e.*, $g(s_i, STK_i) = s_i \oplus STK_i$. For `SKINNY`, $STK_i = \bigoplus_j TK_i^j$, while for `Deoxys-BC`, $STK_i = \bigoplus_j TK_i^j \oplus c_i, 0 \leq i \leq r, 1 \leq j \leq p$, where c_i are the key schedule round constants. When $i = 0$, $TK_0^j = TK^j$, $TK = TK^1||TK^2||\cdots||TK^p$, $0 < j \leq p$.
- the function f is the round function of `Deoxys-BC` (resp. `SKINNY`).
- the function u consists of two operations: the cell position permutation h applied to each of the p n-bit words of the tweakey state, and Linear Feedback Shift Registers ($LFSR$s) applied to each cell within each of the last $(p-1)$ n-bit words of the tweakey state, *i.e.*, $TK_i^1 = h(TK_{i-1}^1), TK_i^j = h(LFSR_j(TK_{i-1}^j)), 2 \leq j \leq p$.

2.2 `Deoxys-BC` and `SKINNY`

Specification of `Deoxys-BC`. `Deoxys-BC` adopts the `AES` round function and the `STK` framework [24]. It encrypts a 128-bit plaintext P with the tweakey TK. The `Deoxys-BC` has two versions, denoted as `Deoxys-BC-256` and `Deoxys-BC-384` for $tk = 256$ and $tk = 384$, respectively. The number r of rounds is 14 and 16 for the two versions, respectively. The ordering of the state cells is shown in Figure 2. Each round applies four transformations sequentially to update the state.

- `AddRoundTweakey` (`ART`): XOR the 128-bit round subtweakey STK_i to the state W_{i-1}.
- `SubBytes` (`SB`): Apply the 8-bit `AES` S-box to each byte of the state X_i.
- `ShiftRows` (`SR`): Rotate the j-th row by j bytes to the left, $j = 0, 1, 2, 3$.
- `MixColumns` (`MC`): Multiply the state Z_i by a 4×4 Maximum Distance Separable (MDS) matrix of `AES`.

After the last round, a final `AddRoundTweakey` operation is performed to produce the ciphertext C. We denote the i-th round internal state as follows:

$$W_{i-1} \xrightarrow[STK_i]{\texttt{ART}} X_i \xrightarrow{\texttt{SB}} Y_i \xrightarrow{\texttt{SR}} Z_i \xrightarrow{\texttt{MC}} W_i.$$

Tweakey Schedule. The cell position permutation h is defined as $[1, 6, 11, 12, 5, 10, 15, 0, 9, 14, 3, 4, 13, 2, 7, 8]$. The $LFSR_2$ and $LFSR_3$ used in `Deoxys-BC`

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

Figure 2: Cell ordering in `Deoxys-BC`

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

Figure 3: Cell ordering in `SKINNY`

Table 2: The $LFSR$s used in `Deoxys-BC` and `SKINNY`-128 tweakey schedule.

$LFSR_2$	$(x_7 \|\|x_6\|\| x_5 \|\|x_4\|\| x_3 \|\|x_2\|\| x_1 \|\|x_0) \rightarrow (x_6 \|\|x_5\|\| x_4 \|\|x_3\|\| x_2 \|\|x_1\|\| x_0 \|\|x_7 \oplus x_5)$
$LFSR_3$	$(x_7 \|\|x_6\|\| x_5 \|\|x_4\|\| x_3 \|\|x_2\|\| x_1 \|\|x_0) \rightarrow (x_0 \oplus x_6 \|\|x_7\|\| x_6 \|\|x_5\|\| x_4 \|\|x_3\|\| x_2 \|\|x_1)$

are shown in Table 2. It should be noted that in [24], the designers originally employed the multiplication operation by coefficients 2^i $(i = 1, 2)$ over $GF(2^8)$ but changed to the current $LFSR$s for efficiency.

Specification of `SKINNY`. The `SKINNY` family [6] has 6 distinct versions, denoted as `SKINNY`-n-t, where $n \in \{64, 128\}$ represents the block size, and $t \in \{n, 2n, 3n\}$ indicates the tweakey size. The ordering of the state cells is shown in Figure 3. Each round of `SKINNY` has five basic operations:

- `SubCells` (`SC`): Apply 4-bit (resp. 8-bit) S-box to each cell of the state X_i, when n is 64 (resp. 128).
- `AddConstants` (`AC`): XOR the round constant to the state Y_i.
- `AddRoundTweakey` (`ART`): XOR the first two rows of the state Z_i with the first two rows of the round subtweakey STK_i.
- `ShiftRows` (`SR`): Rotate each cell in row j to the right by j cells, $j = 0, 1, 2, 3$.
- `MixColumns` (`MC`): Multiply the state W_i by a binary matrix M. M and its inverse M^{-1} are:

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad M^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

We denote the internal state in the i-th round as follows:

$$X_i \xrightarrow{\texttt{SC}} Y_i \xrightarrow[STK_i]{\texttt{ART}\circ\texttt{AC}} Z_i \xrightarrow{\texttt{SR}} W_i \xrightarrow{\texttt{MC}} X_{i+1}.$$

Tweakey Schedule. The cell position permutation h is defined as $[9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. When the block size n is 128 (resp. 64), the $LFSR$s that apply in the first two rows are given in Table 2 (resp. Table 3).

Table 3: The $LFSR$s used in SKINNY tweakey schedule, when the block size n is 64.

$LFSR_2$	$(x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_2\|\|x_1\|\|x_0\|\|x_3 \oplus x_2)$
$LFSR_3$	$(x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_0 \oplus x_3\|\|x_3\|\|x_2\|\|x_1)$

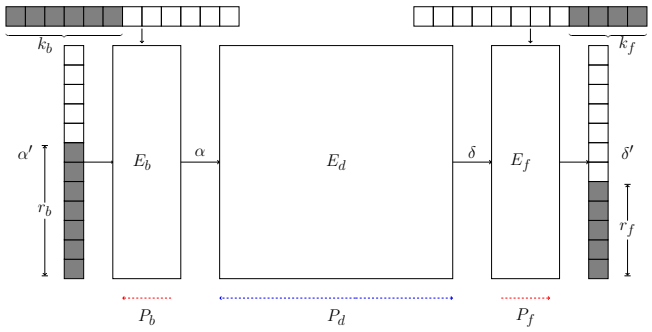

Figure 4: A high-level description of the generic rectangle key recovery attack.

2.3 Generic Rectangle Key Recovery Attack

Since the rectangle attack [7] was proposed, research on the rectangle attack for key recovery has been extensively studied in the literature [7,8,50]. A significant advancement was made by Song *et al.* at Asiacrypt 2022 [44,47], which proposed a unified framework for finding the best rectangle attack as follows.

As shown in Figure 4, the block cipher E is divided into three parts: $E = E_f \circ E_d \circ E_b$. Assume the boomerang distinguisher over E_d holds with a probability $P_d = Pr[E_d^{-1}(E_d(x) \oplus \delta) \oplus E_d^{-1}(E_d(x \oplus \alpha) \oplus \delta) = \alpha] = P^2$.

The differential propagation in E_b (resp. E_f) with probability $P_b = 1$ (resp. $P_f = 1$) is represented as $\alpha' \leftarrow \alpha$ (resp. $\delta \rightarrow \delta'$). Let r_b (resp. r_f) denote the dimension of all possible α' (resp. δ'). It requires subkey information k_b (resp. k_f) to verify the difference α (resp. δ) from plaintext (resp. ciphertext) pairs. Let m_b and m_f denote the number of the bits in k_b and k_f, *i.e.*, $m_b = |k_b|, m_f = |k_f|$.

The generic key recovery attack allows attackers to guess some key bits in advance, thereby sieving the data more quickly. Suppose we guess a part of k_b and k_f, denoted by k'_b and k'_f, at first. Let $m'_b = |k'_b|$ and $m'_f = |k'_f|$, where $0 \leq |m'_b| \leq |m_b|$, $0 \leq |m'_f| \leq |m_f|$. Using the guessed subkey bits, we can get an r'_b-bit condition in the E_b component and an r'_f-bit condition in the E_f component. Additionally, let $r^*_b = r_b - r'_b$ and $r^*_f = r_f - r'_f$. With these parameters, we give a detailed description of the generic classical rectangle attack adapted to the related-tweakey setting where the tweakey schedule is linear.

1. **Data collection.** Collect and store y structures of 2^{r_b} plaintexts. Query the corresponding ciphertexts for each structure under the four related keys

K_1, K_2, K_3, K_4. The time complexity of this step is $T_0 = D_R = 4 \cdot D = 4 \cdot y \cdot 2^{r_b} = \sqrt{s}2^{n/2+2}/P$, where s is the expected number of right quartets and $y = \sqrt{s}2^{n/2-r_b}/P$.

2. **Key candidates extraction.**
   - **Partial encryption and partial decryption.** For each (P_i, C_i), partially encrypt P_i and partially decrypt C_i with the pre-guessed subkey bits k'_b and k'_f of K_i and get P_i^* and C_i^*, where $i \in \{1, 2, 3, 4\}$. The time complexity of this step is $T_1 = 2^{|k'_b \cup k'_f|} \cdot D_R$.
   - **Pair construction.** Insert all the obtained (P_i^*, C_i^*) into a hash table indexed by the inactive bits of P_i^* or C_i^* to construct two sets of pairs $S_1 = \{(P_1^*, C_1^*, P_2^*, C_2^*)\}$, $S_2 = \{(P_3^*, C_3^*, P_4^*, C_4^*)\}$ or $S'_1 = \{(P_1^*, C_1^*, P_3^*, C_3^*)\}$, $S'_2 = \{(P_2^*, C_2^*, P_4^*, C_4^*)\}$. The time complexity of this step is $T_2 = 2^{|k'_b \cup k'_f|} \cdot 2 \cdot D \cdot \min\{2^{r_b^*}, D \cdot 2^{r_f^* - n}\}$.
   - **Quartets generation and processing.** Insert S_1, S_2 or S'_1, S'_2 into a hash table by the inactive bits of C_1^*, C_2^* or P_1^*, P_3^*. Then, generate the quartet for each index. The number of quartets generated is $D^2 \cdot 2^{2r_b^* + 2r_f^* - 2n}$. Determine the key candidates involved in E_b and E_f, and increment the corresponding counters, whose time complexity is denoted as ϵ. The time complexity of this step is $T_3 = 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r_b^* + 2r_f^* - 2n} \cdot \epsilon$,
3. **Exhaustive searching.** According to the key schedule, guess the remaining unknown $(k - k_b - k_f)$-bit key information and exhaustively search over them to recover the correct key. The time complexity is $T_4 = 2^{k-\ell}$, where $\ell \leq q + |k_b \cup k_f| - |k'_b \cup k'_f|$ and $0 \leq q \leq |k'_b \cup k'_f|$ is the number of information bits extracted in Step 2.

The data complexity is D_R. The memory complexity is $M = D_R + 2 \cdot \min\{D \cdot 2^{r_b^*}, D^2 \cdot 2^{r_f^* - n}\} + 2^{q + |k_b \cup k_f| - |k'_b \cup k'_f|}$ for storing data, pairs, and key counters. The success probability of the attack can be estimated using P^2, s and ℓ [39].

3 Tweakey Schedule Properties of `Deoxys-BC` and `SKINNY`

This section identifies the strong key bridges between specific-round-distance subtweakeys of `Deoxys-BC` and `SKINNY` that stem from the LFSR operations. For `Deoxys-BC`, we further examine structural properties of the cell-wise permutation h and leverage them to accelerate the search for rectangle attacks on `Deoxys-BC-384`.

3.1 Strong Key Bridges from LFSRs

For both `Deoxys-BC` and `SKINNY`, the tweakey schedule applies the same permutation h to each tweakey word $TK^j, 0 < j \leq p$. Hence, there is no diffusion between tweakey cells. Moreover, the tweakey schedule applies an LFSR to update the cells of TK^2 and TK^3, and obtains subtweakey cells by XORing p tweakey words. Thus, each subtweakey cell $STK_r[i]$ for all rounds r is generated by only p tweakey cells. This enables to analyze the relation between subtweakeys through each group of p tweakey cells as follows.

(1) LFSRs for 8-bit cells. `Deoxys-BC` and `SKINNY-128` use the same LFSRs, which are given in Table 2 (x_0 stands for the LSB of the cell). As in [35], we represent LFSR_2 and LFSR_3 with two binary matrices $\boldsymbol{L_2}$ and $\boldsymbol{L_3}$ as follows:

$$\boldsymbol{L_2} = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}, \quad \boldsymbol{L_3} = \begin{pmatrix} 0&1&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0 \end{pmatrix}.$$

Each cell $TK^j[i]$ $(1 \le j \le 3, 0 \le i \le 15)$ is represented as a binary vector $\boldsymbol{tk}^j[i] = (tk_7^j[i], \cdots, tk_0^j[i])^\top$.

The independence of subtweakeys does not depend on constants. Therefore, we assume that the i_r-th byte in the r-th subtweakey is

$$STK_r[i_r] = TK^1[i] \oplus \mathrm{LFSR}_2^r(TK^2[i]) \oplus \mathrm{LFSR}_3^r(TK^3[i]),$$

where $i_r = h^{-r}[i]$, $0 \le i \le 15$. Then we obtain

$$STK_r[i_r] = \left[\boldsymbol{I}\ \boldsymbol{L}_2^r\ \boldsymbol{L}_3^r\right] \cdot (\boldsymbol{tk}^1[i], \boldsymbol{tk}^2[i], \boldsymbol{tk}^3[i])^\top.$$

For instance, when $r = 0$, $STK_0[i] = \left[\boldsymbol{I}\ \boldsymbol{I}\ \boldsymbol{I}\right] \cdot (\boldsymbol{tk}^1[i], \boldsymbol{tk}^2[i], \boldsymbol{tk}^3[i])^\top$.

Since the orders of the matrices $\boldsymbol{L_2}$ and $\boldsymbol{L_3}$ are 30, that is, $\boldsymbol{L_2^{30}} = \boldsymbol{L_3^{30}} = \boldsymbol{I}$, the tweakey state values in the 30-th round coincide with those in the 0-th round. In particular, $\boldsymbol{L_3}$ is the inverse of $\boldsymbol{L_2}$, *i.e.*, $\boldsymbol{L_2} \cdot \boldsymbol{L_3} = \boldsymbol{I}$. Beyond this inverse relationship, strong key bridges also exist across intermediate rounds, as stated in the following propositions. We refer to the code repository in Section 1 for computing the number of information bits.

Proposition 1. *For the i_m-th byte of the m-th subtweakey $STK_m[i_m]$, the information bits in $STK_0[i] \cup STK_m[i_m]$ exhibit the following properties, where $1 \le m \le 30$, $0 \le i \le 15$ and $i_m = h^{-m}[i]$:*

- *when $m = 15$, $|STK_0[i] \cup STK_{15}[i_{15}]| = 12$;*
- *when $m = 30$, $|STK_0[i] \cup STK_{30}[i_{30}]| = 8$;*
- *in other cases, $|STK_0[i] \cup STK_m[i_m]| = 16$.*

Proof. As $STK_m[i_m] = \left[\boldsymbol{I}\ \boldsymbol{L}_2^m\ \boldsymbol{L}_3^m\right] \cdot (\boldsymbol{tk}^1[i], \boldsymbol{tk}^2[i], \boldsymbol{tk}^3[i])^\top$ according to the tweakey schedule, the number of information bits in $STK_0[i] \cup STK_m[i_m]$ equals the rank of the matrix $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{L}_2^m & \boldsymbol{L}_3^m \end{bmatrix}$. As $\boldsymbol{L}_3 = \boldsymbol{L}_2^{-1}$ and $\boldsymbol{L}_2^{30} = \boldsymbol{I}$, we have

$$[\boldsymbol{I}\ \boldsymbol{L}_2^m\ \boldsymbol{L}_3^m] = [\boldsymbol{I}\ \boldsymbol{L}_2^m\ \boldsymbol{L}_2^{-m}] = [\boldsymbol{I}\ \boldsymbol{L}_2^m\ \boldsymbol{L}_2^{30-m}].$$

Then it is deduced that

– when $m = 15$,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L_2^{15}} + \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}\right).$$

Since $\texttt{Rank}(\boldsymbol{L_2^{15}} + I) = 4$, which can be verified by codes, we have $\texttt{Rank}(\boldsymbol{C}) = 12$ and $|STK_0[j] \cup STK_{15}[j_{15}]| = 12$. The dependency is illustrated with concrete linear relations in Equations 1 and 2.

– When $m = 30$,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right),$$

we have $|STK_0[j] \cup STK_{30}[j_{30}]| = 8$.

– When m is in other cases,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L_2^m} + \boldsymbol{I} & \boldsymbol{L_2^{-m}} + \boldsymbol{I} \end{bmatrix}\right).$$

Since $\texttt{Rank}(\boldsymbol{L_2^m} + \boldsymbol{I}) = \texttt{Rank}(\boldsymbol{L_2^{-m}} + \boldsymbol{I}) = 8$, we have $\texttt{Rank}(\boldsymbol{C}) = 16$ and $|STK_0[i] \cup STK_m[i_m]| = 16$.

□

Analogously, the case involving three subtweakeys is summarized in Proposition 2, whose proof is provided in Appendix A.2.

Proposition 2. *For the subtweakey bytes $STK_m[i_m]$ and $STK_n[i_n]$, the information bits in $STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]$ exhibit the following properties, where $0 < m < n \leq 30$, $0 \leq i \leq 15$, $i_m = h^{-m}[i]$, and $i_n = h^{-n}[i]$:*

– *when $n \neq 30$ and one of m, n or $n - m$ is 15, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 20$;*
– *when $n = 30$ and $m = 15$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 12$;*
– *when $n = 30$ and $m \neq 15$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 16$;*
– *when (n, m) is in other cases, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 24$.*

If TK^1 is set as the tweak, the dependence of the subtweakey cells becomes irrelevant to TK^1. Without loss of generality, TK^1 can be considered a constant, and $STK_r[i_r]$ is treated as

$$STK_r[i_r] = \text{LFSR}_2^r(TK^2[i]) \oplus \text{LFSR}_3^r(TK^3[i]).$$

Similarly to Proposition 1, the number of information bits in this case is given by the rank of the matrix $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{L_2^r} & \boldsymbol{L_3^r} \end{bmatrix}$, that is $|STK_0[i] \cup STK_r[i_r]| = \texttt{Rank}(\boldsymbol{C})$. Based on the above, we derive the following proposition.

Proposition 3. *For the i_m-th byte of the m-th subtweakey $STK_m[i_m]$, the information bits in $STK_0[i] \cup STK_m[i_m]$ exhibit the following properties, where $1 \leq m \leq 30$, $0 \leq i \leq 15$ and $i_m = h^{-1}[i]$:*

– *when $m = 15$ or $m = 30$, $|STK_0[i] \cup STK_m[i_m]| = 8$ (see Equation 3);*

– *when m is in other cases,* $|STK_0[i] \cup STK_m[i_m]| = 16$.

It is significant to note that when using the multiplication operation by coefficients 2^i $(i = 1, 2)$ in $GF(2^8)$, as prescribed in [24], the update function for 8-bit cells, which has the maximal period, does not exhibit the dependency between cells separated by 15 rounds. This implies that the properties delineated in Propositions 1 through 3 for $m = 15$ or $n - m = 15$ do not apply.

(2) LFSRs for 4-bit cells. Similarly to `SKINNY`-128, the `SKINNY`-64 cipher also employs two 4-bit LFSRs, as specified in Table 3. Given that the orders of these LFSRs are 15, it follows that `SKINNY`-64 exhibits properties similar to those previously discussed.

Proposition 4. *For the subtweakey cell $STK_m[i_m]$, the information bits in $STK_0[i] \cup STK_m[i_m]$ exhibit the following properties, where $1 \leq m \leq 15$, $0 \leq i \leq 15$ and $i_m = h^{-m}[i]$:*

– *when $m = 15$, $|STK_0[i] \cup STK_m[i_m]| = 4$;*
– *when m is in other cases, $|STK_0[i] \cup STK_m[i_m]| = 8$.*

Proposition 5. *For subtweakey cells $STK_m[i_m]$ and $STK_n[i_n]$, the information bits in $STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]$ exhibit the following properties, where $0 < m < n \leq 15$, $0 \leq i \leq 15$, $i_m = h^{-m}[i]$, and $i_n = h^{-n}[i]$:*

– *when $n = 15$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 8$;*
– *when (m, n) is in other cases, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 12$.*

The properties established in the preceding propositions for 8-bit and 4-bit LFSRs can be generalized to systems with an arbitrary number of subtweakeys. This leads to the following corollary.

Corollary 1. *Suppose the order of LFSRs is r, for the i_ℓ, i_m, i_n-th cells of the ℓ, m, n-th subtweakeys, where $i_x = h^{-x}[i], x \geq 0, 0 \leq i \leq 15$, the information bits satisfy the properties:*

- *when $1 \leq m < n \leq r$, $|STK_m[i_m] \cup STK_n[i_n]| = |STK_0[i] \cup STK_{n-m}[i_{n-m}]|$;*
- *when $1 \leq \ell < m < n \leq r$, $|STK_\ell[i_\ell] \cup STK_m[i_m] \cup STK_n[i_n]| = |STK_0[i] \cup STK_{m-\ell}[i_{m-\ell}] \cup STK_{n-\ell}[i_{n-\ell}]|$.*

(3) One column of an equivalent subtweakey of `Deoxys-BC`. In `Deoxys-BC`, the equivalent subtweakey involved in E_f is used consistently to compute the key information bits. However, due to the application of $\texttt{MC}^{-1}$, each bit within a column depends on four bytes. Consequently, analysis can no longer be performed on a byte-wise basis; instead, it necessitates a shift to processing entire columns as fundamental units.

Table 4: The values of $|eqSTK_{15}[I_{15}] \cap STK_0[I_0]|$, $|I_0|$ is the number of bytes involved in the 0-th round, $|I_{15}|$ is the number of bytes involved in the 15-th round, x^* denotes the values is greater than or equal to x.

		$\|I_{15}\|$			
		1	2	3	4
$\|I_0\|$	1	0	0	0	4
	2	0	0	2^*	8
	3	0	1^*	6	12
	4	0	4^*	10	16

Without loss of generality, we let $eqSTK_r = \mathtt{MC}^{-1}(STK_r)$, that is

$$\begin{pmatrix} eqSTK_r[i] \\ eqSTK_r[i+1] \\ eqSTK_r[i+2] \\ eqSTK_r[i+3] \end{pmatrix} = \mathtt{MC}^{-1} \cdot \begin{pmatrix} STK_r[i] \\ STK_r[i+1] \\ STK_r[i+2] \\ STK_r[i+3] \end{pmatrix}$$

where $r \geq 0$, $i \in \{0, 4, 8, 12\}$. We analyze the relationship between the r-th and $(r+15)$-th rounds. The dependence of subtweakey cells in more general cases is evaluated via a concrete rank computation.

Proposition 6. *For $\forall I_{15} \subseteq \{i, i+1, i+2, i+3\}, i \in \{0, 4, 8, 12\}$ of the subtweakey $eqSTK_{15}[I_{15}]$ and $I_0 \subseteq \{h^{15}[i], h^{15}[i+1], h^{15}[i+2], h^{15}[i+3]\}$ of the subtweakey $STK_0[I_0]$, the information bits satisfy*

$$|eqSTK_{15}[I_{15}] \cup STK_0[I_0]| = 8 \times (|I_0| + |I_{15}|) - |eqSTK_{15}[I_{15}] \cap STK_0[I_0]|,$$

where the values of $|eqSTK_{15}[I_{15}] \cap STK_0[I_0]|$ are shown in Table 4.

As an example, for $|I_{15}| = 4$, the relation $|eqSTK_{15}[I_{15}] \cap STK_0[I_0]| = 4 \times |I_0|$ can be verified, matching the outcome of a byte-wise-dependence calculation. It can be concluded that this property applies to all subtweakey pairs with a 15-round interval, thus justifying the corollary below.

Corollary 2. *For the m-th subtweakey, $m \geq 0$, the information bits satisfy the property $|eqSTK_{15+m}[I_{15+m}] \cup STK_m[I_m]| = |eqSTK_{15}[I_{15}] \cup STK_0[I_0]|$.*

It should be noted that the key relation between closely located subtweakey cells, which depends on the LFSRs, has been reported as key cancellation in the distinguisher [35]. The strong key bridges in our paper correspond to the undiscovered specific-round-distance key relationship, generalize the key relations, and enable their use in the key recovery phase.

3.2 Property of h in Deoxys-BC

In the following, we use $\{e_{0,0}, \cdots e_{3,3}\}$ to denote the unit vectors in $\mathbb{F}_{2^8}^{4\times 4}$, where $e_{i,j}$ has a single entry equal to 1 in row i and column j, and 0 elsewhere.

Definition 1. *The column spaces $\mathcal{C}_i$, where $i \in \{0,1,2,3\}$, are defined as*

$$\mathcal{C}_i = \langle e_{0,i}, e_{1,i}, e_{2,i}, e_{3,i} \rangle .$$

For instance, $\mathcal{C}_0$ corresponds to the symbolic matrix

$$\mathcal{C}_0 = \left\{ \begin{bmatrix} x_0 & 0 & 0 & 0 \\ x_1 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 \end{bmatrix} \middle| \forall x_0, x_1, x_2, x_3 \in \mathbb{F}_{2^8} \right\} \equiv \begin{bmatrix} x_0 & 0 & 0 & 0 \\ x_1 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 \end{bmatrix} .$$

Definition 2. *The diagonal spaces $\mathcal{D}_i$ are defined as*

$$\mathcal{D}_i = \mathtt{SR}^{-1}(\mathcal{C}_i) = \left\langle e_{0,i}, e_{1,(i+1 \bmod 4)}, e_{2,(i+2 \bmod 4)}, e_{3,(i+3 \bmod 4)} \right\rangle .$$

Similarly, the inverse-diagonal spaces $\mathcal{ID}_i$ are defined as

$$\mathcal{ID}_i = \mathtt{SR}(\mathcal{C}_i) = \left\langle e_{0,i}, e_{1,(i-1 \bmod 4)}, e_{2,(i-2 \bmod 4)}, e_{3,(i-3 \bmod 4)} \right\rangle .$$

For instance, $\mathcal{D}_0$ and $\mathcal{ID}_0$ corresponds to the symbolic matrix

$$\mathcal{D}_0 = \begin{bmatrix} x_0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 0 \\ 0 & 0 & x_2 & 0 \\ 0 & 0 & 0 & x_3 \end{bmatrix}, \mathcal{ID}_0 = \begin{bmatrix} x_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 \\ 0 & 0 & x_2 & 0 \\ 0 & x_3 & 0 & 0 \end{bmatrix}$$

for all $x_0, x_1, x_2, x_3 \in \mathbb{F}_{2^8}$.

For the permutation h and the round function R used in `Deoxys-BC`, we establish the following properties.

Property 1. The order of h is 8, and for $\forall i \in \{0,1,2,3\}$, $h(\mathcal{C}_i) = \mathcal{ID}_i, h^3(\mathcal{C}_i) = \mathcal{D}_{(i+1 \bmod 4)}, h^4(\mathcal{C}_i) = \mathcal{C}_{(i+2 \bmod 4)}$ and $h^8(\mathcal{C}_i) = \mathcal{C}_i$.

Property 2. Let R be the round function of `Deoxys-BC`, then for $\forall i \in \{0,1,2,3\}$, $R(\mathcal{D}_i) = \mathcal{C}_i$.

For `Deoxys-BC`-384, by exploiting the properties of h and the round function, we can identify differential distinguisher patterns with a minimal number of active S-boxes. These patterns subsequently serve as a basis for constructing rectangle distinguishers and mounting attacks. The details are presented below.

Proposition 7. *Let $\mathcal{O}$ be the all-zero state. Given that $\Delta STK_r \subseteq \mathcal{C}_i$, $\Delta STK_{r+1} = \mathcal{O}$ and $\Delta STK_{r+2} = \mathcal{O}$, for all $i \in \{0,1,2,3\}$, the minimum number of active S-boxes in the 5-round distinguisher is 5, and the minimum number of active S-boxes in the 6-round distinguisher is 10.*

Proof. Let the difference of the input state in round r, denoted as $\Delta W_{r-1} = \Delta STK_r$, be shown in Figure 5.

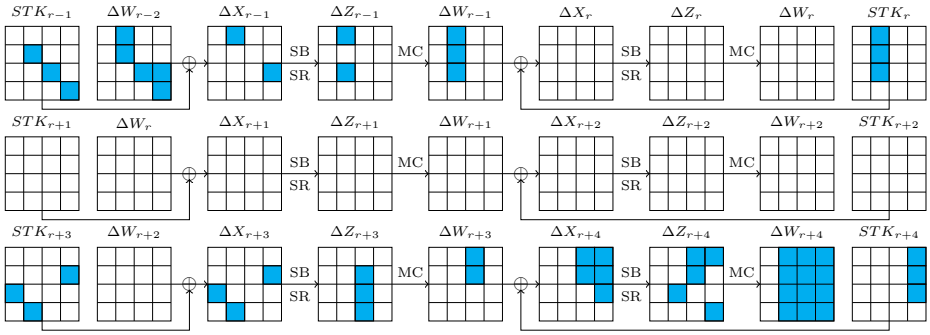

Figure 5: The 5-round and 6-round distinguishers of `Deoxys-BC-384`

For the top 5-round distinguisher in the Figure 5, it is that $\Delta X_r = \Delta X_{r+1} = \Delta X_{r+2} = \mathcal{O}$. For the $(r+3)$-th round, we can obtain

$$\Delta X_{r+3} = \Delta STK_{r+3} \in h^3(\mathcal{C}_i) = \mathcal{D}_{(i+1 \bmod 4)}.$$

Suppose that there are x active bytes in ΔSTK_r. Based on the property that the branch number of the $\mathtt{MC}^{-1}$ is 5, we can derive that there are at least $5-x$ active bytes in ΔZ_{r-1} and x active bytes in ΔX_{r+3}. Thus, for the 5-round distinguisher in this pattern, there are 5 active S-boxes in total.

When adding one round after the 5-round distinguisher, we get that

$$\begin{aligned}\Delta X_{r+4} &= \Delta W_{r+3} \oplus \Delta STK_{r+4} \in R(h^{-3}(\mathcal{C}_i)) \oplus h^{-4}(\mathcal{C}_i) \\ &= R(\mathcal{D}_{(i+1 \bmod 4)}) \oplus \mathcal{C}_{(i+2 \bmod 4)} = \mathcal{C}_I\end{aligned}$$

$I = \{i+1 \bmod 4, i+2 \bmod 4\}$. There are $5-x$ active bytes in ΔW_{r+3} at least and x active bytes in ΔSTK_{r+4}. Thus, for the 6-round distinguisher in this case, there are 10 active S-boxes in total. □

Remark. For `Deoxys-BC-384`, this differential pattern is particularly advantageous for rectangle attacks. On the one hand, the rectangle distinguisher is formed by connecting two short differential distinguishers. On the other hand, the property of h facilitates locating a fully inactive column during both upward and downward propagation, thereby enhancing the key recovery phase.

4 A Full-fledged Model for Searching Rectangle Attacks

Finding the optimal rectangle attack efficiently and accurately remains an unresolved issue. A full rectangle attack consists of the following components if probabilistic extensions are allowed in the outer part.

Distinguisher. A boomerang distinguisher with probability $P_d > 2^{-n}$ can serve as the core component and starting point for a rectangle attack.

Probabilistic extension. Around the distinguisher, the differential can be extended probabilistically from inside to outside over E_b^{-1} and E_f, respectively,

with probability $P_b \leq 1$ and $P_f \leq 1$, as in [4, 43]. Thus, the overall probability of the rectangle attack is $P^2 = P_b P_d P_f$, as shown in Figure 4. The concrete extensions determine k_b and k_f, which are involved in the key recovery attack, as well as the boundaries between the distinguisher and the extensions.

State test. Instead of solely guessing key cells, guessing critical internal state cells may reduce the overall size of guessed key/state cells, potentially bringing a reduction in time complexity. This technique was initially proposed for impossible differential attacks [9]. A slightly different version was reinvented for rectangle attacks in [43]. The first automatic MILP-based model was provided to enhance differential MITM attacks [4]. Later, in [30], the state test technique was extended to apply in impossible-differential, (truncated) differential, MITM, differential, and differential-linear cryptanalysis.

Key guessing strategy. According to the holistic key guessing strategy from [44], guessing a proper subset of key cells $k_b' \subset k_b$, $k_f' \subset k_f$ in advance can lead to relatively balanced time complexities of the stages of the attack, thus optimizing the overall time complexity.

ϵ calculation. In the generic key recovery algorithm [44], the time complexity for extracting the remaining key bits of k_b and k_f are denoted by ϵ, which is estimated manually in previous works.

Key bridging. Exploit the relation between subkey cells according to the key schedule and find independent subkey bits in $k_b \cup k_f$ and $k_b' \cup k_f'$.

In this section, we present a systematic and full-fledged Constraint Programming (CP) model for rectangle attacks, which integrates all the techniques and components above. While a modelling method for the state-test technique was introduced in [4] in the context of truncated differential MITM attacks, no such method has yet been proposed in any existing model dedicated to rectangle attacks. It is worth recalling that the key recovery in rectangle attacks is considerably more intricate. Specifically, in the model for rectangle attacks, a state cell may be guessed alongside pre-guessed key bits or during the phase of extracting the remaining key bits. Our model is able to accommodate this requirement. Furthermore, our model integrates the computation of ϵ for the first time, an inherent component of all differential-style attacks. Without this component, the optimality of attacks cannot be guaranteed. In particular, our model supports two approaches for extracting remaining key bits: the guess-and-filter approach and the approach based on precomputed hash tables. Additionally, the strong key bridges specified in Section 3 are integrated into our model to maximize its practical utility. By incorporating all these state-of-the-art techniques, we propose our model and elaborate on three critical components in the subsequent subsections: the state test, the computation of ϵ, and the key bridging.

For clarity, we use three rounds of `Deoxys-BC` as a toy example to illustrate our model. Suppose these correspond to the final rounds of an attack and equivalent subtweakeys $\mathtt{EK} = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(\mathtt{STK})$ are employed instead. Accordingly, the state transitions for the r-th round ($r \in \{6, 7, 8\}$) are given by $\mathtt{X_r} \xrightarrow[\mathtt{AEK}]{\mathtt{SB}} \mathtt{Y_r} \xrightarrow{\mathtt{SR}} \mathtt{Z_r} \xrightarrow{\mathtt{MC}} \mathtt{X_{r+1}}$, as depicted in Figure 6, where `AEK` denotes the equiv-

Table 5: Main variables employed in our model, where V could be any of {X, Y, Z}

Variables	Domain	Representation	Coverage
dV dEK	{0,1,2} {0,1}	0: zero difference 1: determined non-zero difference 2: arbitrary difference	$\{\mathtt{rE_b}\} \cup \{\mathtt{rE_d}\} \cup \{\mathtt{rE_f}\}$
vV, vEK	{0,1}	0: value uninvolved in key recovery 1: value involved in key recovery	$\{\mathtt{rE_b}\} \cup \{\mathtt{rE_f}\}$
tV	{0,1,2}	0: untested state 1: tested state in key guessing strategy 2: tested state in ϵ calculation	
eV, eEK	{0,1,2}	0: uneliminated state/subkey 1: elimination in key guessing strategy 2: elimination in ϵ calculation	
gEK	{0,1}	0: unguessing subkey 1: pre-guessed subkey	
detV	{0,1}	0: undetermined value 1: determined value	
frSB, frMC	{0,1}	0: filter not obtained 1: obtained filter related to SB/MC	
sV, sEK	{-1,...,S}	-1: uninvolved in ϵ calculation 0: involved in key guessing strategy {1,...,S}: step in ϵ calculation	

alent subtweakey addition operation. The primary variables used in our model are summarized in Table 5; variables specific to the three critical components will be introduced in the respective sections.

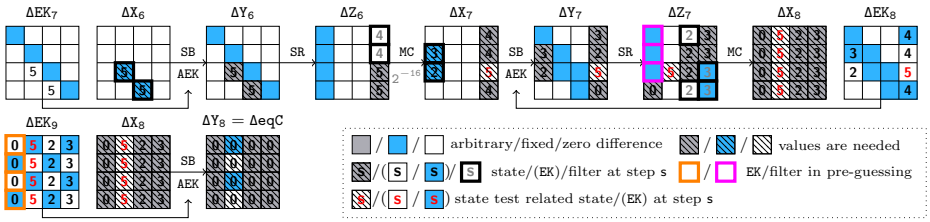

Figure 6: The toy example of E_f in the rectangle attack on Deoxys-BC

4.1 Model for the State Test

The state test is strategically integrated across two core components of our model: key guessing and the ϵ calculation. When activated, it eliminates some subkeys associated with states that have a determined differential. This approach is justified when the cost of processing the eliminated subkeys exceeds that of state testing itself. Let the size of a state be x bits, corresponding to e bits of eliminated subkey. In differential attacks (also in impossible differential and differential MITM attacks), testing a state with a determined difference costs 2^x, while

Algorithm 1: Model for the state test aligning with Figure 7

Input: variables in Table 5, the switch of the state test $\texttt{FlagST} \in \{0,1\}$.

1 $\forall^{\texttt{c}\in\{0,\dots,15\}}_{\texttt{r}\in\{6,7,8\}} \texttt{tX}^{\texttt{c}}_{\texttt{r}} = 0 \leftarrow \texttt{FlagST} = 0$

2 **for** $\texttt{r} \in \{6,7,8\}, \texttt{c} \in \{0,\dots,15\}$ **do**

3 *// pick up states that allow testing*

4 $\begin{cases} \texttt{tX}^{\texttt{c}}_{\texttt{r}} \in \{0,1,2\} \leftarrow \texttt{vX}^{\texttt{c}}_{\texttt{r}} = 1 \\ \texttt{eX}^{\texttt{c}}_{\texttt{r}} = \texttt{tX}^{\texttt{c}}_{\texttt{r}} = 0 \leftarrow \texttt{vX}^{\texttt{c}}_{\texttt{r}} = 0 \end{cases}$

5 *// elimination propagation in SB, AEK, SR*

6 $\begin{cases} \texttt{eZ}^{\texttt{SR(c)}}_{\texttt{r}} = \texttt{eY}^{\texttt{c}}_{\texttt{r}} = \texttt{eEK}^{\texttt{c}}_{\texttt{r}} = \texttt{eX}^{\texttt{c}}_{\texttt{r}} \leftarrow \texttt{dY}^{\texttt{c}}_{\texttt{r}} \leq 1 \\ \texttt{eZ}^{\texttt{SR(c)}}_{\texttt{r}} = \texttt{eY}^{\texttt{c}}_{\texttt{r}} = \texttt{eEK}^{\texttt{c}}_{\texttt{r}} = 0 \quad \leftarrow \texttt{dY}^{\texttt{c}}_{\texttt{r}} = 2 \end{cases}$

7 **for** $\texttt{r} \in \{6,7,8\}, \texttt{col} \in \{0,4,8,12\}, \texttt{i} \in \{0,\dots,3\}$ **do**

8 *// elimination propagation in MC*

9 $\texttt{eX}^{\texttt{col+i}}_{\texttt{r+1}} = \bigwedge_{\texttt{j}\in\{0,\dots,3\}} \left((\texttt{eZ}^{\texttt{col+j}}_{\texttt{r}} = 1) \vee (\texttt{vZ}^{\texttt{col+j}}_{\texttt{r}} = 0) \right)$

testing a state with an arbitrary difference costs 2^{2x}. In rectangle attacks, aligning differentials over four trails further doubles these costs. Therefore, applying the state test is advantageous only when $2^e > 2^{2x}$ for a determined difference, or $2^e > 2^{4x}$ for an arbitrary difference. Consequently, testing is restricted to states with a determined difference in our model.

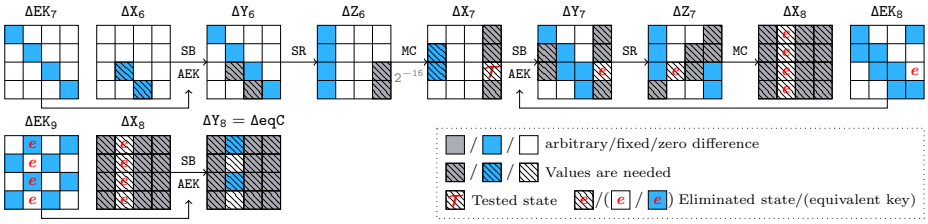

Figure 7: The toy example of the state test in the related-key model.

To clearly describe our model of the state test, we present Algorithm 1, which aligns with the example illustrated in Figure 7. Our algorithm accepts a Boolean variable FlagST as input. As shown in line 1, this variable acts as a switch that enables the state test only when $\texttt{FlagST} = 1$. In line 4, we restrict testing to only those states that are involved ($\texttt{vX} = 1$). Such states are allowed to be tested during the key-guessing strategy phase ($\texttt{tX} = 1$) or the ϵ calculation phase ($\texttt{tX} = 2$). Once a state is selected for testing, subsequent eliminations are triggered. The elimination propagation rule is detailed in lines $5 \sim 9$.

- For SB, as specified in line 6, to pruning searching space, the constraints require that only states corresponding to known differences can be eliminated.
- For MC, lines $7 \sim 9$ specify that all eliminated cells within a column after the MC must remain separate from other involved cells. This ensures that

the corresponding eliminated subkeys form a disjoint set. For example, in Figure 7, cells in the same column of $\mathtt{X}_8$ can be eliminated only if all involved cells in the second column of $\mathtt{Z}_7$ have been eliminated.

In Figure 7, testing $\mathtt{X}_7^{14}$ ($\mathtt{tX}_7^{14} \in \{1, 2\}$) leads to the elimination of states with known differences: $\mathtt{X}_7^{14}$ itself ($\mathtt{eX}_7^{14} = \mathtt{tX}_7^{14}$) and $\mathtt{X}_8^{\{4,5,6,7\}} \left(\bigwedge_{\mathtt{i}\in\{4,5,6,7\}} (\mathtt{eX}_8^{\mathtt{i}} = \mathtt{eX}_7^{14})\right)$, and the elimination of equivalent key cells $\mathtt{EK}_8^{14}$ ($\mathtt{eEK}_8^{14} = \mathtt{eX}_7^{14}$) and $\mathtt{EK}_9^{\{4,5,6,7\}}$ $\left(\bigwedge_{\mathtt{i}\in\{4,5,6,7\}} (\mathtt{eEK}_9^{\mathtt{i}} = \mathtt{eX}_8^{14})\right)$. It is advantageous to test a single state cell rather than guessing five key cells.

Model for Key Guessing Strategy Allowing the State Test. The core principle of the generic key guessing strategy is to permit partial guessing of the involved subkeys of $k_b \cup k_f$, thereby enabling early derivation of corresponding filters during pair and quartet generation. When the state test is employed in the key-guessing strategy, the eliminated subkeys reduce the number of subkeys required to guess. For the first time, we integrate the state test into the key-guessing strategy model, thereby significantly improving its efficiency and flexibility. The detailed model is given in Appendix B.1.

4.2 Model for the ϵ Calculation

The symbol ϵ represents the complexity of processing quartets and extracting the remaining key information after guessing some subkey bits. Previous works calculated ϵ manually, often assuming $\epsilon = 1$ in automatic models. However, this oversimplification may yield seemingly optimal but actually suboptimal results. When numerous subkeys remain unguessed, ϵ can significantly exceed 1, leading to unexpectedly high attack complexities. Rectangle attacks typically employ two ϵ calculation methods: the guess-and-filter method and the time-memory tradeoff using precomputation hash tables, both of which progressively reduce the remaining quartets by iteratively processing subkeys.

This subsection presents an automated model for computing ϵ, which enables the optimization of overall attack complexity without assumptions. The core challenge in developing such a model arises from the inherent sophistication of ϵ calculation in rectangle attacks. This calculation critically depends on three interdependent factors: the amount of the remaining undetermined differences after pre-guessing some key bits, the unknown number of steps required to extract the remaining key bit such that ϵ is minimized, and the propagation of determined values along with their associated filters in each step. To handle the unknown number of steps, we adapt the step-assignment method for integral attacks proposed by Hadipour *et al.* in [19]. That is, a sufficiently large total number of steps $\mathtt{S}$, say 10, is set in advance for computing ϵ. However, we also have to integrate advanced techniques (*e.g.*, state tests) and exploit cipher-specific properties (*e.g.*, non-full key additions), which necessitates multiple, often iterative and interdependent propagation phases, thereby increasing the difficulty of

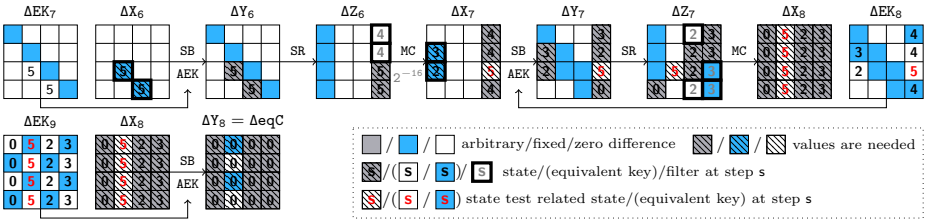

Figure 8: The toy example of the ϵ calculation in the related-key model

modeling ϵ calculation in differential-like attacks well beyond a straightforward step assignment.

Given $\mathtt{S}$, each state cell and each key cell involved in the key recovery is assigned a step number $\mathtt{s} \in \{-1, \ldots, \mathtt{S}\}$. The step number 0 is set aside for the preguessing phase, and the inner cells get a larger number. The state test can be activated at any step throughout this process. When differential propagation is not synchronized with value propagation, two interdependent sets of variables are required to describe them. The time complexity of each step is directly determined by the number of keys processed and the number of associated filters. When using the precomputed hash table approach, the memory complexity of each table must also be considered, which depends on the set of start states for that step.

We present Algorithm 2, which aligns with the toy example of the three-round `Deoxys` in Figure 8, to illustrate the core model of the ϵ calculation. It takes a step bound $\mathtt{S}$ as input, which ensures enough steps to reach minimal complexity while allowing unused steps. We introduce three variables dedicated to this model, $\mathtt{SsEK}_\epsilon^\mathtt{s}$, $\mathtt{frSB}_\epsilon^\mathtt{s}$, and $\mathtt{frMC}_\epsilon^\mathtt{s}$, which denote at step $\mathtt{s}$, respectively, the number of processed keys and tested states, and filters derived from `SB` and `MC`. The model is initialized in lines $1 \sim 6$: all uninvolved states and keys are assigned -1, while previously determined ones in pre-guessing (*e.g.*, $\mathtt{X}_8^{\{0,\ldots,3\}}$ and $\mathtt{EK}_8^{\{0,\ldots,3\}}$ in Figure 8) are set to 0. During the step propagation (lines $7 \sim 11$), we follow the rule that inner states correspond to larger steps and gradually increase the step when going through `AEK`. Step propagation in our model is fully compatible with the elimination propagation of the state test in Algorithm 1, thereby ensuring that the state test can be seamlessly incorporated into the ϵ calculation. This is illustrated by the red-labeled cells in Figures 7 and 8. In line 14, all keys processed and states tested at step $\mathtt{s}$ are aggregated into $\mathtt{SsEK}_\epsilon^\mathtt{s}$. For example, at step 5, $\mathtt{SsEK}_\epsilon^5 = 2 + 2 \times 1 = 4$ correspond to $\mathtt{EK}_7^{6,11}$ and $\mathtt{X}_7^{14}$. In lines $15 \sim 19$, we count the filters from `SB` and `MC` at each step. We introduce temporary variables $\mathtt{MaxDiffsX}_\mathtt{r}^\mathtt{col}$ and $\mathtt{KnowCdZ}_\mathtt{r}^\mathtt{col}$ to guarantee that the filters corresponding to `MC` are counted at step $\mathtt{s}$ once all unknown differences after `MC` are determined at that step. For instance, the filters at $\Delta\mathtt{Z}_6^{12,13}$ are accounted for at step 4.

Following Algorithm 2, we provide the complexity evaluation on ϵ in Appendix B.3, which supports both the guess-and-filter approach and the precomputation hash table method. When applying Algorithm 2 to ciphers with non-full key addition, such as `SKINNY`, differential propagation may proceed out of sync with value propagation and must be handled separately. In such cases, efficiency

Algorithm 2: The core part of ϵ calculation aligning with Figure 8

Input: variables in Table 5; maximum number of steps $\mathtt{S}$.

for $\mathtt{V} \in \{\mathtt{X},\mathtt{Y},\mathtt{Z}\}, \mathtt{r} \in \{6,7,8\}, \mathtt{c} \in \{0,\ldots,15\}$ **do**

  // exclude uninvolved states, mark determined states

$$\begin{cases} \mathtt{sV}^{\mathtt{c}}_{\mathtt{r}} = -1 & \leftarrow \mathtt{vV}^{\mathtt{c}}_{\mathtt{r}} = 0 \\ \mathtt{sV}^{\mathtt{c}}_{\mathtt{r}} = 0 & \leftarrow (\mathtt{detV}^{\mathtt{c}}_{\mathtt{r}} = 1) \vee (\mathtt{eV}^{\mathtt{c}}_{\mathtt{r}} = 1) \\ \mathtt{sV}^{\mathtt{c}}_{\mathtt{r}} \in \{1,\ldots,\mathtt{S}\} & \leftarrow \mathtt{otherwise} \end{cases}$$

for $\mathtt{r} \in \{6,7,8\}, \mathtt{c} \in \{0,\ldots,15\}$ **do**

  // exclude uninvolved subkeys, mark pre-guessed subkeys

$$\begin{cases} \mathtt{sEK}^{\mathtt{c}}_{\mathtt{r}} = -1 & \leftarrow \mathtt{vEK}^{\mathtt{c}}_{\mathtt{r}} = 0 \\ \mathtt{sEK}^{\mathtt{c}}_{\mathtt{r}} = 0 & \leftarrow (\mathtt{gEK}^{\mathtt{c}}_{\mathtt{r}} = 1) \vee (\mathtt{eEK}^{\mathtt{c}}_{\mathtt{r}} = 1) \\ \mathtt{sEK}^{\mathtt{c}}_{\mathtt{r}} \geq \mathtt{sZ}^{\mathtt{SR}^{-1}(\mathtt{c})}_{\mathtt{r}} & \leftarrow \mathtt{otherwise} \end{cases}$$

  // step propagation in SB, SR and AEK

$$\mathtt{sX}^{\mathtt{c}}_{\mathtt{r}} = \max\left(\mathtt{sEK}^{\mathtt{c}}_{\mathtt{r}+1}, \mathtt{sEY}^{\mathtt{c}}_{\mathtt{r}}\right) \wedge \left(\mathtt{sY}^{\mathtt{SR}(\mathtt{c})}_{\mathtt{r}} = \mathtt{sZ}^{\mathtt{c}}_{\mathtt{r}}\right)$$

for $\mathtt{r} \in \{6,7,8\}, \mathtt{c} \in \{0,4,8,12\}, \mathtt{i} \in \{0,\ldots,3\}$ **do**

  // step propagation in MC

$$\mathtt{sZ}^{\mathtt{c}+\mathtt{i}}_{\mathtt{r}} = \left(\max_{\mathtt{j}\in\{0,\ldots,3\}}\left(\mathtt{sX}^{\mathtt{c}+\mathtt{j}}_{\mathtt{r}}\right)\right) \leftarrow (\mathtt{vZ}^{\mathtt{c}+\mathtt{i}}_{\mathtt{r}} = 1)$$

for $\mathtt{s} \in \{1,\ldots,\mathtt{S}\}$ **do**

  // count processed subkeys and tested states

$$\mathtt{SsEK}^{\mathtt{s}}_{\epsilon} = \sum_{\mathtt{r}\in\{6,7,8\}}^{\mathtt{c}\in\{0,\ldots,15\}} \begin{pmatrix} ((\mathtt{sEK}^{\mathtt{c}}_{\mathtt{r}+1} = \mathtt{s}) \wedge (\mathtt{eEK}^{\mathtt{c}}_{\mathtt{r}+1} \neq 2)) + \\ 2 \times ((\mathtt{tX}^{\mathtt{c}}_{\mathtt{r}} = 2) \wedge (\mathtt{sX}^{\mathtt{c}}_{\mathtt{r}} = \mathtt{s})) \end{pmatrix}$$

  // count filters from SB and MC

$$\mathtt{frSB}^{\mathtt{s}}_{\epsilon} = \sum_{\mathtt{r}\in\{6,7,8\}}^{\mathtt{c}\in\{0,\ldots,15\}} ((\mathtt{sX}^{\mathtt{c}}_{\mathtt{r}} = \mathtt{s}) \wedge (\mathtt{dX}^{\mathtt{c}}_{\mathtt{r}} = 1))$$

for $\mathtt{r} \in \{6,7,8\}, \mathtt{col} \in \{0,4,8,12\}$ **let**

$$\begin{bmatrix} \mathtt{MaxDiffsX}^{\mathtt{col}}_{\mathtt{r}} = \max_{\mathtt{i}\in\{0,\ldots,3\}}(\mathtt{sX}^{\mathtt{col}+\mathtt{i}}_{\mathtt{r}} \times (\mathtt{dX}^{\mathtt{col}+\mathtt{i}}_{\mathtt{r}} = 2)); \\ \mathtt{KnowCdZ}^{\mathtt{col}}_{\mathtt{r}} = \sum_{\mathtt{i}\in\{0,\ldots,3\}} (\mathtt{dZ}^{\mathtt{col}+\mathtt{i}}_{\mathtt{r}} \leq 1). \end{bmatrix}$$

$$\mathtt{frMC}^{\mathtt{s}}_{\epsilon} = \sum_{\mathtt{r}\in\{6,7,8\}}^{\mathtt{col}\in\{0,4,8,12\}} (\mathtt{KnowCdZ}^{\mathtt{col}}_{\mathtt{r}} \times (\mathtt{MaxDiffsX}^{\mathtt{col}}_{\mathtt{r}} = \mathtt{s}))$$

can be improved by exploiting SB property for extra key derivation. This process is formalized in Appendix B.2.

4.3 Model for Key Bridging

We present models for the strong key-bridges detailed in Section 3, which can reduce the number of involved key bits compared with classical approaches. Because strong key bridges can be used when the number of rounds under attack is sufficiently large, as in the attacks on Deoxys-BC-384 and SKINNY-n-$3n$, we illustrate our model using them.

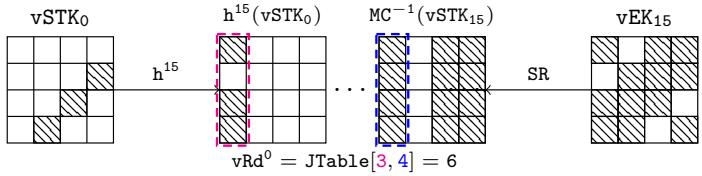

Figure 9: A toy example of counting the redundant key cells for `Deoxys-BC-384`

Algorithm 3: Strong key bridging of `Deoxys-BC-384` (Figure 9)

Input: the variables in Table 5; the subkey permutation $\mathtt{h}$; the tweakey schedule property table `JTable` (Table 4).

for $\mathtt{col} \in \{0, 4, 8, 12\}$ **do**

  // counting for redundant subkey bits in each column

  let

$$\begin{bmatrix} \mathtt{ExtvEK}^{\mathtt{col}} = \exists_{\mathtt{i}\in\{0,\ldots,3\}} \left(\mathtt{vSTK}_0^{\mathtt{h}^{15}(\mathtt{col}+\mathtt{i})}\right) \wedge \exists_{\mathtt{i}\in\{0,\ldots,3\}} \left(\mathtt{vEK}_{15}^{\mathtt{SR}(\mathtt{col}+\mathtt{j})}\right) \\ \mathtt{SvSTK}_0^{\mathtt{col}} = \sum_{\mathtt{i}\in\{0,\ldots,3\}} \left(\mathtt{vSTK}_0^{\mathtt{h}^{15}(\mathtt{col}+\mathtt{i})}\right) \\ \mathtt{SvEK}_{15}^{\mathtt{col}} = \sum_{\mathtt{i}\in\{0,\ldots,3\}} \left(\mathtt{vEK}_{15}^{\mathtt{SR}(\mathtt{col}+\mathtt{i})}\right) \end{bmatrix}$$

  $\mathtt{vRd}^{\mathtt{col}} = \mathtt{JTable}\left[\mathtt{SvSTK}_0^{\mathtt{col}}, \mathtt{SvEK}_{15}^{\mathtt{col}}\right] \times \mathtt{ExtvEK}_{15}^{\mathtt{col}}$

$$\mathtt{m_b} + \mathtt{m_f} = \sum_{\mathtt{r}\in\{\mathtt{rE_b}\}\cup\{\mathtt{rE_f}\}}^{\mathtt{c}\in\{0,\ldots,15\}} \left(\mathtt{vEK}_\mathtt{r}^\mathtt{c} + 2 \times (\mathtt{tX}_\mathtt{r}^\mathtt{c} \geq 1)\right) - \sum_{\mathtt{col}\in\{0,4,8,12\}} \left(\mathtt{vRd}^{\mathtt{col}}\right)$$

Model the Subkey Relation for `Deoxys-BC-384`. To leverage the strong key bridging in `Deoxys-BC-384`, we use the *table-lookup* approach to count and exclude the redundant key bits, $|k_b \cap k_f|$, from the overall key count. We locate keys that lie within the same lane and column, and the column-wise corresponding number of redundant key bits is then directly obtained via table lookup.

In Algorithm 3, we embed properties of the `Deoxys-BC-384` tweakey schedule (described in Section 3.1) into the model. A toy example is provided in Figure 9 for illustration. In this algorithm, we identify the column-wise relationship between $\mathtt{vSTK}_0$ and $\mathtt{vEK}_{15}$ within the same lane. Therefore, $\mathtt{h}^{15}$ and $\mathtt{SR}$ are applied to locate the cells position in $\mathtt{vSTK}_0$ and $\mathtt{vEK}_{15}$, respectively. The variable $\mathtt{ExtvEK}^{\mathtt{col}}$ identifies columns where $\mathtt{vSTK}_0$ and $\mathtt{vEK}_{15}$ lie in the same lane and both contain involved keys within that column. $\mathtt{SvSTK}_0^{\mathtt{col}}$ and $\mathtt{SvEK}_{15}^{\mathtt{col}}$ count, for each column, the number of involved keys in $\mathtt{vSTK}_0$ and $\mathtt{vEK}_{15}$ that belong to the same lane. By querying `Jtable`, the number of redundant key bits per column is obtained, which helps reduce the overall count of involved keys. In Figure 9, grid cells represent involved keys. The red and blue boxes correspond to the first column of $\mathtt{h}^{15}(\mathtt{vSTK}_0)$ and $\mathtt{SR}(\mathtt{vEK}_{15})$, with 3 and 4 involved key cells, respectively. Then 6 redundant key bits can be captured by querying `JTable`.

Algorithm 3 can be adapted for use in both the key-guessing strategy and the ϵ calculation. For the key-guessing strategy, we directly substitute the variables corresponding to the involved keys with those of the pre-guessed keys. In the ϵ calculation, we must ensure that 1) all keys processed in preceding steps are

eligible for redundancy counting, and 2) the redundancy is not counted multiple times across steps. For clarity, we provide details in Appendix B.4 on the integration of the key bridging model into the ϵ calculation.

Model the Subkey Relation for `SKINNY`-n-3n. For `SKINNY`, we introduce a *sliding-window* technique to count the number of involved subkeys, *i.e.*, the strong key bridge, and integrate it with the key pre-guessing and ϵ calculation phase. The details of the model are provided in Appendix B.5.

4.4 Elasticizing the Full-Fledged Model

When all components described in this section are activated, that is, when the complete model is employed to search for attack parameters, all potential complex interdependencies among them are taken into account. This enables the model to return a globally optimal solution without relying on any pre-assumptions. However, solving the complete model for large-round block ciphers can be computationally demanding. For instance, solving the model for a 15-round rectangle attack on `Deoxys-BC-384` takes over 20 hours on an *AMD Ryzen 9 7945HX* processor with 32 threads.

To balance optimality and efficiency, each component of our model is equipped with a configurable switch. Disabling deeper components accelerates solving and yields locally optimal solutions, which help prune the search space in subsequent phases. This section presents three elastic searching configurations that enable the full-fledged model to produce locally optimal solutions and provides guidance on how to progress toward a globally optimal result. With this elasticity, the solving time for the 15-round rectangle attack on `Deoxys-BC-384` is reduced to less than one hour.

- **Determine the length of distinguishers.** With only the distinguisher enabled, our model identifies the distinguisher of the highest probability for a given number of rounds. If the boomerang probability approaches 2^{-n} where n is the block size, it is recommended to reduce the distinguisher by one round to enhance the flexibility of the attack, which is justified by the principle that, under probabilistic extension, a distinguisher with fewer rounds can completely cover the longest one.
- **Differential pattern searching.** With the key guessing strategy and the ϵ calculation disabled, we formulate a multi-objective function as minimizing $\mathtt{a} \times \mathtt{P_r} + \mathtt{b} \times (\mathtt{m_b} + \mathtt{m_f})$, where constants $\mathtt{a}$ and $\mathtt{b}$ represent objective weights, and $\mathtt{m_b} + \mathtt{m_f}$ is computed by assisting with strong key bridging. This configuration helps identify differential patterns that are amenable to optimal key-recovery attacks. Then, by incorporating these pattern constraints into the full-fledged model, the subsequent search space can be pruned.
- **Attack boundary determination.** By disabling the ϵ calculation only and assuming $\epsilon = 1$, we can obtain the maximum achievable attack rounds and the corresponding lower bound on attack complexity. Then, by enabling ϵ calculation and imposing constraints derived from the previously obtained boundary, we guide the search toward solutions matching this bound.

5 Rectangle Attacks on Deoxys and SKINNY

In this section, applying our model to Deoxys-BC and SKINNY, we obtain a series of improved attacks. Utilizing the strong key bridges and the properties of h, we achieve a 15-round rectangle attack on Deoxys-BC-384 with its time complexity reduced by a factor of 2^{40}, and the first 33-round SKINNY-64-192 rectangle attack, which is 2 rounds more than the previous works. We also give improved attacks on Deoxys-BC-256, Deoxys-I, and SKINNY-128-384; for more details, refer to Appendix C. Before introducing the concrete attacks, we provide some experimental notes.

5.1 Experimental Results on Deoxys-BC and SKINNY

Comparison. To demonstrate the impact of the state test and the ϵ calculation in our model, we conduct experiments with these components selectively disabled. The corresponding results are summarized in Table 6. The first two cases demonstrate that the state test helps reduce time complexity. In contrast, the next cases reveal that without modeling the ϵ calculation, say assuming $\epsilon = 1$, the model returns an inaccurate time-complexity estimate, which hinders the identification of the globally optimal attack, and the last line shows the effect when closing the strong key bridge.

Table 6: Comparison of results with the state test and the ϵ calculation

Cipher	#R	State Test	ϵ Cal.	Key Bridge	Data	Memory	Time	ϵ
Deoxys-I-256	14	on	on	off	2^{117}	2^{120}	2^{242}	1
	14	off	on	off	2^{116}	2^{117}	2^{256}	1
Deoxys-BC-384	15	on	on	on	2^{115}	2^{128}	2^{334}	1
	15	on	off	on	2^{115}	2^{115}	2^{310}	$\gg 2^{24}$
	15	on	on	off	2^{115}	2^{115}	2^{348}	1

Configuration of the full-fledged model. To balance search efficiency and optimality, we adopt different configurations, detailed in Section 4.4, to solve the full-fledged model for rectangle attacks on Deoxys variants. For Deoxys-BC-256 and Deoxys-I-128, the complete model is employed directly, and the optimal attack parameters are obtained in a single solve. For Deoxys-BC-384 and Deoxys-I-256, we instead apply three elasticizing strategies that progressively prune the search space and ultimately identify the optimal configuration.

Attacks on SKINNY. With the strong key bridge and the models for key-recovery attack, we have substantially improved the rectangle attack on SKINNY-n-$3n$. For other variants of SKINNY, such as SKINNY-n-$2n$, however, no exploitable strong key bridge has been identified. A review of previous rectangle attacks on SKINNY-n-$2n$ shows that their key-recovery phases were already carefully designed and

finely tuned, achieving $\epsilon \approx 1$. This means a limited room for further improvements. Thus, we mainly focus on the attacks on SKINNY-n-3n. Our attacks on SKINNY-n-3n are given in Appendix C.2 and C.3.

5.2 Improved 15-Round Attack on Deoxys-BC-384

The search for attacks on Deoxys-BC-384 exploits the properties of the permutation h in the related-tweakey setting, as specified in Proposition 7. By flexibly selecting the number of rounds for E_0, E_1, and E_m, we obtain 32 byte-wise activeness patterns for the attack on 15 rounds with identical estimated complexities (as the probability is estimated before instantiating). We further identify a set of bit-wise solutions that individually have lower data, memory, or time complexity. The attack with lower time complexity achieves $D, M, T = (2^{122.2}, 2^{122.2}, 2^{331.2})$, while the attack with lower memory complexity yields $D, M, T = (2^{115.2}, 2^{115.2}, 2^{352.4})$. To avoid redundancy, we present the attack, as depicted in Figure 10, which offers a better overall trade-off among the three metrics. More details of other attacks on Deoxys-BC-384 are provided in the code repository.

Data complexity. The probability of the whole rectangle attack is $P^2 = 2^{-97.4}$, and the other parameters of the attack are: $m_b = r_b = 80$, $r_f = 128$, and $m_f = (16+13+6+3) \times 8 = 304$ instead of 336 due to the state test. It is noted that $|k_b \cup k_f| = 341$ and $|k_b \cap k_f| = 43$ due to the strong key bridge. The data complexity is $D_R = 4D = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{114.7}$, where D is the data complexity under each of the four related keys.

Time complexity. The best guessing parameters are $m'_b = r'_b = 80, m'_f = (16 + 5 + 1) \times 8 = 176, r'_f = 56, r^*_b = 0, r^*_f = 72$, which means guessing 10 bytes of k_b and 22 bytes of k_f. In particular, there exists a strong key bridge between k'_b and k'_f, leading to $|k'_b \cup k'_f| = 216$ and $|k'_b \cap k'_f| = 40$. The time complexity of our new attack is as follows.

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{114.7}, \\
T_1 &= 2^{|k'_b \cup k'_f|} \cdot D_R = \sqrt{s} \cdot 2^{216+114.7} = \sqrt{s} \cdot 2^{330.7}, \\
T_2 &= 2^{|k'_b \cup k'_f|} \cdot D \cdot 2^{r^*_b} = \sqrt{s} \cdot 2^{328.7}, \\
T_3 &= 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{329.4} \cdot \epsilon,
\end{aligned}
$$

and $T_4 = 2^{k-\ell}$ for the exhaustive search. As detailed in Table 7, we can achieve $\epsilon = 1$ table look-up, which is calculated using six pre-computed hash tables. As shown in Figure 10, we employ the state-test technique in Step 6. Throughout all steps, we utilize the strong key-bridge technique. After Step 6, we further filter the quartets using three dependent subtweakey bits, resulting in $s \cdot 2^{329.4-16-3} = s \cdot 2^{310.4}$ final quartets.

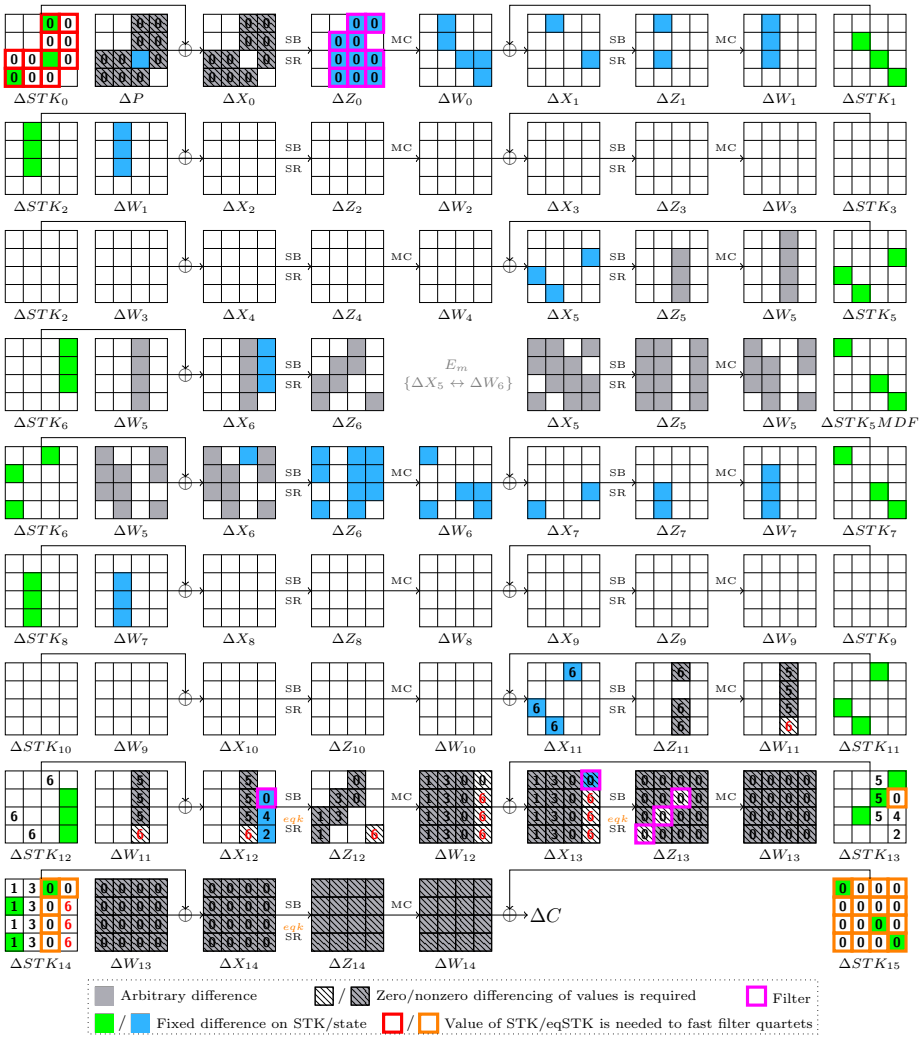

Figure 10: Rectangle attack on 15-round `Deoxys-BC-384`. To maintain symmetry across all rounds, the subteakeys shown for E_f are not equivalent keys. Nevertheless, the step number is placed according to the position of the equivalent keys.

Table 7: Pre-computed tables for the 15-round attack on `Deoxys-BC-384`.

Step	Starting bytes	Involved subtweakey	Filter	Memory	Time
1	$Z_{13}[0,7,10,13]$	$eqSTK_{14}[0,1,2,3]$	$\Delta Z_{12}[0,1]$	$2^{16\times 8}$	1
2	$Z_{12}[3]$	$eqSTK_{13}[15]$	$\Delta Y_{12}[15]$	$2^{9\times 8}$	2^{-8}
3	$Z_{13}[1,4,11,14]$	$eqSTK_{14}[4,5,6,7]$	$\Delta Z_{12}[4,7]$	$2^{16\times 8}$	2^{-8}
4	$Z_{12}[6]$	$eqSTK_{13}[14]$	$\Delta Y_{12}[14]$	$2^{9\times 8}$	2^{-16}
5	$Z_{12}[2,5,8]$	$eqSTK_{13}[8,9,10]$	$\Delta Z_{11}[10]$	$2^{13\times 8}$	2^{-8}
6	$W_{11}[8,9,10,11]$	$eqSTK_{12}[2,7,8]$	$\Delta Y_{11}[2,5,8]$	$2^{12\times 8}$	2^{-16}

Memory complexity. The memory complexity of our attack is $M_R = D_R + D \cdot 2^{r_b^*} + 2^{|k_b \cup k_f| - |k_b' \cup k_f'|} + M_\epsilon = 2^{128}$.

If we set $s = 4, \ell = 68$, then the data, memory, and time complexities of our attack are $2^{115.7}, 2^{128}, 2^{331.7}$, respectively. The success probability of the attack is 97.7% even when accounting for the effect of the state test discussed in [43].

Remark. In the attack above, the tweakey is 384 bits. If we treat a portion of the tweakey as tweak–specifically $t = 64$ and $k = 320$–the 15-round attack remains feasible. For example, set eight cells $TK1[2, 3, 6, 7, 8, 9, 12, 13]$ to be tweak. By Proposition 3, $STK_0[2, 3, 6, 7, 8, 9, 12, 13]$ carries the same information as $STK_{15}[0, 2, 9, 11, 12, 13, 14, 15]$. When pre-guessing the same subtweakey cells as in the above attack, we have $|k_b' \cup k_f'| = 216 - 32 = 184$. Therefore, a 15-round attack can be achieved with the time complexity reduced by a factor of 2^{32}. Namely, the data, memory, and time complexities of the attack are $2^{115.7}, 2^{128}, 2^{297.7}$, respectively. It is emphasized that such a 15-round attack on `Deoxys-BC-384` cannot be realized without leveraging the strong bridge between STK_0 and STK_{15}.

6 Conclusion

In this paper, we identified strong dependencies between subtweakeys arising from the LFSRs and revealed structural properties of the cell-position permutation employed in the tweakey schedule for `Deoxys`. Furthermore, we devised a full-fledged automatic constraint programming model that enables one-step search for rectangle attacks. In our new model, three novel components are integrated: the state-test technique, explicit ϵ computation, and the strong key bridges. Applying this model to `Deoxys-BC` and `SKINNY` showed that prior key-recovery algorithms overlooked stronger rectangle attacks. By leveraging this new model, we presented improved attacks on `Deoxys-BC`, which feature reduced time and data complexities. We also extended the previous attacks on `Deoxys-I` by one additional round. Notably, for `SKINNY`, our attacks achieved 1 or 2 more rounds of cryptanalysis compared to the best previously reported attacks. Finally, we note that the model for explicit ϵ computation can be broadly applicable to other differential-style attacks (*e.g.*, truncated or impossible-differential attacks) where involved key bits are extracted from data pairs instead of quartets, suggesting potential applications in future one-step automatic search models.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported by the National Key R&D Program of China (No. 2024YFA1013000), the National Natural Science Foundation of China (Grants 62372213, 62132008, 62572464,62372294), the National Cryptologic Science Fund of China (Grant No. 2025NCSF01012), the Guangdong Special Support Program for Distinguished Talents (No. 2024JC08X015), the Shenzhen Major Science and Technology Special Project (No. KJZD20240903101108011). Yincen Chen was also supported by the Program of China Scholarship Council (Grant CSC202506780028).

References

1. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014-2019), https://competitions.cr.yp.to/caesar.html
2. NIST: Lightweight cryptography competition (2019-2023), https://csrc.nist.gov/projects/lightweight-cryptography
3. Information security encryption algorithms part 7: Tweakable block ciphers (2022), https://www.iso.org/standard/80505.html, includes Deoxys-TBC and SKINNY as specified TBCs
4. Ahmadian, Z., Khalesi, A., M'foukh, D., Moghimi, H., Naya-Plasencia, M.: Improved differential meet-in-the-middle cryptanalysis. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 280–309. Springer (2024), https://doi.org/10.1007/978-3-031-58716-0_10
5. Bariant, A., Leurent, G.: Truncated boomerang attacks and application to AES-Based ciphers. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 3–35. Springer (2023), https://doi.org/10.1007/978-3-031-30634-1_1
6. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016), https://doi.org/10.1007/978-3-662-53008-5_5
7. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. pp. 340–357. Lecture Notes in Computer Science, Springer (2001), https://doi.org/10.1007/3-540-44987-6_21
8. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2365, pp. 1–16. Springer (2002), https://doi.org/10.1007/3-540-45661-9_1
9. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 179–199. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_10
10. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: A security analysis of Deoxys and its internal tweakable block ciphers. IACR Trans. Symmetric Cryptol. **2017**(3), 73–107 (2017), https://doi.org/10.13154/tosc.v2017.i3.73-107

11. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: a new cryptanalysis tool. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 683–714. Springer (2018), https://doi.org/10.1007/978-3-319-78375-8_22
12. Cogliati, B., Lampe, R., Seurin, Y.: Tweaking Even-Mansour ciphers. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 189–208. Springer (2015), https://doi.org/10.1007/978-3-662-47989-6_9
13. Delaune, S., Derbez, P., Vavrille, M.: Catching the fastest boomerangs application to SKINNY. IACR Trans. Symmetric Cryptol. **2020**(4), 104–129 (2020), https://doi.org/10.46586/tosc.v2020.i4.104-129
14. Dong, X., Qin, L., Sun, S., Wang, X.: Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 3–33. Springer (2022), https://doi.org/10.1007/978-3-031-07082-2_1
15. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The SKEIN hash function family (2008)
16. Goldenberg, D., Hohenberger, S., Liskov, M.D., Schwartz, E.C., Seyalioglu, H.: On tweaking Luby-Rackoff blockciphers. In: Kurosawa, K. (ed.) Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4833, pp. 342–356. Springer (2007), https://doi.org/10.1007/978-3-540-76900-2_21
17. Grochow, T., List, E., Nandi, M.: Dovemac: A TBC-based PRF with smaller state, full security, and high rate. IACR Trans. Symmetric Cryptol. **2019**(3), 43–80 (2019), https://doi.org/10.13154/tosc.v2019.i3.43-80
18. Hadipour, H., Bagheri, N., Song, L.: Improved rectangle attacks on SKINNY and CRAFT. IACR Trans. Symmetric Cryptol. **2021**(2), 140–198 (2021), https://doi.org/10.46586/tosc.v2021.i2.140-198
19. Hadipour, H., Gerhalter, S., Sadeghi, S., Eichlseder, M.: Improved search for integral, impossible differential and zero-correlation attacks application to Ascon, Fork-SKINNY, SKINNY, MANTIS, PRESENT and QARMAv2. IACR Trans. Symmetric Cryptol. **2024**(1), 234–325 (2024), https://doi.org/10.46586/tosc.v2024.i1.234-325
20. Hadipour, H., Sadeghi, S., Eichlseder, M.: Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 128–157. Springer (2023), https://doi.org/10.1007/978-3-031-30634-1_5
21. Hirose, S., Minematsu, K.: Compactly committing authenticated encryption using encryptment and tweakable block cipher. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) Selected Areas in Cryptography - SAC 2023 - 30th International Conference, Fredericton, Canada, August 14-18, 2023, Revised Selected Papers. Lecture Notes

in Computer Science, vol. 14201, pp. 233–252. Springer (2023), https://doi.org/10.1007/978-3-031-53368-6_12

22. Hoang, V.T., Menda, S.: Robust AE with committing security. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part IX. Lecture Notes in Computer Science, vol. 15492, pp. 312–342. Springer (2024), https://doi.org/10.1007/978-981-96-0947-5_11
23. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A fast tweakable block cipher mode for highly secure message authentication. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 34–65. Springer (2017), https://doi.org/10.1007/978-3-319-63697-9_2
24. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer (2014), https://doi.org/10.1007/978-3-662-45608-8_15
25. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41. Submitted to CAESAR **124** (2016)
26. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 14–30. Springer (2012), https://doi.org/10.1007/978-3-642-32009-5_2
27. Liskov, M.D., Rivest, R.L., Wagner, D.A.: Tweakable block ciphers. In: Yung, M. (ed.) Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002), https://doi.org/10.1007/3-540-45708-9_3
28. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings. IACR Trans. Symmetric Cryptol. **2017**(3), 37–72 (2017), https://doi.org/10.13154/tosc.v2017.i3.37-72
29. Mennink, B.: Optimally secure tweakable blockciphers. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 428–448. Springer (2015), https://doi.org/10.1007/978-3-662-48116-5_21
30. M'foukh, D., Naya-Plasencia, M., Neumann, P.: The state-test technique on differential attacks: a 26-round attack on craft and other applications. In: Hanaoka, G., Yang, B. (eds.) Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 16245, pp. 253–284. Springer (2025), https://doi.org/10.1007/978-981-95-5018-0_9

31. Moazami, F., Mehrdad, A., Soleimany, H.: Impossible differential cryptanalysis on Deoxys-BC-256. ISC Int. J. Inf. Secur. **10**(2), 93–105 (2018), https://doi.org/10.22042/isecure.2018.114245.405
32. Naito, Y.: Full PRF-secure message authentication code based on tweakable block cipher. In: Au, M.H., Miyaji, A. (eds.) Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9451, pp. 167–182. Springer (2015), https://doi.org/10.1007/978-3-319-26059-4_9
33. Naito, Y., Sugawara, T.: Lightweight authenticated encryption mode of operation for tweakable block ciphers. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 66–94 (2020), https://doi.org/10.13154/tches.v2020.i1.66-94
34. Peyrin, T., Seurin, Y.: Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 33–63. Springer (2016), https://doi.org/10.1007/978-3-662-53018-4_2
35. Qin, L., Dong, X., Wang, A., Hua, J., Wang, X.: Mind the TWEAKEY schedule: Cryptanalysis on skinnye-64-256. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 287–317. Springer (2022), https://doi.org/10.1007/978-3-031-22963-3_10
36. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004), https://doi.org/10.1007/978-3-540-30539-2_2
37. Sasaki, Y.: Improved related-tweakey boomerang attacks on Deoxys-BC. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 87–106. Springer (2018), https://doi.org/10.1007/978-3-319-89339-6_6
38. Schroeppel, R.: The Hasty pudding cipher (1998)
39. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology **21**(1), 131–147 (2008), http://dx.doi.org/10.1007/s00145-007-9013-7
40. Shen, Y., Peters, T., Standaert, F.: Multiplex: TBC-based authenticated encryption with Sponge-like rate. IACR Trans. Symmetric Cryptol. **2024**(2), 1–34 (2024), https://doi.org/10.46586/tosc.v2024.i2.1-34
41. Shi, D., Sun, S., Song, L., Hu, L., Yang, Q.: Exploiting non-full key additions: Full-fledged automatic Demirci-Selçuk meet-in-the-middle cryptanalysis of SKINNY. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 67–97. Springer (2023), https://doi.org/10.1007/978-3-031-30634-1_3
42. Song, L., Qin, X., Hu, L.: Boomerang connectivity table revisited: Application to SKINNY and AES. IACR Trans. Symmetric Cryptol. **2019**(1), 118–141 (2019), https://doi.org/10.13154/tosc.v2019.i1.118-141

43. Song, L., Yang, Q., Chen, Y., Hu, L., Weng, J.: Probabilistic extensions: A one-step framework for finding rectangle attacks and beyond. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 339–367. Springer (2024), https://doi.org/10.1007/978-3-031-58716-0_12
44. Song, L., Zhang, N., Yang, Q., Shi, D., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle attacks: A unified and generic framework for key recovery. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 410–440. Springer (2022), https://doi.org/10.1007/978-3-031-22963-3_14
45. Wang, H., Peyrin, T.: Boomerang switch in multiple rounds. application to AES variants and deoxys. IACR Trans. Symmetric Cryptol. **2019**(1), 142–169 (2019), https://doi.org/10.13154/tosc.v2019.i1.142-169
46. Wang, L., Guo, J., Zhang, G., Zhao, J., Gu, D.: How to build fully secure tweakable blockciphers from classical blockciphers. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 455–483 (2016), https://doi.org/10.1007/978-3-662-53887-6_17
47. Yang, Q., Song, L., Zhang, N., Shi, D., Wang, L., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. J. Cryptol. **37**(2), 19 (2024), https://doi.org/10.1007/s00145-024-09499-1
48. Zhao, B., Dong, X., Jia, K.: New related-tweakey boomerang and rectangle attacks on Deoxys-BC including BDT effect. IACR Trans. Symmetric Cryptol. **2019**(3), 121–151 (2019), https://doi.org/10.13154/tosc.v2019.i3.121-151
49. Zhao, B., Dong, X., Jia, K., Meier, W.: Improved related-tweakey rectangle attacks on reduced-round Deoxys-BC-384 and Deoxys-I-256-128. In: Hao, F., Ruj, S., Gupta, S.S. (eds.) Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11898, pp. 139–159. Springer (2019), https://doi.org/10.1007/978-3-030-35423-7_7
50. Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G.: Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. Des. Codes Cryptogr. **88**(6), 1103–1126 (2020), https://doi.org/10.1007/s10623-020-00730-1
51. Zong, R., Dong, X., Wang, X.: Related-tweakey impossible differential attack on reduced-round Deoxys-BC-256. Sci. China Inf. Sci. **62**(3), 32102:1–32102:12 (2019), https://doi.org/10.1007/s11432-017-9382-2

Supplementary Materials

A More Tweakey Schedule Properties

A.1 Dependency of Subtweakeys

Let $(y_7, y_6, \cdots, y_0)^\top = (L_2^{15} + \boldsymbol{I}) \cdot (x_7, x_6, \cdots, x_0)^\top$. The rank of $\boldsymbol{L_2^{15} + I}$ is 4, as there are 4 linear relations, as shown below.

$$\boldsymbol{L_2^{15} + I} = \begin{pmatrix} 1&1&0&1&0&0&0&1 \\ 1&1&0&0&1&0&0&0 \\ 0&1&1&0&0&1&0&0 \\ 0&0&1&1&0&0&1&0 \\ 0&0&0&1&1&0&0&1 \\ 1&0&1&0&1&1&0&0 \\ 0&1&0&1&0&1&1&0 \\ 0&0&1&0&1&0&1&1 \end{pmatrix}, \quad \begin{aligned} y_4 &= y_0 + y_3, \\ y_5 &= y_0 + y_1 + y_3, \\ y_6 &= y_0 + y_1 + y_2 + y_3, \\ y_7 &= y_0 + y_1 + y_2. \end{aligned} \tag{1}$$

Let K_i $(0 \le i < 384)$ denote the master key bits. Suppose TK^1 takes $K_0 \sim K_{127}$, TK^2 takes $K_{128} \sim K_{255}$, and TK^3 takes $K_{256} \sim K_{383}$, respectively.

If $STK_r[i_r] = TK^1[i] \oplus LFSR_2^r(TK^2[i]) \oplus LFSR_3^r(TK^3[i])$ and let $i = 0$, we have

$$\begin{aligned}
STK_0[0][0] &= K_0 + K_{128} + K_{256}, \\
STK_0[0][1] &= K_1 + K_{129} + K_{257}, \\
STK_0[0][2] &= K_2 + K_{130} + K_{258}, \\
STK_0[0][3] &= K_3 + K_{131} + K_{259}, \\
STK_0[0][4] &= K_4 + K_{132} + K_{260}, \\
STK_0[0][5] &= K_5 + K_{133} + K_{261}, \\
STK_0[0][6] &= K_6 + K_{134} + K_{262}, \\
STK_0[0][7] &= K_7 + K_{135} + K_{263}, \\
STK_{15}[1][0] &= K_0 + K_{129} + K_{131} + K_{133} + K_{257} + K_{259} + K_{261}, \\
STK_{15}[1][1] &= K_1 + K_{130} + K_{132} + K_{134} + K_{258} + K_{260} + K_{262}, \\
STK_{15}[1][2] &= K_2 + K_{131} + K_{133} + K_{135} + K_{259} + K_{261} + K_{263}, \\
STK_{15}[1][3] &= K_3 + K_{128} + K_{132} + K_{256} + K_{260}, \\
STK_{15}[1][4] &= K_4 + K_{129} + K_{133} + K_{257} + K_{261}, \\
STK_{15}[1][5] &= K_5 + K_{130} + K_{134} + K_{258} + K_{262}, \\
STK_{15}[1][6] &= K_6 + K_{131} + K_{135} + K_{259} + K_{263}, \\
STK_{15}[1][7] &= K_7 + K_{128} + K_{132} + K_{134} + K_{256} + K_{260} + K_{262}.
\end{aligned}$$

Among them, there are four dependent linear relations as follows.

$$\begin{aligned}
STK_{15}[1][4] = \; & STK_0[0][0] + STK_0[0][3] + STK_0[0][4] + STK_{15}[1][0] + \\
& STK_{15}[1][3], \\
STK_{15}[1][5] = \; & STK_0[0][0] + STK_0[0][1] + STK_0[0][3] + STK_0[0][5] + \\
& STK_{15}[1][0] + STK_{15}[1][1] + STK_{15}[1][3], \\
STK_{15}[1][6] = \; & STK_0[0][0] + STK_0[0][1] + STK_0[0][2] + STK_0[0][3] + \\
& STK_0[0][6] + STK_{15}[1][0] + STK_{15}[1][1] + STK_{15}[1][2] + \\
& STK_{15}[1][3], \\
STK_{15}[1][7] = \; & STK_0[0][0] + STK_0[0][1] + STK_0[0][2] + STK_0[0][7] + \\
& STK_{15}[1][0] + STK_{15}[1][1] + STK_{15}[1][2].
\end{aligned} \tag{2}$$

If $STK_r[i_r] = LFSR_2^r(TK^2[i]) \oplus LFSR_3^r(TK^3[i])$ and let $i = 0$, we have

$$\begin{aligned}
STK_0[0][0] &= K_{128} + K_{256}, \\
STK_0[0][1] &= K_{129} + K_{257}, \\
STK_0[0][2] &= K_{130} + K_{258}, \\
STK_0[0][3] &= K_{131} + K_{259}, \\
STK_0[0][4] &= K_{132} + K_{260}, \\
STK_0[0][5] &= K_{133} + K_{261}, \\
STK_0[0][6] &= K_{134} + K_{262}, \\
STK_0[0][7] &= K_{135} + K_{263}, \\
STK_{15}[1][0] &= K_{129} + K_{131} + K_{133} + K_{257} + K_{259} + K_{261}, \\
STK_{15}[1][1] &= K_{130} + K_{132} + K_{134} + K_{258} + K_{260} + K_{262}, \\
STK_{15}[1][2] &= K_{131} + K_{133} + K_{135} + K_{259} + K_{261} + K_{263}, \\
STK_{15}[1][3] &= K_{128} + K_{132} + K_{256} + K_{260}, \\
STK_{15}[1][4] &= K_{129} + K_{133} + K_{257} + K_{261}, \\
STK_{15}[1][5] &= K_{130} + K_{134} + K_{258} + K_{262}, \\
STK_{15}[1][6] &= K_{131} + K_{135} + K_{259} + K_{263}, \\
STK_{15}[1][7] &= K_{128} + K_{132} + K_{134} + K_{256} + K_{260} + K_{262}.
\end{aligned}$$

It can be seen that $STK_{15}[1]$ can be derived from $STK_0[0]$. Specifically,

$$\begin{aligned}
STK_{15}[1][0] &= STK_0[0][1] + STK_0[0][3] + STK_0[0][5], \\
STK_{15}[1][1] &= STK_0[0][2] + STK_0[0][4] + STK_0[0][6], \\
STK_{15}[1][2] &= STK_0[0][3] + STK_0[0][5] + STK_0[0][7], \\
STK_{15}[1][3] &= STK_0[0][0] + STK_0[0][4], \\
STK_{15}[1][4] &= STK_0[0][1] + STK_0[0][5], \\
STK_{15}[1][5] &= STK_0[0][2] + STK_0[0][6], \\
STK_{15}[1][6] &= STK_0[0][3] + STK_0[0][7], \\
STK_{15}[1][7] &= STK_0[0][0] + STK_0[0][4] + STK_0[0][6].
\end{aligned} \tag{3}$$

A.2 Proof of Proposition 2

Proof. Similarly to Proposition 1, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]|$ equals the rank of matrix $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{L}_2^m & \boldsymbol{L}_3^m \\ \boldsymbol{I} & \boldsymbol{L}_2^n & \boldsymbol{L}_3^n \end{bmatrix}$.

- When $n \neq 30, m = 15$,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2^{15} + \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{L}_2^{2n} + \boldsymbol{I} \end{bmatrix}\right).$$

  Since $\texttt{Rank}(\boldsymbol{L}_2^{15} + \boldsymbol{I}) = 4, \texttt{Rank}(\boldsymbol{L}_2^{2n} + \boldsymbol{I}) = 8$, we get $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = \texttt{Rank}(\boldsymbol{C}) = 20$.
  For $n = 15$ or $n - m = 15$, the results are the same as in the case of $m = 15$, and will not be elaborated here.
- When $n = 30, m = 15$,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2^{15} + \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right),$$

  we get $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = \texttt{Rank}(\boldsymbol{C}) = 12$.
- When $n = 30, m \neq 15$,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2^m + \boldsymbol{I} & \boldsymbol{L}_2^{-m} + \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right),$$

  we get $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = \texttt{Rank}(\boldsymbol{C}) = 16$.
- When (m, n) in other cases,

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2^m + \boldsymbol{I} & \boldsymbol{L}_2^{-m} + \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{L}_2^n + \boldsymbol{I} & \boldsymbol{L}_2^{-n} + \boldsymbol{I} \end{bmatrix}\right).$$

  Due to the determinant of $\boldsymbol{C}$ is $\det(\boldsymbol{C}) = -\det(\boldsymbol{L}_2^{-m}(\boldsymbol{L}_2^{m-n} + \boldsymbol{I})(\boldsymbol{L}_2^m + \boldsymbol{I})(\boldsymbol{L}_2^n + \boldsymbol{I})) \neq 0$, we can get that $\texttt{Rank}(\boldsymbol{C}) = 24$ and $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 24$.

□

B More Detailed Models

B.1 Model for Key-Guessing Strategy Allowing the State Test

In Algorithm 4, tested states are treated as determined states, as shown in line 4, and contribute to filter generation when combined with other determined states.

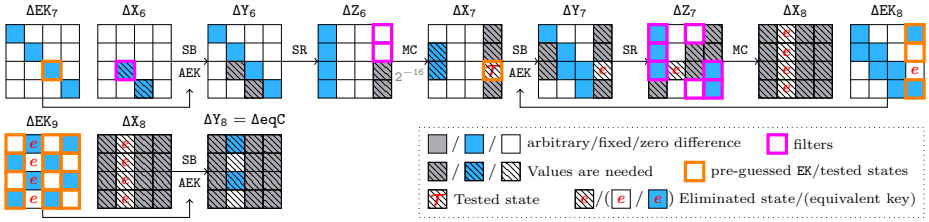

Figure 11: A toy example of the key pre-guessing strategy in the related key setting

Since eliminated states always exhibit known differences, they are independent of the filter collection process, and their corresponding subkeys are consequently excluded from guessing. For example, in Figure 11, testing $\mathtt{X}_7^{14}$ together with the determined $\mathtt{X}_7^{\{12,13,15\}}$ obtained from key pre-guessing, we are able to determine the values before MC, *i.e.*, $\mathtt{Z}_6^{\{12,\ldots,15\}}$. The procedures for determination propagation and obtaining filters follow the established methodology in [44] and are not reiterated here. Finally, in line 11, the costs of tested states and subkey pre-guessing are considered jointly.

B.2 Model for Cost-Free Key Derivation in ϵ Calculation on Ciphers with Non-Full Key Addition

To illustrate cost-free key derivation in ϵ calculation for ciphers with non-full key addition, we use SKINNY as an example and present Algorithm 5 (see also Figure 12). In the algorithm, for $\mathtt{V} \in \{\mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{W}\}$, we introduce several variables in addition to those in Table 5:

- $\mathtt{rdV} \in \{0, 1\}$ indicates whether the corresponding difference is involved in the key-recovery phase ($\mathtt{rdV} = 1$) or not ($\mathtt{rdV} = 0$).
- $\mathtt{sdV} \in \{-1, \ldots, \mathtt{S}\}$ track the step propagation of the difference.
- $\mathtt{sDdV}, \mathtt{sDvV}, \mathtt{sDvSTK} \in \{-1, \ldots, \mathtt{S}\}$ represent the differences, values, and keys that can be deduced at the corresponding step.

The nonfull key addition of SKINNY allows the difference after MC to be deduced without any cost when certain conditions are met, where $\mathtt{c} \in 0, 1, 2, 3$. As illustrated in Figure 12, the first condition is $\Delta\mathtt{X}_\mathtt{r}^\mathtt{c} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12} \leftarrow \mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+12} = 0$, which follows directly from the linear relation of $\mathtt{MC}^{-1}$: $\Delta\mathtt{W}_{\mathtt{r}-1}^{\mathtt{c}+12} = \Delta\mathtt{X}_\mathtt{r}^\mathtt{c} \oplus \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12}$. The remaining conditions can be derived in a similar manner.

$$
\begin{cases}
\Delta\mathtt{X}_\mathtt{r}^\mathtt{c} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12} \leftarrow \mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+12} = 0 \\
\Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+4} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12} \leftarrow \mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+8} = 0 \\
\Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+8} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12} \leftarrow (\mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+4} = 0) \wedge (\mathtt{dX}_\mathtt{r}^{\mathtt{c}+4} = 0) \\
\Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+4} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+12} \leftarrow (\mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+4} = 0) \wedge (\mathtt{dX}_\mathtt{r}^{\mathtt{c}+8} = 0) \\
\Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+4} = \Delta\mathtt{X}_\mathtt{r}^{\mathtt{c}+8} \leftarrow (\mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c}+4} = 0) \wedge (\mathtt{dX}_\mathtt{r}^{\mathtt{c}+12} = 0)
\end{cases}
$$

Algorithm 4: Model for key guessing strategy allowing the state test aligning with Figure 11

Input: variables in Table 5.

1 **for** $\mathtt{c} \in \{0..15\}$ **do** $\mathtt{detY}_8^{\mathtt{c}} = 1$

2 **for** $\mathtt{r} \in \{6,\ldots,8\}, \mathtt{c} \in \{0,...,15\}$ **do**

3 *// determination propagation (SB, SR, AEqk), filters from SB*

4 $\begin{cases} \mathtt{detX}_{\mathtt{r}}^{\mathtt{c}} = 1 \wedge \mathtt{frSB}_{\mathtt{r}}^{\mathtt{c}} = \mathtt{dX}_{\mathtt{r}}^{\mathtt{c}} - \mathtt{dY}_{\mathtt{r}}^{\mathtt{c}} & \leftarrow (\mathtt{detY}_{\mathtt{r}}^{\mathtt{c}} = 1 \wedge \mathtt{gEK}_{\mathtt{r}}^{\mathtt{c}} = 1) \vee \mathtt{tX}_{\mathtt{r}}^{\mathtt{c}} = 1 \\ \mathtt{detX}_{\mathtt{r}}^{\mathtt{c}} = 0 \wedge \mathtt{frSB}_{\mathtt{r}}^{\mathtt{c}} = 0 & \leftarrow \mathtt{otherwise} \end{cases}$

5 $\mathtt{detY}_{\mathtt{r}}^{\mathtt{SR(c)}} = \mathtt{detZ}_{\mathtt{r}}^{\mathtt{c}}$

6 **for** $\mathtt{r} \in \{6,\ldots,8\}, \mathtt{col} \in \{0,4,8,12\}, \mathtt{i} \in \{0,...,3\}$ **do**

7 *// determination propagation (MC), filters from MC*

8 $\mathtt{let} \left[\begin{array}{l} \mathtt{mdetX}_{\mathtt{r}}^{\mathtt{col}} = \min_{\mathtt{i}\in\{0,\ldots,3\}} (\mathtt{detX}_{\mathtt{r}}^{\mathtt{col+i}}); \\ \mathtt{AdetX}_{\mathtt{r}}^{\mathtt{col}} = \bigwedge_{\mathtt{i}\in\{0,\ldots,3\}} (\mathtt{detX}_{\mathtt{r}}^{\mathtt{col+i}} = 1 \vee \mathtt{dX}_{\mathtt{r}}^{\mathtt{col+i}} \leq 1); \\ \mathtt{EdX}_{\mathtt{r}}^{\mathtt{col}} = \exists_{\mathtt{i}\in\{0,\ldots,3\}} (\mathtt{dX}_{\mathtt{r}}^{\mathtt{col+i}} \geq 1). \end{array} \right]$

9 $\begin{cases} \left(\begin{array}{c} \mathtt{detZ}_{\mathtt{r}-1}^{\mathtt{col+i}} = \mathtt{mdetX}_{\mathtt{r}}^{\mathtt{col}} \wedge \\ \mathtt{frMC}_{\mathtt{r}}^{\mathtt{col+i}} = (\mathtt{dZ}_{\mathtt{r}-1}^{\mathtt{col+i}} \leq 1) \times \mathtt{EdW}_{\mathtt{r}}^{\mathtt{col}} \end{array} \right) & \leftarrow \mathtt{AdetX}_{\mathtt{r}}^{\mathtt{col}} = 1 \\ \mathtt{frMC}_{\mathtt{r}}^{\mathtt{c+i}} = 0 \wedge \mathtt{detZ}_{\mathtt{r}-1}^{\mathtt{c+i}} = 0 & \leftarrow \mathtt{AdetX}_{\mathtt{r}}^{\mathtt{col}} = 0 \end{cases}$

10 *// count pre-guessed subkeys and tested states in E_f*

11 $\mathtt{m}'_{\mathtt{f}} = \sum_{\mathtt{r}\in\{6,\ldots,8\}}^{\mathtt{c}\in\{0,\ldots,15\}} (\mathtt{gEK}_{\mathtt{r}}^{\mathtt{c}} + 2 \times (\mathtt{tX}_{\mathtt{r}}^{\mathtt{c}} = 1))$

12 *// count filters from SB and MC in E_f*

13 $\mathtt{FR}_{\mathtt{f}} = \sum_{\mathtt{r}\in\{6,\ldots,8\}}^{\mathtt{c}\in\{0,\ldots,15\}} (\mathtt{frSB}_{\mathtt{r}}^{\mathtt{c}} + \mathtt{frMC}_{\mathtt{r}}^{\mathtt{c}})$

In Algorithm 5, we first inherit the propagation steps for differentials, values, and keys from the preceding step-propagation process, as specified in lines $1 \sim 3$ and $8 \sim 11$. To reserve assignment space for `sDdV`, `sDvV`, and `sDvSTK`, states and keys are permitted to remain unassigned during value and differential propagation, say assign the same step as uninvolved state, *i.e.*, -1. Lines $4 \sim 7$ model one of the conditions of the difference derivation given above, namely $\mathtt{dX}_{\mathtt{r}}^{\mathtt{c}} = \mathtt{dX}_{\mathtt{r}}^{\mathtt{c+12}} \leftarrow \mathtt{dW}_{\mathtt{r}-1}^{\mathtt{c+12}} = 0$. For example, in Figure 12, we have $\mathtt{dW}_8^{13} = 0$ and $\Delta\mathtt{X}_9^{13}$ determined at step 1 (*i.e.*, $\mathtt{sdX}_9^{13} = 1$). Because $\Delta\mathtt{W}_8^{13} = \Delta\mathtt{X}_9^{1} \oplus \Delta\mathtt{X}_9^{13}$, the difference $\Delta\mathtt{X}_9^{1}$ can also be determined at step 1 without additional cost, yielding $\mathtt{sdX}_9^{1} = 1$. Line 13 constrains that the value of a cell after `SB` can be derived

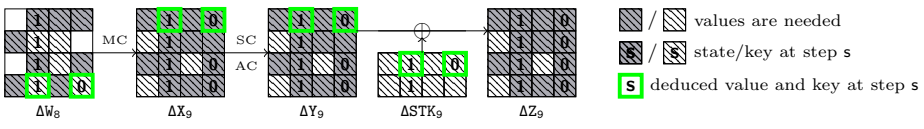

Figure 12: A toy example of cost-free key derivation in ϵ calculation on `SKINNY`

if the cell is unassigned before and its differences before and after SB are both determined. Subsequently, line 15 permits the value of the corresponding key to be derived at the greater step between Y and Z. As shown in Figure 12, since $\mathtt{sDdX}_9^1 = 1$ and $\mathtt{sDvY}_9^1 = \mathtt{sdY}_9^1 = 1$, the value $\mathtt{Y}_9^1$ can be deduced at step 1 ($\mathtt{sDvY}_9^1 = 1$), after which the value $\mathtt{STK}_9^1$ is obtained, *i.e.*, $\mathtt{sDvSTK}_9^1 = 1$. Finally, we propagate the deduced values, keys, and differential following the same logic that is given in Algorithm 2.

B.3 Model for Complexity Evaluation for ϵ Calculation

Algorithm 6 accepts a Boolean variable $\mathtt{Tag} \in \{0, 1\}$ to select the evaluation method: $\mathtt{Tag} = 0$ corresponds to the guess-and-determine approach, while $\mathtt{Tag} = 1$ indicates the use of precomputed hash tables. We introduce $\mathtt{MSB}_\epsilon^{\mathtt{s}}$ to denote the memory consumption associated with SB. As specified in line 3, only those

Algorithm 5: Model for cost-free key derivation in ϵ calculation on SKINNY aligning with Figure 12.

Input: variables in Table 5; maximum number of steps S; model of step propagation relevant with value and difference in ϵ calculation.

1 **for** $\mathtt{V} \in \{\mathtt{W}_8, \mathtt{X}_9, \mathtt{Y}_9, \mathtt{Z}_9\}, \mathtt{c} \in \{0, \ldots, 15\}$ **do**

2 *// inherit step from preceding differential propagation*

3 $\begin{cases} \mathtt{sDdV}^{\mathtt{c}} = -1 & \leftarrow \mathtt{rdV}^{\mathtt{c}} = 0 \\ \mathtt{sDdV}^{\mathtt{c}} = \min(\mathtt{sV}^{\mathtt{c}}, \mathtt{sdV}^{\mathtt{c}}) & \leftarrow (\mathtt{sdV}^{\mathtt{c}} \geq 1) \vee (\mathtt{sV}^{\mathtt{c}} \geq 1) \\ \mathtt{sDdV}^{\mathtt{c}} \in \{0, \ldots, \mathtt{S}\} & \leftarrow \mathtt{otherwise} \end{cases}$

4 **for** $\mathtt{c} \in \{0, \ldots, 3\}$ **do**

5 *// back-track difference derivation for constrain* $(\Delta \mathtt{X}^{\mathtt{c}} = \Delta \mathtt{X}^{\mathtt{c}+12} \leftarrow \Delta \mathtt{W}^{\mathtt{c}+12} = 0)$

6 **if** $(\mathtt{dW}_8^{\mathtt{c}+12} = 0)$ **do**

7 $\begin{cases} \mathtt{sDdX}_9^{\mathtt{c}} = \max\left(\mathtt{sdX}_9^{\mathtt{c}+12}, \mathtt{sDdX}_9^{\mathtt{c}+12}\right) \leftarrow (\mathtt{sDdX}_9^{\mathtt{c}} \notin \{-1, \mathtt{sdX}_9^{\mathtt{c}}\}) \wedge (\mathtt{dX}_9^{\mathtt{c}} = 2) \\ \mathtt{sDdX}_9^{\mathtt{c}+12} = \max\left(\mathtt{sdX}_9^{\mathtt{c}}, \mathtt{sDdX}_9^{\mathtt{c}}\right) \leftarrow \left(\mathtt{sDdX}_9^{\mathtt{c}+12} \notin \{-1, \mathtt{sdX}_9^{\mathtt{c}+12}\}\right) \wedge \left(\mathtt{dX}_9^{\mathtt{c}+12} = 2\right) \end{cases}$

8 **for** $\mathtt{V} \in \{\mathtt{W}_8, \mathtt{X}_9, \mathtt{Y}_9, \mathtt{Z}_9\}, \mathtt{c} \in \{0, \ldots, 15\}$ **do**

9 *// inherit step from preceding value propagation*

10 $\begin{cases} \mathtt{sDvV}^{\mathtt{c}} = -1 & \leftarrow \mathtt{vV}^{\mathtt{c}} = 0 \\ \mathtt{sDvV}^{\mathtt{c}} = \mathtt{sV}^{\mathtt{c}} & \leftarrow \mathtt{sV}^{\mathtt{c}} \geq 1 \\ \mathtt{sDvV}^{\mathtt{c}} \in \{0, \ldots, \mathtt{S}\} & \leftarrow \mathtt{otherwise} \end{cases}$

11 $\begin{cases} \mathtt{sDvSTK}_9^{\mathtt{c}} = -1 & \leftarrow (\mathtt{sSTK}_9^{\mathtt{c}} \geq 0) \vee (\mathtt{vSTK} = 0) \\ \mathtt{sDvSTK}_9^{\mathtt{c}} \in \{0, \ldots, \mathtt{S}\} & \leftarrow \mathtt{otherwise} \end{cases}$

12 *// value derivation*

13 $\mathtt{sDvY}_9^{\mathtt{c}} = \max\left(\mathtt{sDdX}_9^{\mathtt{c}}, \mathtt{sdY}_9^{\mathtt{c}}\right) \leftarrow (\mathtt{sDdX}_9^{\mathtt{c}} \geq 0) \wedge (\mathtt{sDvY}_9^{\mathtt{c}} \geq 0)$

14 *// involved key derivation*

15 $\mathtt{sDvSTK}_9^{\mathtt{c}} = \max\left(\mathtt{sDvY}_9^{\mathtt{c}}, \mathtt{sDvZ}_9^{\mathtt{c}}\right) \leftarrow \mathtt{sDvSTK}_9^{\mathtt{c}} \geq 0$

16 *// propagation of deduced value follow Algorithm 2*

Algorithm 6: Model for complexity evaluation for ϵ calculation aligning with Figure 8

Input: variables inherited form Algorithm 2, $\mathtt{Tag} \in \{0, 1\}$ to switch between two approaches: guess-and-filter and pre-computation hash tables.

for $\mathtt{SB} \in \{1, ..., \mathtt{S}\}$ **do**

  // memory cost corresponding to SB

  $\mathtt{MSB}_{\epsilon}^{\mathtt{s}} = \sum_{\mathtt{r}\in\{6,7,8\}}^{\mathtt{c}\in\{0,...,15\}} ((\mathtt{sY}_{\mathtt{r}}^{\mathtt{c}} < \mathtt{s}) \wedge (\mathtt{sX}_{\mathtt{r}}^{\mathtt{c}} = \mathtt{s}) \wedge (\mathtt{eX}_{\mathtt{r}}^{\mathtt{c}} \neq 2))$

  // memory cost for storing values and differences related to MC

  for $\mathtt{r} \in \{6, 7, 8\}, \mathtt{col} \in \{0, 4, 8, 12\}$ **let**

$$\begin{bmatrix} \mathtt{CondV}_{\mathtt{r}}^{\mathtt{col}} = \left(\max_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}}) = \mathtt{s}\right) \wedge \left(\exists_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} < \mathtt{s})\right); \\ \mathtt{MemoV}_{\mathtt{r}}^{\mathtt{col}} = \min\left(\sum_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} < \mathtt{s}), \sum_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sZ}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} = \mathtt{s})\right); \\ \mathtt{CondD}_{\mathtt{r}}^{\mathtt{col}} = \left(\max_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} \times (\mathtt{dX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} = 2)) = \mathtt{s}\right) \wedge \\ \qquad \left(\bigwedge_{\mathtt{i}\in\{0,...,3\}} (\mathtt{sZ}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} \neq \mathtt{s})\right); \\ \mathtt{MemoD}_{\mathtt{r}}^{\mathtt{col}} = \sum_{\mathtt{i}\in\{0,...,3\}} ((\mathtt{sX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} < \mathtt{s}) \wedge (\mathtt{dX}_{\mathtt{r}}^{\mathtt{col}+\mathtt{i}} = 2)). \end{bmatrix}$$

  $\mathtt{MMCv}_{\epsilon}^{\mathtt{s}} = \sum_{\mathtt{r}\in\{\{6,7,8\}}^{\mathtt{c}\in\{0,4,8,12\}} (\mathtt{CondV}_{\mathtt{r}}^{\mathtt{c}} \times \mathtt{MemoV}_{\mathtt{r}}^{\mathtt{c}})$

  $\mathtt{MMCd}_{\epsilon}^{\mathtt{s}} = \sum_{\mathtt{r}\in\{\{6,7,8\}}^{\mathtt{c}\in\{0,4,8,12\}} (\mathtt{CondD}_{\mathtt{r}}^{\mathtt{c}} \times \mathtt{MemoD}_{\mathtt{r}}^{\mathtt{c}})$

let $\left[\mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} \in \{1, 2\}\right]$ *// table is for pairs or quartets at each step.*

// initialize the time consumption of the first step

$\mathtt{T}_{\epsilon}^{1} = \mathtt{SsEK}_{\epsilon}^{1} - \mathtt{Tag} \times \mathtt{Flag}_{\mathtt{PQ}}^{1} \times (\mathtt{frSB}_{\epsilon}^{1} - \mathtt{frMC}_{\epsilon}^{1})$

for $\mathtt{s} \in \{1, ..., \mathtt{S}\}$ **do**

  // time consumption of steps 1 to S

$$\begin{cases} \mathtt{T}_{\epsilon}^{\mathtt{s}} = \mathtt{T}_{\epsilon}^{\mathtt{s}-1} + \mathtt{sEK}_{\epsilon}^{\mathtt{s}} - \mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} \times (\mathtt{frSB}_{\epsilon}^{\mathtt{s}} + \mathtt{frMC}_{\epsilon}^{\mathtt{s}}) \leftarrow \mathtt{Tag} = 1 \\ \mathtt{T}_{\epsilon}^{\mathtt{s}} = \mathtt{T}_{\epsilon}^{\mathtt{s}-1} - 2 \times (\mathtt{frSB}_{\epsilon}^{\mathtt{s}-1} + \mathtt{frMC}_{\epsilon}^{\mathtt{s}-1}) + \mathtt{SsEK}_{\epsilon}^{\mathtt{s}} \leftarrow \mathtt{Tag} = 0 \end{cases}$$

for $\mathtt{s} \in \{0, ..., \mathtt{S}\}$ **do**

  // memory consumption for pre-computation hash table

  $\mathtt{M}_{\epsilon}^{\mathtt{s}} = \mathtt{SsEK}_{\epsilon}^{\mathtt{s}} + \mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} \times (2 \times (\mathtt{MSB}_{\epsilon}^{\mathtt{s}} + \mathtt{MMCv}_{\epsilon}^{\mathtt{s}}) + \mathtt{MMCd}_{\epsilon}^{\mathtt{s}} - \mathtt{frSB}_{\epsilon}^{\mathtt{s}} - \mathtt{frMC}_{\epsilon}^{\mathtt{s}})$

non-eliminated states after SB and serve as starting points for step propagation need to be stored in hash tables. As shown in Figure 8, at step 4, the values of $\mathtt{X}_7^{\{12,13,15\}}$ require to be stored. We define $\mathtt{MMCv}_{\epsilon}^{\mathtt{s}}$ and $\mathtt{MMCd}_{\epsilon}^{\mathtt{s}}$ as the memory costs for storing values and differences related to MC, respectively. For $\mathtt{MMCv}_{\epsilon}^{\mathtt{s}}$, we collect the values necessary to compute the states before MC. In certain cases, we store values before MC as linear combinations of states after MC, rather than the values after MC determined in the preceding steps, which may reduce memory cost. At step 5 in Figure 8, the values of $\mathtt{X}_7^{\{12,...,15\}}$ are necessary to calculate

Algorithm 7: Key bridging of Deoxys using in ϵ calculation

Input: the variables in Table 5; the subkey permutation table hTable; the tweakey schedule property table JTable (Table 4); redundant key bits for the key-guessing strategy $\mathtt{gRd}^{\mathtt{col}}$, where $\mathtt{col} \in \{0, 4, 8, 12\}$.

1 **for** $\mathtt{s} \in \{1, \ldots, \mathtt{S}\}, \mathtt{col} \in \{0, 4, 8, 12\}$ let

2 *// counting for redundant subkey bits in each column*

3
$$\begin{bmatrix} \mathtt{MsK}^{\mathtt{col}} = \max_{\mathtt{j}\in\{0,\ldots,3\}} \left(\mathtt{sSTK}_0^{\mathtt{hTable}^{-15}(\mathtt{col}+\mathtt{j})}, \mathtt{sEK}_{15}^{\mathtt{SR}(\mathtt{col}+\mathtt{j})}\right) \\ \mathtt{SsSTK}_0^{\mathtt{col}} = \sum_{\mathtt{j}\in\{0,\ldots,3\}} \left(0 \le \mathtt{sSTK}_0^{\mathtt{hTable}^{-15}(\mathtt{col}+\mathtt{j})} \le \mathtt{BigsEK}^{\mathtt{col}}\right) \\ \mathtt{SsEK}_{15}^{\mathtt{col}} = \sum_{\mathtt{j}\in\{0,\ldots,3\}} \left(0 \le \mathtt{sEK}_{15}^{\mathtt{SR}(\mathtt{col}+\mathtt{j})} \le \mathtt{BigsEK}^{\mathtt{col}}\right) \end{bmatrix}$$

4 **for** $\mathtt{s} \in \{1, \ldots, \mathtt{S}\}$ **do**

5 $\quad \mathtt{sRd_s} = \sum_{\mathtt{c}\in\{0,4,8,12\}} (\mathtt{JTable}\,[\mathtt{SsSTK}_0^{\mathtt{col}}, \mathtt{SsEK}_{15}^{\mathtt{col}}]$

6 $\quad\quad\quad\quad\quad\quad \times ((\mathtt{MsK}^{\mathtt{col}} = \mathtt{s}) \wedge (\mathtt{gRd}^{\mathtt{col}} = 0)))$

$\mathtt{Z}_6^{\{14,15\}}$. However, a lower memory cost can be achieved by storing $\mathtt{X}_7^{14}$, the tested state, and $\mathtt{Z}_6^{\{14,15\}}$, the value before MC instead.

In the precomputation hash table approach, the parameter $\mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} \in \{1, 2\}$ indicates whether a table stores pairs ($\mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} = 1$) or quartets ($\mathtt{Flag}_{\mathtt{PQ}}^{\mathtt{s}} = 2$). The variables $\mathtt{T}_\epsilon^{\mathtt{s}}$ and $\mathtt{M}_\epsilon^{\mathtt{s}}$ represent the time and memory complexity at step s, respectively. In line 14, we provide two methods for computing $\mathtt{T}_\epsilon^{\mathtt{s}}$, each corresponding to a different strategy for ϵ calculation. Notably, $\mathtt{MSB}_\epsilon^{\mathtt{s}}$, $\mathtt{MMCv}_\epsilon^{\mathtt{s}}$, and $\mathtt{MMCd}_\epsilon^{\mathtt{s}}$ are included in $\mathtt{M}_\epsilon^{\mathtt{s}}$ simultaneously at each step.

B.4 Model for Using Strong Key Bridging of Deoxys and in the ϵ Calculation

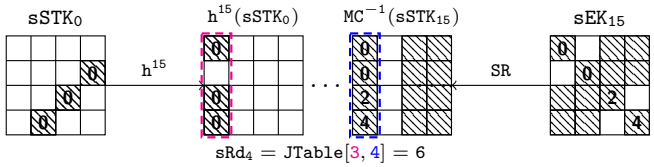

Figure 13: A toy example of counting redundant keys for Deoxys in the ϵ calculation

We present Algorithm 7 and a toy example in Figure 13 to illustrate how the strong key bridging of Deoxys is integrated into ϵ calculation. For cells that lie within the same lane and form the same column, we define three auxiliary variables: the maximum step ($\mathtt{MsK}^{\mathtt{col}}$), available cells in round 0 ($\mathtt{SsSTK}_0^{\mathtt{col}}$), and

Algorithm 8: Strong key bridging of SKINNY-n-3n (Figure 14)

Input: variables in Table 5; the subkey permutation $\mathtt{h_S}$;

1 **for** $\mathtt{c} \in \{0, \ldots, 15\}$ **do**

2 *// counting involved key within the same lane across 30 rounds*

3 $\mathtt{vLstk}_0^{\mathtt{c}} = \min\left(3, \sum_{\mathtt{r}\in\{0,1,2,29\}} \left(\mathtt{vSTK}_{\mathtt{r}}^{\mathtt{h_S(c)}}\right)\right)$

4 $\mathtt{vLstk}_1^{\mathtt{c}} = \min\left(3, \sum_{\mathtt{r}\in\{1,2,29,30\}} \left(\mathtt{vSTK}_{\mathtt{r}}^{\mathtt{h_S(c)}}\right)\right)$

5 $\mathtt{vLstk}_2^{\mathtt{c}} = \min\left(3, \sum_{\mathtt{r}\in\{2,29,30,31\}} \left(\mathtt{vSTK}_{\mathtt{r}}^{\mathtt{h_S(c)}}\right)\right)$

6 *// the sum of the number of involved keys*

7 $\mathtt{m_b} + \mathtt{m_f} = \sum_{\mathtt{c}\in\{0,\ldots,15\}} \left(\max_{\mathtt{r}\in\{0,1,2\}} \left(\mathtt{vLstk}_{\mathtt{r}}^{\mathtt{c}}\right)\right)$

available cells in round 15 ($\mathtt{SsEK}_{15}^{\mathtt{col}}$). Redundant keys are collected only at the maximum step assigned to such columns, which maximizes the benefit of strong key bridging and prevents multiple counting. To further ensure that redundant key bits are counted only once, line 5 references the variable $\mathtt{gRd}^{\mathtt{col}}$, which counts the redundant key bits in the key-guessing strategy. If $\mathtt{gRd}^0 = 0$, the 6-bit redundant key is collected only at step 4 as shown in Figure 13; otherwise $\mathtt{gRd}^0 = \mathtt{JTable}[3, 2] = 1$ and $\mathtt{sRd}_4 = 0$.

B.5 Model the Subkey Relation for SKINNY-n-3n.

The key bridging for SKINNY-n-3n adheres to the conditions established in Propositions 2 and 5. Key information repeats between subkeys with a period of 30 rounds, thereby establishing the strong key bridging of SKINNY-n-3n. Consequently, collecting key information over 30 rounds is sufficient. To integrate this strong key bridging into our full-fledged model, we present Algorithm 8, which aligns with Figure 14 and implements the idea of *slide windows* to count the number of relevant key bits.

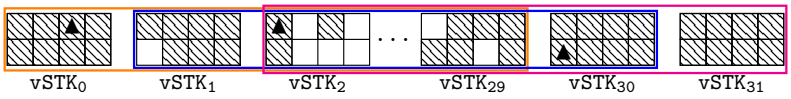

Figure 14: A toy example of counting the involved key bits for SKINNY-n-3n

In lines 2 ∼ 5 of Algorithm 8, the variables vLstk denote the involved keys that remain within the same lane across 30 rounds. These correspond to vSTK highlighted by the yellow, blue, and purple slide windows in Figure 14, respectively. As a result of our stronger key bridging, the total number of involved keys is significantly reduced. In Figure 14, for $\mathtt{vSTK}_0^2, \mathtt{vSTK}_2^0$, and $\mathtt{vSTK}_{30}^4$ in the same

lane, there contains only 2 of them in each slide window, compared to 3 under the classical approach.

Algorithm 8 can be readily adapted for both the key-guessing strategy and the ϵ calculation. In the key guessing strategy, the variables for involved keys are replaced with those for guessed keys; in the ϵ calculation, the algorithm is used to count the keys processed at each step.

C Other Attacks

C.1 Improved 11-Round Attack on Deoxys-BC-256

The longest known distinguisher on Deoxys-BC-256 spans 9 rounds [10], with a probability of $P^2 = 2^{-120.4}$ improved using the BDT technique [45]. To enhance the flexibility of our attack, as described in Section 4, we employ an 8-round boomerang distinguisher. In this attack, the full-fledged model is used for finding the optimal attack parameters. The probability of the whole attack of $P^2 = 2^{-106}$, as detailed in Table 13. The whole attack is as shown in Figure 15, where E_b covers 1 round and E_f covers 2 rounds.

Data complexity. The probability of the whole attack is $P^2 = 2^{-106}$, and other parameters are $r_b = 4\times 8 = 32, r_f = 12\times 8 = 96, m_b = 4\times 8 = 32, m_f = 18\times 8 = 144$. The data complexity is $D_R = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{119}$.

Time complexity. The best key-guessing strategy is: $m'_b = m_b = 4 \times 8 = 32$, $m'_f = 5 \times 8 = 40$. Therefore, the filters obtained are $r'_b = 4 \times 8 = 32$, $r'_f = 4 \times 8 = 32$, and $r^*_b = 0$, $r^*_f = 64$. We get $\epsilon = 1$ table look-up, calculated by employing the pre-computed hash table, detailed in Table 8. The number of steps we set to calculate ϵ is 5, while 4 steps are required only to minimize the time and memory complexity of accessing the hash tables. The time complexity of our attack is as follows:

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{119},\\
T_1 &= 2^{m'_b+m'_f} \cdot D_R \cdot \frac{2}{11} = \sqrt{s} \cdot 2^{188.5},\\
T_2 &= 2^{m'_b+m'_f} \cdot D \cdot 2^{r^*_b} = \sqrt{s} \cdot 2^{189},\\
T_3 &= 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{178},\\
T_4 &= 2^{k-\ell}.
\end{aligned}
$$

Memory complexity. The memory complexity of our attack is $M_R = D_R + D \cdot 2^{r^*_b} + 2^{m_b+m_f-m'_b-m'_f} + M_\epsilon = D_R$.

We set $s = 1$ and $\ell = 68$. Then, the data, memory, and time complexities of our attack are 2^{119}, 2^{119}, and $2^{189.8}$, respectively. The success probability is about 84.0%.

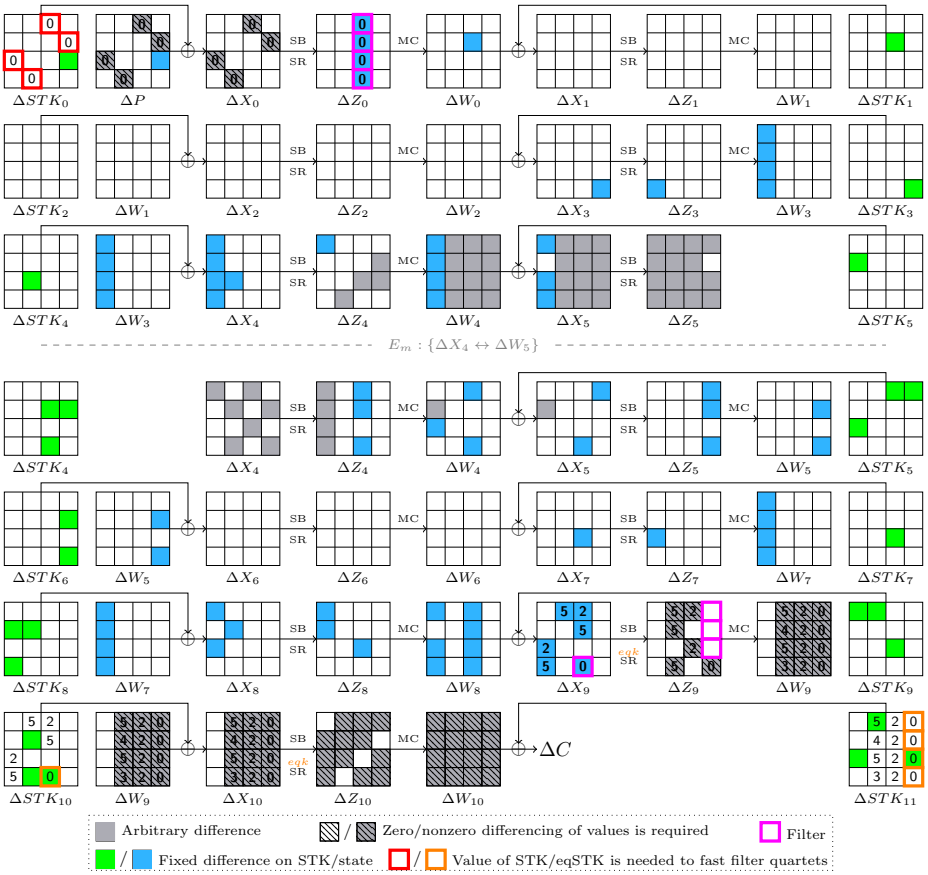

Figure 15: Rectangle attack on 11-round `Deoxys-BC-256`

Table 8: Pre-computed hash tables for the 11-round rectangle attack on `Deoxys-BC-256`, where $\bar{i}$ denotes the table at step i is built for pairs, $i \in \{2 \sim 5\}$.

Step	Starting bytes	Involved subtweakey	Filter	Memory	Time
2	$Z_{10}[2,5,8,15]$	$eqSTK_{10}[2,8]$, $eqSTK_{11}[8,9,10,11]$	$\Delta X_9[2,8]$, $\Delta Z_9[9,10]$	$2^{14\times 8}$	$2^{-2\times 8}$
$\bar{3}$	$Z_{10}[11]$	$eqSTK_{11}[7]$	-	$2^{3\times 8}$	$2^{-1\times 8}$
$\bar{4}$	$Z_{10}[1]$	$eqSTK_{11}[5]$	-	$2^{3\times 8}$	1
5	$W_9[5,7]$, $Z_{10}[4,14]$	$eqSTK_{10}[3,4,9]$, $eqSTK_{11}[4,6]$	$\Delta X_9[3,4,9]$, $\Delta Z_9[6]$	$2^{13\times 8}$	$2^{-3\times 8}$

C.2 33-Round Attack on `SKINNY-64-192`

Setting for the middle part. For `Deoxys`-BC, a 2-round E_m suffices to handle the dependency between the upper and the lower trails of the boomerang distinguisher. However, for `SKINNY` the dependency can span as many as six middle

rounds, as observed in [42]. Given that E_m covers a large number of rounds, accurately computing its probability is challenging. A common approach is to estimate the probability of E_m empirically, for example, the experimental treatments in [14, 18]. To demonstrate the full capabilities of our model in key recovery attacks, we therefore search for optimal attacks on the middle part of the boomerang distinguishers from [14].

For SKINNY-64-192, our model shows that a 33-round attack is possible, as illustrated in Figure 16. The boomerang distinguisher used in the attack is displayed in Table 17 and has probability $P^2 = 2^{-57.56}$ with the clustering effect considered. As can be seen in Figure 16, both the plaintext and the ciphertext are fully active, so $r_b = r_f = 64$; k_b contains 19 cells while k_f has 40 cells. However, $|k_b \cup k_f| = 160$ and $|k_b \cap k_f| = 76$, as $k_b \subset k_f$.

Data complexity The probability of the whole attack is $Pr = P^2 = 2^{-57.56}$, so the data complexity is $D_R = 4D = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{62.78}$.

Time complexity The best strategy is to guess 46 subtweakey cells in advance, as marked in red squares in Figure 16. There are only $|k'_b \cup k'_f| = 108$ information bits due to the strong key bridge. Under this key guessing strategy, $r'_b = r_b, r'_f = 28$, and $r^*_b = 0, r^*_f = 36$. The time complexity of the attack is as follows:

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{62.78}, \\
T_1 &= 2^{|k'_b \cup k'_f|} \cdot D_R = \sqrt{s} \cdot 2^{108+62.78} = \sqrt{s} \cdot 2^{170.78}, \\
T_2 &= 2^{|k'_b \cup k'_f|} \cdot D = \sqrt{s} \cdot 2^{108+60.78} = \sqrt{s} \cdot 2^{168.78}, \\
T_3 &= 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r^*_f - 2n} \cdot \epsilon = s \cdot 2^{108+60.78\times 2+36\times 2-128} \cdot \epsilon = s \cdot 2^{173.56} \cdot \epsilon, \\
T_4 &= 2^{k-\ell}.
\end{aligned}
$$

To extract the other subtweakey cells, it takes five steps, and the complexity ϵ is 1. For convenience, we denote the number of quartets remaining as Q, and the detailed computation of ϵ proceeds as follows.

1. Derive $STK_{29}[1,3]$. After guessing k'_b, k'_f, $\Delta X_{29}[13,15]$ is known. Since $\Delta W_{28}[13] = \Delta X_{29}[1] \oplus \Delta X_{29}[13]$ and $\Delta W_{28}[15] = \Delta X_{29}[3] \oplus \Delta X_{29}[15]$, we can determine $\Delta X_{29}[1,3]$. Therefore, we can derive $Y_{29}[1,3]$ from the known input and output differences of the two S-boxes. Then $STK_{29}[1,3] = Y_{29}[1,3] \oplus Z_{29}[1,3]$. As the candidates for $STK_{29}[1,3]$ suggested by the two pairs in a quartet should coincide, there is a $2c$-bit filter and thus $Q \cdot 2^{-2c}$ quartets remain.
2. Guess $STK_{29}[0]$, then we can determine $\Delta X_{28}[13]$. Note that the differential transition from ΔW_{27} to ΔX_{28} implies $\Delta X_{28}[13] = \Delta X_{28}[9] = \Delta X_{28}[1]$, $\Delta X_{28}[13] = \Delta X_{28}[9]$ acts as a $2c$-bit filter, and we can also determine $\Delta X_{28}[1]$. Similar to the first step, we can derive $STK_{28}[1]$. Two derived $STK_{28}[1]$ values should be the same, which is a c-bit filter. The time complexity of this step is $Q \cdot 2^{-c}$ and $Q \cdot 2^{-4c}$ quartets remain.

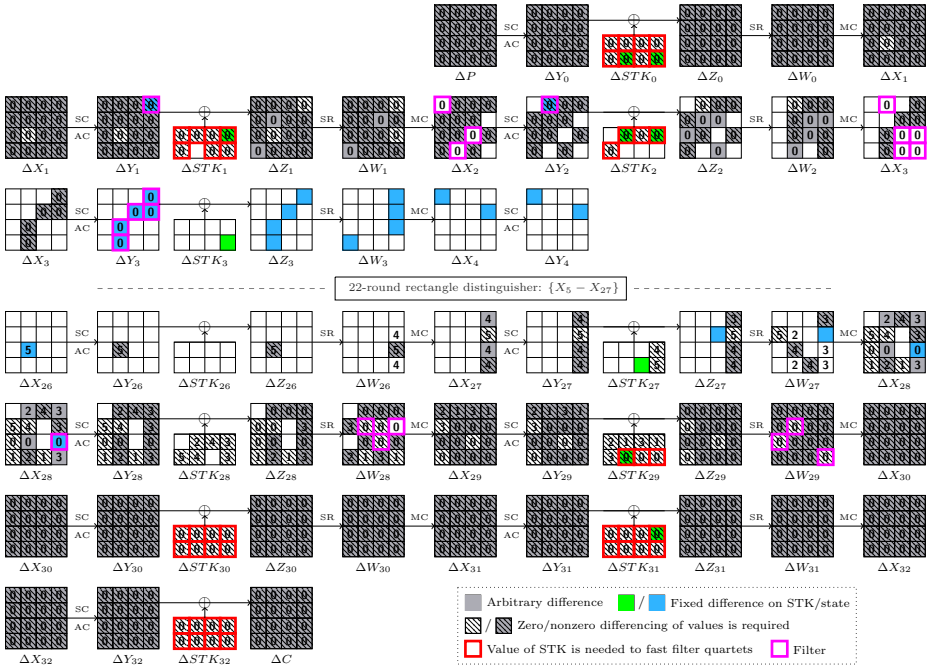

Figure 16: Rectangle attack on 33-round `SKINNY-64-192`

3. Guess $STK_{29}[2,4]$, and then compute $\Delta X_{28}[15]$. Note that $\Delta X_{28}[15] = \Delta X_{28}[3] = \Delta X_{28}[7]$ should hold. Thus, we can determine $\Delta X_{28}[3,7]$. From the differences before and after the S-boxes, derive $Y_{28}[3,7]$ and $STK_{28}[3,7]$. Again, the values of $STK_{28}[3,7]$ suggested by the two pairs should be the same, which is a $2c$-bit filter. The time complexity of this step is $Q \cdot 2^{-2c}$ and $Q \cdot 2^{-4c}$ quartets remain.
4. Guess $STK_{28}[2]$, compute $\Delta X_{27}[15]$. And $\Delta X_{27}[15] = \Delta X_{27}[3] = \Delta X_{27}[11]$ should hold. Likewise, derive $STK_{27}[3]$ and $Z_{27}[11]$, and further derive $STK_{28}[5]$ from $Z_{27}[11]$ and other known state cells. The derived values for the two derived subtweakey cells should coincide, meaning a $2c$-bit filter. The time complexity of this step is $Q \cdot 2^{-3c}$ and $Q \cdot 2^{-5c}$ quartets remain.
5. Guess $STK_{28}[4]$ and $STK_{27}[7]$, compute $\Delta X_{26}[9]$ and check if it equals the specific value. This is a $2c$-bit filter. The time complexity of this step is $Q \cdot 2^{-3c}$ and $Q \cdot 2^{-5c}$ quartets remain.

We choose $s = 4$ and $\ell = 24$, we obtain a 33-round attack on `SKINNY-64-192`, whose data, memory and time complexities are $2^{63.78}$, $2^{63.78}$, and $2^{175.56}$, respectively. The probability of success is about 92.3%.

Remark. If we use the same 22-round boomerang distinguisher in [14], extend the same number of rounds, and choose the same key guessing strategy, we can obtain a similar rectangle attack on 31-round `SKINNY-64-192` with the same data and memory complexities but a lower time complexity due to the strong

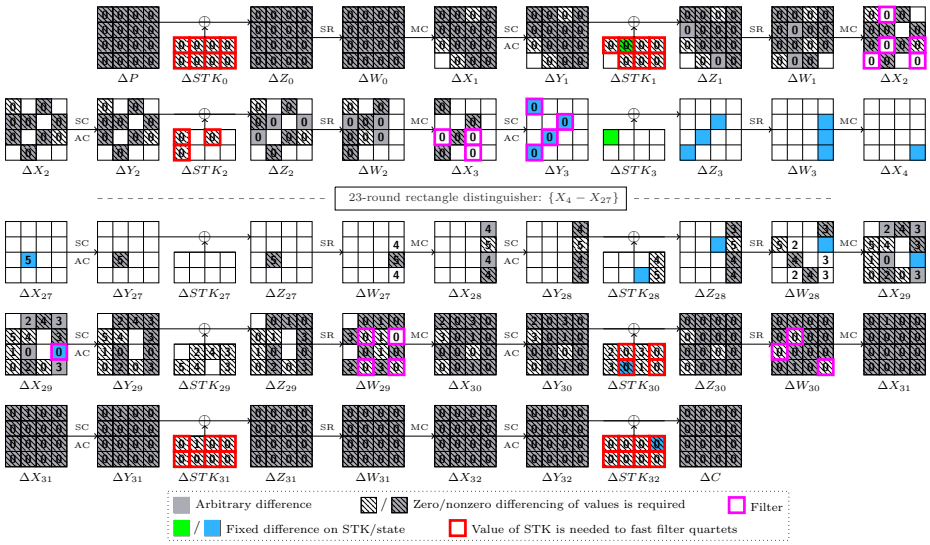

Figure 17: Rectangle attack on 33-round SKINNY-128-384

key bridges. Specifically, the data, memory, and time complexities of our attack are $2^{62.78}$, $2^{62.79}$, and $2^{154.07}$, respectively. The success probability of the attack is 66.6%. If we append one round to the 31-round attack and guess the same subtweakey bits as in the 31-round attack together with the whole additional subtweakey, we have $|k_b \cup k_f| = 144$ and $|k'_b \cup k'_f| = 96$. This means the extra round only brings 4-bit new key information. Thus, we can have a 32-round attack with time complexity increased by a factor of 2^4, *i.e.*, the data, memory, and time complexities of the attack are $2^{62.78}$, $2^{62.79}$, and $2^{158.07}$, respectively.

C.3 33-Round Attack on SKINNY-128-384

For SKINNY-128-384, we obtain a 33-round attack using our model, as shown in Figure 17.

Data complexity. The probability of the boomerang distinguisher in the attack is $2^{-115.09}$. The parameters for this attack are: $r_b = 12 \times 8, r_f = 16 \times 8, m_b = 18 \times 8$ and $m_f = 32 \times 8$. It is noted that $|k_b \cup k_f| = 308$ and $|k_b \cap k_f| = 92$. The data complexity of our attack is $D_R = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{123.54}$.

Time complexity. The best attack strategy is $|k'_b| = 18 \times 8, |k'_f| = 19 \times 8, |k'_b \cup k'_f| = 236$, $|k'_b \cap k'_f| = 60$, $r^*_b = 0, r^*_f = 8 \times 8$, as marked with red squares in Figure 17. As detailed in Table 9, we obtain an $\epsilon = 1$ table look-up, which is calculated using the pre-computed hash table. The time complexity of our attack is as follows.

$T_1 = 2^{|k_b' \cup k_f'|} \cdot D_R = \sqrt{s} \cdot 2^{236+123.54} = \sqrt{s} \cdot 2^{359.54}$,

$T_2 = 2^{|k_b' \cup k_f'|} \cdot D \cdot 2^{r_b^*} = \sqrt{s} \cdot 2^{336+121.54} = \sqrt{s} \cdot 2^{357.54}$,

$T_3 = 2^{|k_b' \cup k_f'|} \cdot D^2 \cdot 2^{2r_b^*+2r_f^*-2n} \cdot \epsilon = s \cdot 2^{236+243.09+2\times64-2\times128} \cdot \epsilon = s \cdot 2^{351.09} \cdot \epsilon$,

$T_4 = 2^{384-\ell}$.

Memory complexity. The memory complexity is $M_R = D_R + D \cdot 2^{r_b^*} + 2^{|k_b \cup k_f| - |k_b' \cup k_f'|} = \sqrt{s} \cdot 2^{123.54} + \sqrt{s} \cdot 2^{121.54} + 2^{72}$.

If we set $s = 1, \ell = 40$, then the data, memory, and time complexities of our attack are $2^{123.54}, 2^{123.54}, 2^{359.54}$ respectively. The success probability is about 82.1%.

Table 9: Pre-compute hash table for 33-round rectangle attack on SKINNY-128-384, where $\bar{i}$ denotes the table at step i is built for pairs, $i \in \{1 \sim 5\}$.

Step	Starting bytes	Involved subtweakey	Filter	Memory	Time
1	$Z_{31}[1], Z_{30}[6]$, $X_{31}[13]$	$STK_{31}[1]$, $STK_{30}[6]$	$\Delta W_{29}[10]$	2^{96}	2^{-8}
2	$Z_{30}[0], Z_{29}[1]$, $X_{30}[12], \Delta X_{29}[9]$	$STK_{30}[0]$, $STK_{29}[1]$	$\Delta W_{28}[5, 13]$	2^{80}	$2^{-2\times8-16}$
$\bar{3}$	$Z_{30}[2, 4], X_{30}[14]$, $X_{30}[8] \oplus X_{30}[12]$, $Z_{29}[3]$	$STK_{30}[2, 4]$, $STK_{29}[3, 7]$	$\Delta W_{28}[11, 15]$	2^{96}	$2^{-2\times8-24}$
4	$Z_{29}[2, 5], Z_{28}[3]$, $X_{29}[13, 14]$	$STK_{29}[2, 5], STK_{28}[3]$	$\Delta W_{27}[7, 15]$	2^{120}	$2^{-3\times8-28}$
5	$Z_{29}[4], X_{28}[15]$, $X_{29}[8] \oplus X_{29}[12]$	$STK_{29}[4], STK_{28}[7]$	$\Delta X_{27}[9]$	2^{64}	$2^{-3\times8-32}$

C.4 Improved Attacks on Deoxys Authenticated Encryption

Improved 10-Round Attack on Deoxys-I-128 Using our model described in Section 4, we obtain the improved attack on Deoxys-I-128, which extends the known best attack [10] by one round, as shown in Figure 18.

Data complexity. The probability of the whole attack is $Pr = P^2 = 2^{-50}$, and the other parameters of the attack are: $r_b = m_b = 32$, $r_f = 72, m_f = 88$. The data complexity of our attack is $D_R = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{91}$.

Time complexity. The best key guessing parameters are $m_b' = m_b = 32$ and $m_f' = 0$, therefore, $r_b^* = 32$ and $r_f^* = 72$. Using the approach of the pre-computation hash table, with the setting of the maximum step of calculation equal to 5, we obtain $\epsilon = 2^{16}$ table look-ups, as detailed in Table 10. Therefore, the time complexity of our attack is as follows:

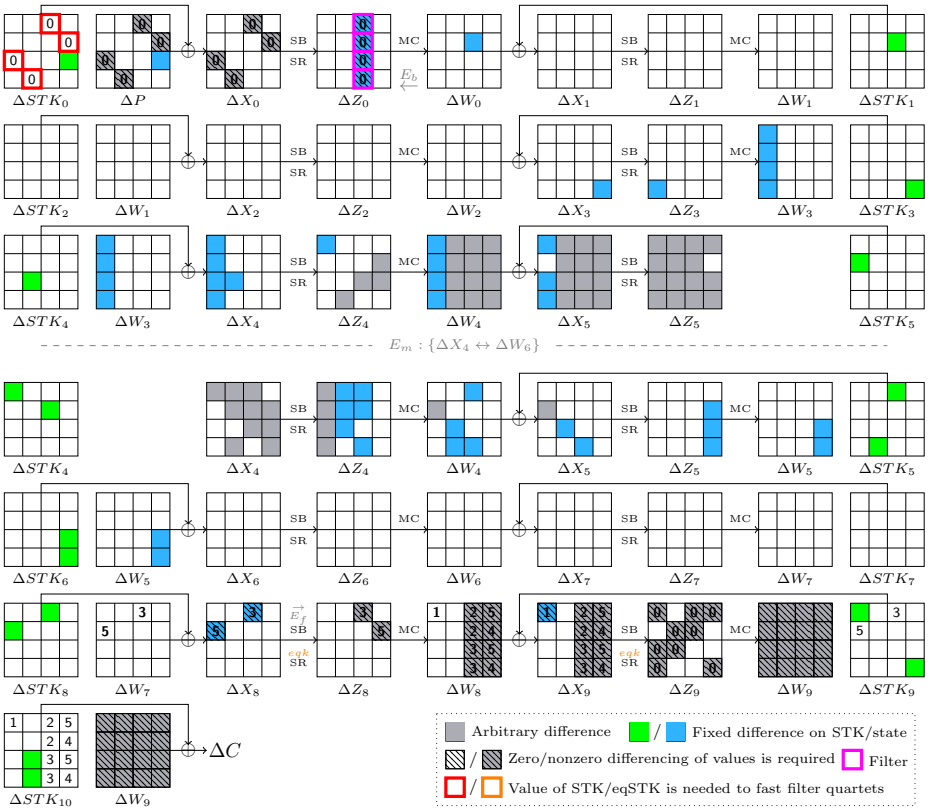

Figure 18: Rectangle attack on 10-round `Deoxys-I-128`

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{91}, \\
T_1 &= 2^{m_b'+m_f'} \cdot D_R \cdot \frac{2}{10} = \sqrt{s} \cdot 2^{120.7}, \\
T_2 &= 2^{m_b'+m_f'} \cdot D \cdot 2^{r_b^*} = \sqrt{s} \cdot 2^{121}, \\
T_3 &= 2^{m_b'+m_f'} \cdot D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{114}, \\
T_4 &= 2^{k-\ell}.
\end{aligned}
$$

Memory complexity. The memory complexity of our attack is $M_R = D_R + D \cdot 2^{r_b^*} + 2^{m_b+m_f-m_b'-m_f'} + M_\epsilon = D_R$.

We set $s = 1$ and $\ell = 16$. Then, the data, memory, and time complexities of our attack are 2^{91}, 2^{91}, and $2^{121.8}$, respectively. The success probability is about 84.1%.

Improved 14-Round Attack on `Deoxys-I-256` Using our full-fledged model described in Section 4, we obtain the improved attack on `Deoxys-I-256`, as

Table 10: Pre-compute hash table for 10-round rectangle attack on `Deoxys-I-128`, where $\bar{i}$ denotes the table at step i is built for pairs, $i \in \{1 \sim 5\}$.

Step	Starting bytes	Involved subtweakey	Filter	Memory	Time
1	$Z_9[0]$	$eqSTK_{10}[0]$	$\Delta X_9[0]$	$2^{3\times 8}$	$2^{-1\times 8}$
$\bar{2}$	$Z_9[5,8]$	$eqSTK_{10}[8,9]$	-	$2^{6\times 8}$	$2^{1\times 8}$
$\bar{3}$	$Z_8[8]$,$Z_9[2,15]$	$eqSTK_9[8]$, $eqSTK_{10}[10,11]$	$\Delta X_8[8]$, $\Delta Z_8[9,10,11]$	$2^{5\times 8}$	1
$\bar{4}$	$Z_9[3,9]$	$eqSTK_{10}[13,15]$	-	$2^{6\times 8}$	$2^{2\times 8}$
$\bar{5}$	$Z_8[13]$, $Z_9[6,12]$	$eqSTK_9[1]$, $eqSTK_{10}[12,14]$	$\Delta X_8[1]$, $\Delta Z_8[12,14,15]$	$2^{5\times 8}$	$2^{1\times 8}$

shown in Figure 19. The 9-round distinguisher is employed in our attack to maintain the flexibility of the entire model, while the longest one covers 10 rounds.

Data complexity. The probability of the whole attack is $Pr = P^2 = 2^{-98.2}$, and the other parameters of the attack are: $r_b = 80$, $m_b = 88$ $r_f = 96$ and $m_f = 152$. The data complexity of our attack is $D_R = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{115.1}$.

Time complexity. The best key guessing parameters are $m'_b = m_b = 88$ and $m'_f = 32$, therefore, $r^*_b = 0$ and $r^*_f = 72$. Using the approach of the pre-computation hash table, with the number of steps set to 5, we obtain $\epsilon = 1$ table look-up, as detailed in Table 11. Therefore, the time complexity of our attack is as follows:

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{115.1}, \\
T_1 &= 2^{m'_b+m'_f} \cdot D_R \cdot \frac{3}{14} = \sqrt{s} \cdot 2^{232.9}, \\
T_2 &= 2^{m'_b+m'_f} \cdot D \cdot 2^{r^*_b} = \sqrt{s} \cdot 2^{233.1}, \\
T_3 &= 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{234.2}, \\
T_4 &= 2^{k-\ell}.
\end{aligned}
$$

Memory complexity. The memory complexity of our attack is $M_R = D_R + D \cdot 2^{r^*_b} + 2^{m_b+m_f-m'_b-m'_f} + M_\epsilon = 2^{m_b+m_f-m'_b-m'_f} + M_\epsilon$.

We set $s = 1$ and $\ell = 32$. Then, the data, memory, and time complexities of our attack are $2^{115.1}$, 2^{121}, and $2^{235.1}$, respectively. The success probability is about 84.1% when accounting for the effect of the state test.

D Boomerang Distinguishers of `Deoxys-BC` and `SKINNY`

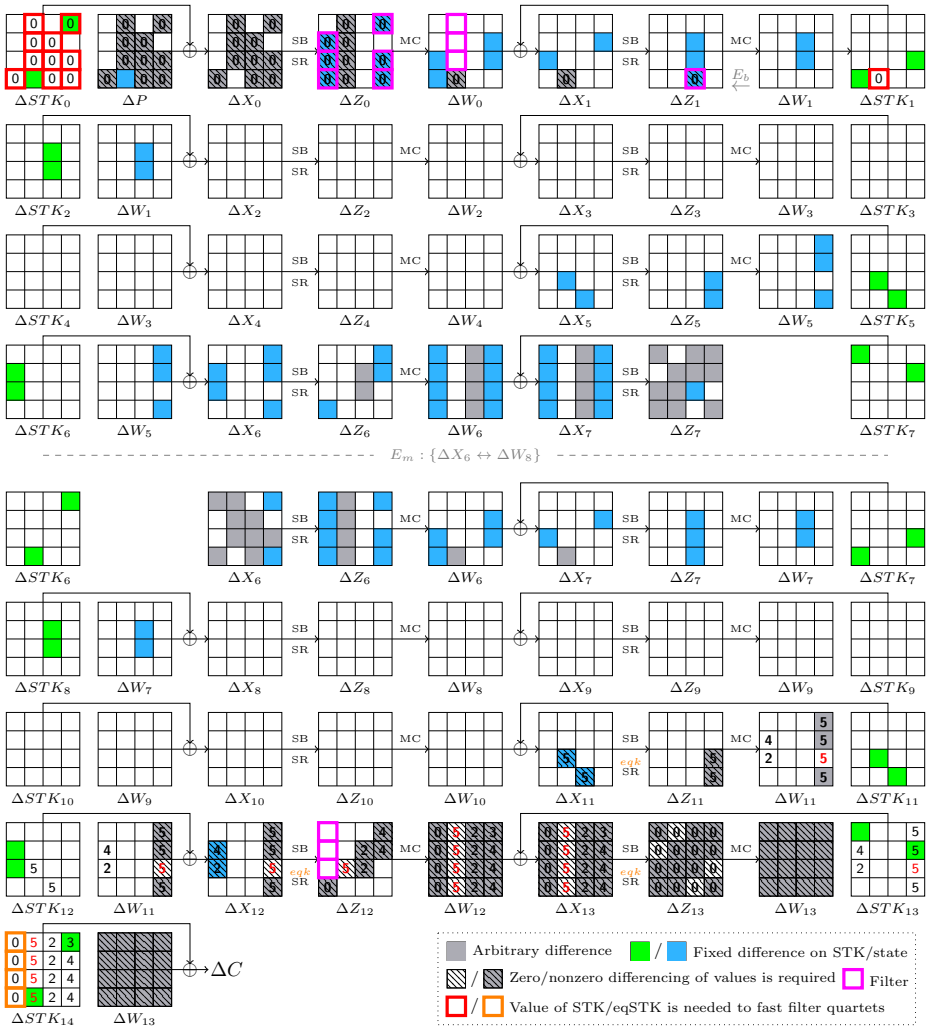

Figure 19: Rectangle attack on 14-round `Deoxys-I-256`

Table 11: Pre-compute hash table for 10-round rectangle attack on `Deoxys-I-256`.

Step	Starting bytes	Involved subtweakey	Filter	Memory	Time
2	$Z_{13}[2,5,8,15]$	$eqSTK_{13}[2]$, $eqSTK_{14}[8,9,10,11]$	$\Delta X_{12}[2]$, $\Delta Z_{12}[8,11]$	$2^{15\times 8}$	$2^{-1\times 8}$
4	$Z_{13}[3,6,9,13]$	$eqSTK_{13}[1]$, $eqSTK_{14}[12,13,14,15]$	$X_{12}[1]$, $Z_{12}[14,15]$	$2^{15\times 8}$	$2^{-2\times 8}$
5	$Z_{12}[3,9,12]$	$eqSTK_{13}[12,13,15]$, $eqSTK_{12}[6,11]$, $W_{11}[14]$	$\Delta X_{11}[6,11]$, $\Delta Z_{11}[14,15]$	$2^{11\times 8}$	$2^{-3\times 8}$

Table 12: A 7-round related-tweakey boomerang distinguisher used to attack 10-round `Deoxys-I`-128-128. The blue cells denote where DDT is employed.

ΔTK_0^1 :	00 00 00 00 00 00 00 00 00 00 00 00 00 00 dd 00					
ΔTK_0^2 :	00 00 00 00 00 00 00 00 00 00 00 00 00 00 $b7$ 00					
Round i	ΔW_{i-1}	ΔSTK_i	ΔX_i	ΔY_i	ΔZ_i	Pr
2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 66	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 66	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 50	00 00 00 00 00 00 00 00 00 00 00 00 50 00 00 00	2^{-12}
4	50 00 00 00 50 00 00 00 $f0$ 00 00 00 $a0$ 00 00 00	00 00 00 00 00 00 00 00 00 ab 00 00 00 00 00 00	50 00 00 00 50 00 00 00 $f0$ ab 00 00 $a0$ 00 00 00	30 00 00 00 ** 00 00 00 ** ** 00 00 ** 00 00 00	30 00 00 00 00 00 00 ** 00 00 ** ** 00 ** 00 00	2^{-38}
5	60 ** ** ** 30 ** ** ** 30 ** ** ** 50 ** ** **	00 00 00 00 30 00 00 00 00 00 00 00 00 00 00 00	60 ** ** ** 00 ** ** ** 30 ** ** ** 50 ** ** **	** ** ** ** 00 ** ** ** ** ** ** ** ** ** ** **	** ** ** ** ** ** ** 00 ** ** ** ** ** ** ** **	
∇TK_0^1 :	00 $1d$ 00 00 00 00 00 00 60 00 00 00 00 00 00 00					
∇TK_0^2 :	00 $f2$ 00 00 00 00 00 00 $6a$ 00 00 00 00 00 00 00					
Round i	∇W_{i-1}	∇STK_i	∇X_i	∇Y_i	∇Z_i	
4	** ** ** 00 00 ** ** ** 00 00 ** ** 00 ** 00 **	cc 00 00 00 00 00 $3e$ 00 00 00 00 00 00 00 00 00	** ** ** 00 00 ** ** ** 00 00 ** ** 00 ae 00 **	** $e1$ $d4$ 00 00 ** $e1$ $3d$ 00 00 ** $e1$ 00 00 00 **	** $e1$ $d4$ 00 ** $e1$ $3d$ 00 ** $e1$ 00 00 ** 00 ae 00	
5	00 00 90 00 ** 00 00 00 00 $d9$ 00 00 00 38 $1d$ 00	00 00 90 00 00 00 00 00 00 00 00 00 00 38 00 00	00 00 00 00 ** 00 00 00 00 $d9$ 00 00 00 00 $1d$ 00	00 00 00 00 ac 00 00 00 00 56 00 00 00 00 $b9$ 00	00 00 00 00 00 00 00 ac 00 00 00 56 00 00 00 $b9$	
6	00 00 00 00 00 00 00 00 00 00 00 $d0$ 00 00 00 93	00 00 00 00 00 00 00 00 00 00 00 $d0$ 00 00 00 93	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1

Table 13: An 8-round related-tweakey boomerang distinguisher used to attack 11-round `Deoxys-BC`-256 in Section C.1. The blue cells denote where DDT is employed.

ΔTK_0^1:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $f9$ 00					
ΔTK_0^2:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 be 00					
Round i	ΔW_{i-1}	ΔSTK_i	ΔX_i	ΔY_i	ΔZ_i	Pr
1	00 00 00 00 00 00 85 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 85 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $0b$	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $0b$	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $0b$	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48	00 00 00 00 00 00 00 00 00 00 00 00 48 00 00 00	2^{-12}
4	48 00 00 00 48 00 00 00 $d8$ 00 00 00 90 00 00 00	00 00 00 00 00 00 00 00 00 $1d$ 00 00 00 00 00 $0b$	48 00 00 00 48 00 00 00 $d8$ $1d$ 00 00 90 00 00 00	31 00 00 00 ** 00 00 00 ** ** 00 00 ** 00 00 00	31 00 00 00 00 00 00 ** 00 00 ** ** 00 ** 00 00	2^{-40}
5	62 ** ** ** 31 ** ** ** 31 ** ** ** 53 ** ** **	00 00 00 00 31 00 00 00 00 00 00 00 00 00 00 00	62 ** ** ** 00 ** ** ** 31 ** ** ** 53 ** ** **	** ** ** ** 00 ** ** ** ** ** ** ** ** ** ** **	** ** ** ** ** ** ** 00 ** ** ** ** ** ** ** **	
∇TK_0^1:	00 ba 00 $6e$ 00 df 00 00 00 00 00 00 00 00 00 00					
∇TK_0^2:	00 $b1$ 00 $c6$ 00 $b3$ 00 00 00 00 00 00 00 00 00 00					
Round i	∇W_{i-1}	∇STK_i	∇X_i	∇Y_i	∇Z_i	
4	** 00 ** 00 00 ** ad ** 00 00 ** 00 00 ** 03 **	00 00 00 00 00 00 ad $e8$ 00 00 00 00 00 00 03 00	** 00 ** 00 00 ** 00 ** 00 00 ** 00 00 $d2$ 00 **	** 00 $f1$ 00 00 ** 00 $9c$ 00 00 ** 00 ** 00 $d2$ 00	** 00 $f1$ 00 ** 00 $9c$ 00 ** 00 00 00 ** 00 $d2$ 00	
5	00 00 94 00 ** 00 00 00 $b5$ 00 00 00 00 00 $2b$ 00	00 00 94 $b0$ 00 00 00 00 $b5$ 00 00 00 00 00 00 00	00 00 00 $b0$ ** 00 00 00 00 00 00 00 00 00 $2b$ 00	00 00 00 $7c$ $c1$ 00 00 00 00 00 00 00 00 00 $1f$ 00	00 00 00 $7c$ 00 00 00 $c1$ 00 00 00 00 00 00 00 $1f$	
6	00 00 00 00 00 00 00 $d9$ 00 00 00 00 00 00 00 $e7$	00 00 00 00 00 00 00 $d9$ 00 00 00 00 00 00 00 $e7$	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
7	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 60 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 60 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 $b3$ 00 00 00 00 00	00 00 00 00 00 00 00 00 $b3$ 00 00 00 00 00 00 00	2^{-12}
8	$b3$ 00 00 00 ce 00 00 00 $7d$ 00 00 00 $b3$ 00 00 00	00 00 00 00 ce $a1$ 00 00 00 00 00 00 $b3$ 00 00 00	$b3$ 00 00 00 00 $a1$ 00 00 $7d$ 00 00 00 00 00 00 00	$3a$ 00 00 00 00 $1d$ 00 00 $e7$ 00 00 00 00 00 00 00	$3a$ 00 00 00 $1d$ 00 00 00 00 00 $e7$ 00 00 00 00 00	2^{-42}

Table 14: A 10-round related-tweakey boomerang distinguisher used to attack 15-round `Deoxys-BC`-384 in Section 5.2. The red cells denote where BCT is employed, and the blue cells denote where DDT is employed.

ΔTK_0^1 :	00 00 00 3c 00 00 00 00 a9 00 8c 00 00 00 00 59					
ΔTK_0^2 :	00 00 00 82 00 00 00 00 53 00 4f 00 00 00 00					
ΔTK_0^3 :	00 00 00 44 00 00 00 00 8f 00 b9 00 00 00 00					
Round i	ΔW_{i-1}	ΔSTK_i	ΔX_i	ΔY_i	ΔZ_i	Pr
1	00 ee 00 00 00 ce 00 00 00 00 9b a9 00 00 00 c8	00 00 00 00 00 ce 00 00 00 00 9b 00 00 00 00 c8	00 ee 00 00 00 00 00 00 00 00 00 a9 00 00 00 00	00 5f 00 00 00 00 00 00 00 00 00 e1 00 00 00 00	00 5f 00 00 00 00 00 00 00 e1 00 00 00 00 00 00	2^{-28}
2	00 5f 00 00 00 67 00 00 00 86 00 00 00 00 00 00	00 5f 00 00 00 67 00 00 00 86 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 5f 67 00 00 00 00 86 00 00	00 00 00 00 00 00 00 5f 67 00 00 00 00 86 00 00	00 00 00 00 00 00 00 $**$ $**$ 00 00 00 00 $**$ 00 00	00 00 00 00 00 00 $**$ 00 00 00 $**$ 00 00 00 $**$ 00	$2^{-25.4}$
6	00 00 $**$ 00 00 00 $**$ 00 00 00 $**$ 00 00 00 $**$ 00	00 00 00 ce 00 00 00 9b 00 00 00 c8 00 00 00 00	00 00 $**$ ce 00 00 $**$ 9b 00 00 $**$ c8 00 00 $**$ 00	00 00 $**$ $**$ 00 00 $**$ $**$ 00 00 $**$ $**$ 00 00 $**$ 00	00 00 $**$ $**$ 00 $**$ $**$ 00 $**$ $**$ 00 00 00 00 00 $**$	
∇TK_0^1 :	00 00 00 00 00 34 c8 fc 00 00 00 00 00 00 00 00					
∇TK_0^2 :	00 00 00 00 00 d4 32 e6 00 00 00 00 00 00 00 00					
∇TK_0^3 :	00 00 00 00 00 53 7b 28 00 00 00 00 00 00 00 00					
Round i	∇W_{i-1}	∇STK_i	∇X_i	∇Y_i	∇Z_i	
5	$**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$	$**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$ $**$	$**$ 00 $**$ $**$ $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 00 $**$ $**$ $**$	$**$ $**$ 00 $**$ $**$ $**$ 00 $**$ 00 $**$ $**$ $**$ $**$ 00 $**$ $**$	$**$ $**$ 00 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 $**$	
6	$**$ 00 00 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 00 85 $**$ 00 $**$	00 00 29 00 ac 00 00 00 00 00 00 00 85 00 00 00	$**$ 00 29 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 00 00 $**$ 00 $**$	fd 00 9c 9f 9f f1 00 77 67 9f 00 00 00 d9 00 04	fd 00 9c 9f f1 00 77 9f 00 00 67 9f 04 00 d9 00	
7	ed 00 00 00 00 00 00 00 00 00 57 25 e5 00 00 ba	ed 00 00 00 00 00 00 00 00 00 57 00 00 00 00 ba	00 00 00 00 00 00 00 00 00 00 00 25 e5 00 00 00	00 00 00 00 00 00 00 00 00 00 00 ca ca 00 00 00	00 00 00 00 00 00 00 00 00 ca 00 00 00 ca 00 00	2^{-28}
8	00 00 00 00 00 8f 00 00 00 ca 00 00 00 45 00 00	00 00 00 00 00 8f 00 00 00 ca 00 00 00 45 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1

Table 15: A 10-round related-tweakey boomerang distinguisher used to attack 14-round `Deoxys-I`-256-128. The red cells denote where BCT is employed, and the blue cells denote where DDT is employed.

ΔTK_0^1 :	00 00 00 00 00 00 00 $d3$ 00 00 00 00 30 00 00 00					
ΔTK_0^2 :	00 00 00 00 00 00 00 $f6$ 00 00 00 00 02 00 00 00					
ΔTK_0^3 :	00 00 00 00 00 00 00 17 00 00 00 00 05 00 00 00					
Round i	ΔW_{i-1}	ΔSTK_i	ΔX_i	ΔY_i	ΔZ_i	Pr
2	00 00 00 00 00 00 ce 00 00 00 79 00 00 00 00 00	00 00 00 00 00 00 ce 00 00 00 79 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 ce 00 00 00 00 79 00	00 00 00 00 00 00 00 00 00 ce 00 00 00 00 79 00	00 00 00 00 00 00 00 00 00 ea 00 00 00 00 45 00	00 00 00 00 00 00 00 00 00 00 00 ea 00 00 00 45	2^{-28}
6	00 00 00 af 00 00 00 60 00 00 00 00 00 00 00 60	00 00 00 00 $b4$ 00 00 00 $b6$ 00 00 00 00 00 00 00	00 00 00 af $b4$ 00 00 60 $b6$ 00 00 00 00 00 00 60	00 00 00 $f1$ fd 00 00 $**$ $**$ 00 00 00 00 00 00 ce	00 00 00 $f1$ 00 00 $**$ fd 00 00 $**$ 00 ce 00 00 00	$2^{-28.2}$
7	$5b$ 00 $**$ fe 68 00 $**$ eb $a5$ 00 $**$ ff 00 00 00 00	32 00 00 00 00 00 00 37 00 00 00 00 00 00 00 00	69 00 $**$ fe 68 00 $**$ dc $a5$ 00 $**$ ff 00 00 00 00	$**$ 00 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 $**$ $**$ $**$ 00 $**$ $**$	$**$ 00 $**$ $**$ 00 $**$ $**$ $**$ $**$ $**$ $**$ 00 $**$ $**$ 00 $**$	
∇TK_0^1 :	00 00 00 00 00 00 00 00 00 $d3$ 30 00 00 00 00 00					
∇TK_0^2 :	00 00 00 00 00 00 00 00 00 $c7$ $a8$ 00 00 00 00 00					
∇TK_0^3 :	00 00 00 00 00 00 00 00 00 $d2$ 44 00 00 00 00 00					
Round i	∇W_{i-1}	∇STK_i	∇X_i	∇Y_i	∇Z_i	
6	$**$ $**$ 00 $**$ 00 $**$ $**$ 00 00 $**$ $**$ $**$ $**$ 32 $**$ $**$	00 00 00 37 00 00 00 00 00 00 00 00 00 32 00 00	$**$ $**$ 00 $**$ 00 $**$ $**$ 00 00 $**$ $**$ $**$ $**$ 00 $**$ $**$	cb $**$ 00 22 00 95 $**$ 00 00 $5a$ $e5$ $**$ $**$ 00 1e cd	cb $**$ 00 22 95 $**$ 00 00 $e4$ $**$ 00 $5a$ cd $**$ 00 1e	
7	00 00 00 00 00 00 00 $d2$ $c1$ 00 00 $b4$ $b6$ $**$ 00 00	00 00 00 00 00 00 00 00 00 00 00 $b4$ $b6$ 00 00 00	00 00 00 00 00 00 00 $d2$ $c1$ 00 00 00 00 $**$ 00 00	00 00 00 00 00 00 00 $d6$ 0c 00 00 00 00 6d 00 00	00 00 00 00 00 00 $d6$ 00 00 00 0c 00 00 00 6d 00	
8	00 00 00 00 00 00 ce 00 00 00 79 00 00 00 00 00	00 00 00 00 00 00 ce 00 00 00 79 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
9	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1
10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1

Table 16: A 23-round related-tweakey boomerang distinguisher for SKINNY-128-384 proposed in [14], where R_{11} to R_{13} denote 3-round E_m, u statisfies DDT[0x20][u]>0 and DDT[u][0x5b] >0, v satisfies DDT[0x5b][v]>0 and DDT[v][0xc0] >0, $w_{1/2}$ satisfies DDT[v][$w_{1/2}$] > 0 and DDT[$w_{1/2}$][0x04] >0, u' satisfies DDT[0x02][u'] >0 and DDT[0x42][u'] >0, v' satisfies DDT[u'][v'] >0 and DDT[v'][0x50] >0.

Round	State	Upper differential	Lower differential
R_0	ΔX	0,0,0,0,0,0,0,01,0,0,0,0,0,0,0,20	
	ΔY	0,0,0,0,0,0,0,8,0,0,0,0,0,0,0,u	
	ΔSTK	0,0,0,0,0,0,0,0	
R_1	ΔX	0,0,u,0,0,0,0,01,0,0,0,0,0,0,0,0	
	ΔY	0,0,5b,0,0,0,0,8,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,5b,0,0,0,0,0	
$R_2 - R_6$	ΔX	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,0,0,0,0,0,0	
R_7	ΔX	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,0,0,0,5b,0,0	
R_8	ΔX	0,0,0,0,0,0,0,0,0,0,5b,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,0,v,0,0,0,0,0	
	ΔSTK	0,0,0,0,0,0,0,0	
R_9	ΔX	v,0,0,0,0,0,0,0,v,0,0,0,v,0,0,0	
	ΔY	w_1,0,0,0,0,0,0,0,w_2,0,0,0,c0,0,0,0	
	ΔSTK	0,0,0,0,0,0,0,0	
R_{10}	ΔX	w_1,0,w_2,0,w_1,0,0,c0,0,0,0,w_2,0,w_1,0,w_2,c0	
	ΔY	04,0,04,04,0,04,0,0,04,0,04,0,04,0,04,04	
	ΔSTK	0,0,0,0,0,0,0,0	
R_{11}	ΔX	0,04,0,04,04,0,04,0,0,04,0,0,0,0,04,0	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-
	ΔY	0,*,0,*,*,0,*,0,0,*,0,0,0,0,*,0	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-
	ΔSTK	0,0,0,0,0,0,0,61	0,0,0,0,0,0,0,0
R_{12}		middle part	middle part
R_{13}	∇X	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-	0,*,0,*,*,0,0,*,0,0,0,*,*,0,0,0
	∇Y	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-	0,02,0,42,02,0,0,02,0,0,0,02,42,0,0,0
	∇STK	0,68,0,0,0,0,0,0	0,0,0,0,0,0,0,0
R_{14}	∇X		0,0,0,0,0,02,0,42,02,0,0,0,0,0,0,42
	∇Y		0,0,0,0,0,u',0,58,u',0,0,0,0,0,0,u'
	∇STK		0,0,0,0,0,0,0,58
R_{15}	∇X		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u',0
	∇Y		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,v',0
	∇STK		0,0,0,0,0,0,0,0
R_{16}	∇X		0,v',0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇Y		0,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇STK		0,50,0,0,0,0,0,0
$R_{17} - R_{21}$	∇X		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇Y		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇STK		0,0,0,0,0,0,0,0
R_{22}	∇X		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇Y		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇STK		0,0,0,0,50,0,0,0

Table 17: A 22-round related-tweakey boomerang distinguisher for SKINNY-64-192, where ΔX and ΔY dentoe the differences before and after the S-box layer, R_{10} to R_{15} denote the 6-round E_m, u statisfies DDT[0x1][u]>0 and DDT[$u\oplus$0x9][0xb] >0, v satisfies DDT[0x1][v]>0 and DDT[v][0xb] >0.

Round	State	Upper differential	Lower differential
R_0	ΔX	1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0	
	ΔY	9,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0	
	ΔSTK	9,0,0,0,0,0,0,8	
$R_1 - R_4$	ΔX	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,0,0,0,0,0,0	
R_5	ΔX	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,1,0,0,0,0,0	
R_6	ΔX	0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0	
	ΔY	0,0,8,0,0,0,u,0,0,0,0,0,0,0,v,0	
	ΔSTK	0,0,8,0,0,0,9,0	
R_7	ΔX	0,v,0,0,0,0,0,0,0,0,$u\oplus$0x9,0,0,0,0	
	ΔY	0,b,0,0,0,0,0,0,0,0,0,b ,0,0,0,0	
	ΔSTK	0,0,0,0,b,0,0,0	
R_8	ΔX	0,0,0,0,0,b,0,0,0,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,e,0,0,0,0,0,0,0,0,0,0	
	ΔSTK	0,0,0,0,5,e,0,0	
R_9	ΔX	0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,0	
	ΔY	0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0	
	ΔSTK	0,0,0,0,0,0,2,0	
R_{10}	ΔX	0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,2	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-
	ΔY	0,0,0,*,0,0,0,0,0,0,0,0,0,0,0,*	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-
	ΔSTK	0,0,0,3,0,0,1,0	0,0,0,0,0,0,0,0
$R_{11} - R_{14}$		middle part	middle part
R_{15}	∇X	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-	0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,2
	∇Y	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,-	0,0,0,*,0,0,0,0,0,0,0,0,0,0,0,*
	∇STK	0,0,0,0,0,0,0,6	0,a,0,0,0,0,0,0
$R_{16} - R_{20}$	∇X		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇Y		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇STK		0,0,0,0,0,0,0,0
R_{21}	∇X		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇Y		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	∇STK		0,0,0,0,a,0,0,0