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

Anonymous submission

Abstract. The TWEAKEY framework, introduced by Jean, Nikolić, and Peyrin at ASIACRYPT 2014, provides a generic construction for designing tweakable block ciphers. Prominent instances include Deoxys-BC and SKINNY. 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}, yielding the best attacks to date. Moreover, we extend the longest existing attacks on the authenticated encryption scheme, Deoxys-I-256-128 and Deoxys-I-128-128, by one round. For SKINNY, we improve upon prior best attacks by one round for SKINNY-128-384 and by three 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.* [29] 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 more and more significant cornerstone for designing symmetric-key cryptographic schemes. Examples include highly secure message authentication codes [19, 25,34] and various authenticated encryption schemes in different setting and with flexible security goals [23, 24, 35, 36, 40]. The trends can also be witnessed by the candidates submitted to the CAESAR Competition [1] and NIST lightweight cryptography competition [3].

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 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 [29], or tweaking generic constructions of block cipher such as tweaking Luby-Rackoff cipher [18] and tweaking Key-Alternating cipher [14]. 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 [28,32,37,48]. 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 tweak injection method, such as Hasty Pudding Cipher [39], Threefish [17], and `TWEAKEY` framework [26]. These ad-hoc designs are very efficient, usually conjectured with high security bound and hence meet requirements of practical applications. Their instantiations are being pushed towards standardization and adoptions in practice. However, such ad-hoc tweakable block ciphers lack of provable security, and thus their security must be thoroughly scrutinized before practical utilization.

In particular, the `TWEAKEY` framework [26] enables the construction of tweakable block ciphers with arbitrary tweak and key sizes from a given round function. A notable specialization is the `STK` (Superposition TWEAKEY) construction. Prominent instances include `Deoxys-BC` [27] and `SKINNY` [7], both of which are ISO/IEC standards [2,4]. `Deoxys-BC` comes in two versions, `Deoxys-BC-384` and `Deoxys-BC-256`, and serves as the underlying block cipher of the AE scheme `Deoxys` [27], where `Deoxys-I` was selected as a third-round candidates of the CAESAR competition while `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 `STK` construction treats the key and tweak uniformly, referred to as the *tweakey*. To ensure security against related-tweakey attacks, the `STK` construction employs cell-wise operations in the tweakey schedule. Specifically, for an n-bit block cipher with $tk = np$ tweakey bits, the construction applies the same cell-position permutation to each of the p n-bit tweakey words independently, and then updates each cell of $p-1$ tweakey words using linear operations. After that, XORing the p updated tweakey words gives rise to a new subtweakey. Importantly, the `STK` construction does not allow diffusion across cells. Furthermore, the chosen linear operations (typically finite field multiplications or linear feedback shift registers) must ensure that, for any active cell position, at most $p-1$ difference cancellations can occur during the XOR operation within one period. This property makes it straightforward to analyze the number of active S-boxes in both differential [10] and linear [31] cryptanalysis. As a result, the `STK` framework enables ciphers such as `Deoxys-BC` and `SKINNY` to achieve simple, provable bounds on their resistance to differential and linear attacks even in the related-tweakey setting.

Difference cancellations in the subtweakeys allow differential trails to be extended over additional rounds in the related-tweakey setting, as demonstrated in [7,12]. This implies that rectangle attacks [8] (boomerang attacks [46] as well), which combine two differential trails, can be particularly effective against tweakable block ciphers designed under the `STK` construction, since they can enjoy the benefits of difference cancellation twice. Indeed, rectangle attacks have already proven powerful in the cryptanalysis of both `Deoxys-BC` and `SKINNY`.

The security of `Deoxys-BC` has been extensively studied using rectangle, boomerang, and impossible differential attacks, as in [6, 12, 33, 38, 53]. Among these, rectangle attacks have proven to be the most powerful. In [12], Cid *et al.* presented the first third-party analysis of `Deoxys-BC`, introducing a Mixed Integer Linear Programming (MILP) model to search for efficient rectangle distinguishers. As a result, 13 rounds of `Deoxys-BC-384` and 10 rounds of `Deoxys-BC-256` can be attacked. Subsequently, the *boomerang connectivity table* (BCT) [13] and its variants, such as *boomerang difference table* (BDT) [47], were introduced to better capture the dependency between the two differential trails in a boomerang distinguisher. Incorporating the BTD into the MILP model of [12], the authors of [50, 51] identified new boomerang distinguishers of `Deoxys-BC`, which proved more suitable for key recovery attacks. This advancement extended the rectangle attack to 14 rounds of `Deoxys-BC-384` and 11 rounds of `Deoxys-BC-256`. Further improvements were achieved by using refined rectangle key-recovery algorithms, thereby reducing the time complexities of attacks on both versions [16,44]. Most recently, Song *et al.* proposed a one-step framework that integrates the boomerang distinguisher and key recovery phases into a single optimization model [43]. This unified approach pushed the rectangle attack on `Deoxys-BC-384`, leaving only a one-round security margin.

`SKINNY` has also been subject to extensive cryptanalysis, including impossible differential, integral, meet-in-the-middle (`MITM`), and rectangle attacks, among others [16, 21, 22, 41]. In the single-key setting, the integral and `MITM` attacks are the most effective, while the rectangle attacks dominate in the related-tweakey setting. The first rectangle attacks on `SKINNY` were proposed in [30], though they did not formally address the dependency in the middle rounds. This limitation was later overcome by Delaune *et al.* [15], who developed an MILP model capable of handling such dependencies and hence obtained stronger boomerang distinguishers. Combining this MILP model with a new rectangle key recovery algorithm, Dong *et al.* [16] built an MILP model that unifies the distinguisher search and the key recovery process, yielding improved rectangle attacks on `SKINNY`. Later, Song *et al.* [44] generalized the rectangle key recovery algorithm, which further strengthened the attacks on `SKINNY`.

Motivations. Although rectangle attacks are the most effective method for analyzing both `Deoxys-BC` and `SKINNY` in the related-tweakey setting, certain properties of their tweakey schedule remain unexplored. While it is well known that the number of difference cancellations is inherently limited, the following features of the tweakey schedule have received little attention and, to our 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*.

With respect to automatic search models for rectangle attacks, the existing approaches [16,43] 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. Furthermore, the state-test technique [11], 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 these models. Most importantly, none of the existing models account for the strong key bridges described above, leaving room for further improvements.

Our contributions. We conduct a formal study 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 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`: when $p = 3$, 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. Like previous one-step models, 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 un-guessed 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-256-128` and `Deoxys-I-128-128`, by one round. For `SKINNY`, our results improve the prior best attacks by one round for `SKINNY-128-384` and three 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.

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

Cipher	#R	Data	Mem.	Time	Attack	Setting	$\|K\|$	Ref.
`Deoxys-BC-384`	15	$2^{115.7}$	2^{128}	$2^{371.7}$	Rect.	RTK	384	[43]
	15	$2^{115.7}$	2^{128}	$\mathbf{2^{297.7}}$	Rect.	RTK	320	Sect. 5.1
	15	$2^{115.7}$	2^{128}	$\mathbf{2^{331.7}}$	Rect.	RTK	384	Sect. 5.1
`Deoxys-BC-256`	11	$2^{126.78}$	2^{128}	$2^{222.49}$	Rect.	RTK	256	[49]
	11	$\mathbf{2^{119}}$	$\mathbf{2^{119}}$	$\mathbf{2^{189.8}}$	Rect.	RTK	256	Sect. C.1
`Deoxys-I-256-128`	13	$2^{125.2}$	2^{136}	$2^{186.7}$	Rect.	RTK	256	[51]
	14	$2^{115.1}$	2^{121}	$2^{235.1}$	Rect.	RTK	128	Sect. C.3
`Deoxys-I-128-128`	9	2^{117}	2^{117}	2^{118}	Rect.	RTK	128	[12]
	10	2^{91}	2^{91}	$2^{121.8}$	Rect.	RTK	256	Sect. C.3
`SKINNY-128-384`	26	2^{122}	2^{328}	2^{331}	Int.	STK	360	[21]
	27	$2^{124.99}$	2^{344}	$2^{362.61}$	ID	RTK	384	[22]
	32	$2^{123.54}$	$2^{129.54}$	$2^{344.78}$	Rect.	RTK	384	[49]
	33	$2^{123.54}$	$2^{123.54}$	$2^{359.54}$	Rect.	RTK	384	Sect. C.2
`SKINNY-64-192`	26	2^{62}	2^{164}	2^{166}	Int.	STK	180	[21]
	27	$2^{63.64}$	2^{172}	$2^{183.26}$	ID	RTK	192	[22]
	31	$2^{62.78}$	$2^{62.79}$	$2^{182.07}$	Rect.	RTK	192	[16]
	31	$2^{62.78}$	$2^{62.79}$	$\mathbf{2^{154.07}}$	Rect.	RTK	192	Sect. 5.2
	34	$2^{63.39}$	$2^{64.12}$	$2^{190.78}$	Rect.	RTK	192	Sect. 5.2

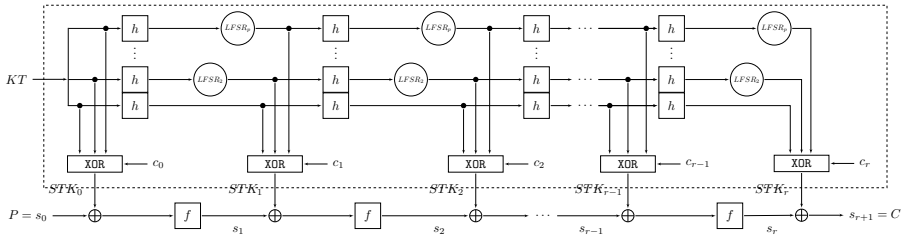

Figure 1: The STK framework [26].

2 Preliminaries

2.1 The STK Framework

The TWEAKEY framework was proposed by Jean *et al.* [26] to build an n-bit tweakable block cipher with the variablesize 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.

Moreover, Jean *et al.* proposed a specialized version of TWEAKEY, called the Superposition Tweakey (STK) framework. Following the STK framework, several tweakable block ciphers have been proposed, including Deoxys-BC, SKINNY, etc.

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 $(TK^1, \cdots, TK^p)$. As shown in Figure 1, for Deoxys and SKINNY, the functions g, f and u are defined as follows:

- 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, $STK_i = \bigoplus_j^p TK_i^j$, while for Deoxys-BC, $STK_i = \bigoplus_j^p TK_i^j \oplus c_i, 0 \leq i \leq r$, 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 (LFSRs) applied to each cell within each of the last $(p-1)$ n-bit words of the tweakey state, *i.e.*, $TK_i^1 = h(TK_{i-1}^1), TK_i^j = h(LFSR_j(TK_i^j)), 2 \leq j \leq p$.

2.2 Deoxys-BC and SKINNY

Description of Deoxys-BC. Deoxys-BC is a building cornerstone for the authenticated encryption scheme Deoxys [27], one finalist of the CAESAR competition.

Table 2: The LFSRs used in Deoxys-BC and SKINNY-128 tweakey schedule.

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

It is an AES-like design, which adopts AES's round function and the STK framework [26]. 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. The ordering of the state cells is shown in Figure 2. Each round applies four transformations in the following order.

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

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

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

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 $LFSR_s$ used in Deoxys-BC is shown in Table 2. It should be noted that in [26], the designer originally employed the multiplication operation over $GF(2^8)$.

The Deoxys AEAD modes. Based on Deoxys-BC, Jean *et al.* introduced two nonce-based AEAD modes: Deoxys-I and Deoxys-II. Deoxys-I is mainly intended for the nonce-respecting setting, and Deoxys-II provides security even in the nonce-misuse setting. In [12], it is noted that for Deoxys a message cannot exceed 2^{60} blocks, while a maximum of 2^{64} messages can be encrypted under the same secret key. Thus, when attacking one of the Deoxys-I versions, an adversary can obtain at most 2^{124} blocks of data under the same key.

Description of SKINNY. The block cipher family SKINNY was proposed at CRYPTO 2016 [7]. It has 6 distinct versions, denoted as SKINNY-n-t, where $n \in \{64, 128\}$

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

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

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 a 64-bit (resp. 128-bit) constant to the state Y_i, when n is 64 (resp. 128).
- 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: $M = \begin{pmatrix} 1&0&1&1 \\ 1&0&0&0 \\ 0&1&1&0 \\ 1&0&1&0 \end{pmatrix}$, $M^{-1} = \begin{pmatrix} 0&1&0&0 \\ 0&1&1&1 \\ 0&1&0&1 \\ 1&0&0&1 \end{pmatrix}$.

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 LFSRs are given in Table 2 (resp. Table 3).

2.3 The Generic Rectangle Key Recovery Attack

Since the rectangle attack [8] was proposed, research on the rectangle attack for key recovery has been extensively studied in the literature [8,9,52]. A significant advancement was made by Song *et al.* at Asiacrypt 2022 [44], which proposed a unified framework for finding the best rectangle attack. It contains a generic key recovery algorithm and supports any key recovery strategy.

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 rectangle distinguisher over E_d holds with probability $P_d = Pr[E_d^{-1}(E_d(x) \oplus \delta) \oplus E_d^{-1}(E_d(x \oplus \alpha) \oplus \delta)] = P^2$.

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'). And, one requires subkey information k_b (resp. k_f) to verify the difference α' (resp. δ') for plaintext (resp. ciphertext) pairs. Let m_b and m_f denote the number of the bits in k_b and k_f, *i.e.*, $m_b =$

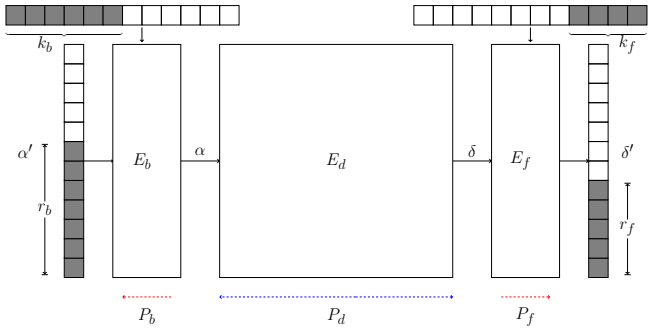

Figure 4: A high-level description of the generic rectangle key recovery attack.

$|k_b|, m_f = |k_f|$. In the generic key recovery attack, it allows attackers to guess some key bits in advance, which can sieve the data faster. 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 r'_b-bit condition in the E_b component and 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.

1. **Data collection.** Collect and store y structures of 2^{r_b} plaintexts. The time complexity of this step is $T_0 = D = y \cdot 2^{r_b} = \sqrt{s}2^{n/2}/P$, 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_1, C_1), partially encrypt P_1 and partially decrypt C_1 with the guessed subkey bits k'_b and k'_f and get P^*_1 and C^*_1. The time complexity of this step is $T_1 = 2^{|k'_b \cup k'_f|} \cdot D$.
   - **Pair construction.** Insert all the obtained (P^*_1, C^*_1) into a hash table by the inactive bits of P^*_1 or C^*_1 to construct a set of pairs $S = \{(P^*_1, C^*_1, P^*_2, C^*_2)\}$ or $S = \{(P^*_1, C^*_1, P^*_3, C^*_3)\}$. The time complexity of this step is $T_2 = 2^{|k'_b \cup k'_f|} \cdot D \cdot \min\{2^{r^*_b - 1}, D \cdot 2^{r^*_f - n - 1}\}$.
   - **Quartets generation and processing.** Insert S into a hash table by the inactive bits of C^*_1 and C^*_2 or P^*_1 and P^*_3. Then, generate the quartet for each index. Determine the key candidates involved in E_b and E_f and increase the corresponding counters. The time complexity of this step is $T_3 = 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2} \cdot \epsilon$, where $\epsilon \geq 1$ depending on the concrete situation.
3. **Exhaustive searching.** According to the key schedule algorithm, 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 $T_4 = 2^{k-l}$, where $l \leq q + |k_b \cup k_f| - |k'_b \cup k'_f|$ and $0 \leq q \leq |k'_b \cup k'_f|$.

The data complexity is D. The memory complexity is $M = D + \min\{D \cdot 2^{r^*_b - 1}, D^2 \cdot 2^{r^*_f - n - 1} + 2^{q + |k_b \cup k_f| - |k'_b \cup k'_f|}\}$ for storing the data, the pairs and the key counters.

3 Tweakey Schedule Properties of Deoxys-BC and SKINNY

Both Deoxys-BC and SKINNY utilize the STK framework. This design combines a cell-wise permutation h with linear feedback shift registers (LFSRs) to linearly update the internal state. This section analyzes their key schedule algorithms, identifying strong key bridges between specific subtweakeys that stem from the LFSR operations. For Deoxys-BC, we further examine the structural properties of h. These properties are then leveraged to accelerate the search for rectangle attacks on Deoxys-BC-384.

3.1 Strong Key Bridges from LFSRs

For both SKINNY and Deoxys-BC, the tweakey schedule applies the same permutation h to each tweakey word $TK^j, 0 < j \leq p$, and then applies an LFSR to update the cells of TK^2 and TK^3. A subtweakey is obtained by XORing the p tweakey words. Therefore, the i-th ($i \geq 0$) subtweakey cell of the round r ($r \geq 0$) $STK_r[i]$ involves only p tweakey cells, which is the same as those involved in $STK_0[h^{-r}(i)]$. Since there is no diffusion between tweakey cells, we can analyze the relation between subtweakeys through each group of p tweakey cells.

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

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

We denote the three tweakey words as TK^1, TK^2, TK^3. Each cell $TK^j[i] (1 \leq j \leq 3, 0 \leq i \leq 15)$ 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

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

where $i_r = h^{-r}[i], 0 \leq i \leq 15$. 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 $STK_0[i] = \left[\boldsymbol{I}\ \boldsymbol{I}\ \boldsymbol{I}\right] \cdot (\boldsymbol{tk}^1[i], \boldsymbol{tk}^2[i], \boldsymbol{tk}^3[i])^\top$.

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 internal 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 proposition.

Proposition 1. *For the i_m-th byte of the m-th subtweakey $STK_m[i_m]$, where $1 \le m \le 30, 0 \le i \le 15$, the information bits of $|STK_0[j] \cup STK_m[i_m]|$ exhibit the following properties:*

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

Proof. The tweakey schedule algorithm shows that

$$STK_m[i_m] = \left[\boldsymbol{I}\ \boldsymbol{L}_2^m\ \boldsymbol{L}_3^m\right] \cdot (\boldsymbol{tk}^1[i], \boldsymbol{tk}^2[i], \boldsymbol{tk}^3[i])^\top$$

and the number of information bits is the rank of the matrix $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{L}_2^m & \boldsymbol{L}_3^m \end{bmatrix}$. Due to $\boldsymbol{L}_3 = \boldsymbol{L}_2^{-1}$ and $\boldsymbol{L}_2^{30} = \boldsymbol{I}$, it can be concluded that

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

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 $\texttt{Rank}(\boldsymbol{L_2^{15}}+I) = 4$, we have $\texttt{Rank}(\boldsymbol{C}) = 12$ and $|STK_0[j] \cup STK_{15}[j_{15}]| = 12$.
- 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 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 the following proposition.

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

- *when $m = 15, n \ne 30$ or $n = 15$ or $n - m = 15, n \ne 30$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 20$;*
- *when $m = 15, n = 30$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 12$;*
- *when $m \ne 15, n = 30$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 16$;*

– *when* (m, n) *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 independence of the subtweakey becomes independent of TK^1. Without loss of generality, TK^1 can be considered a constant, and $STK_r[i_r]$ is expressed as

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

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 $|STK_0[i] \cup STK_r[i_r]| = \mathtt{Rank}(\boldsymbol{C})$. Based on the above, we derive the following proposition.

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

– *when* $m = 15$ *or* $m = 30$, $|STK_0[i] \cup STK_m[i_m]| = 8$*;*
– *when* m *in other cases,* $|STK_0[i] \cup STK_m[i_m]| = 16$.

It is significant to note that using the multiplication operation on $GF(2^8)$, as prescribed in [26], effectively eliminates the dependency separated by 15 rounds. This implies that the properties delineated in Propositions 1 through 3 for $m = 15$ or $n - m = 15$ are rendered invalid.

(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* $STK_m[i_m]$*, where* $i_m = h^{-m}[i], 1 \leq m \leq 15, 0 \leq i \leq 15$*, the information bits of* $STK_0[i] \cup STK_m[i_m]$ *exhibit the following properties:*

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

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

– *when* $n = 15$, $|STK_0[i] \cup STK_m[i_m] \cup STK_n[i_n]| = 8$*;*
– *when* (m, n) *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, the information bits satisfy the properties:*

- *when* $1 \leq m < n \leq r$*, the information bits are* $|STK_m[i_m] \cup STK_n[i_n]| = |STK_0[i] \cup STK_{n-m}[i_{n-m}]|$*;*
- *when* $1 \leq l < m < n \leq r$*, the information bits are* $|STK_l[i_l] \cup STK_m[i_m] \cup STK_n[i_n]| = |STK_0[i] \cup STK_{m-l}[i_{m-l}] \cup STK_{n-l}[i_{n-l}]|$.

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.

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

(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.

Without loss of generality, we let $eqSTK = \texttt{MC}^{-1}(STK)$, that is

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

where $r \geq 1, i \in \{0, 4, 8, 12\}$. We analyze the relationship between the r-th and $(r+15)$-th rounds. The independence of other rounds, which will be addressed later via rank calculation, depends on the final attack parameters.

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-independent 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 $|eqSTK_{15+m}[I_{15+m}] \cup STK_m[I_m]| = |eqSTK_{15}[I_{15}] \cup STK_0[I_0]|$.*

3.2 Property of h Used in `Deoxys-BC`

Definition 1. *The column spaces $\mathcal{C}_i$ 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

$$\mathcal{C}_0 = \left\{ \begin{bmatrix} x_1 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 \end{bmatrix} \middle| \forall x_1, x_2, x_3, x_4 \in \mathbb{F}_2^8 \right\} \equiv \begin{bmatrix} x_1 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 \end{bmatrix}.$$

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

$$\mathcal{D}_0 = \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & x_4 \end{bmatrix}, \mathcal{ID}_0 = \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 \\ 0 & 0 & x_3 & 0 \\ 0 & x_4 & 0 & 0 \end{bmatrix}$$

for all $x_1, x_2, x_3, x_4 \in \mathbb{F}_2^8$.

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, for $\forall i \in \{0,1,2,3\}$, $h^{-1}(\mathcal{C}_i) = \mathcal{ID}_i, h^{-3}(\mathcal{C}_i) = \mathcal{D}_{(i+1 \bmod 4)}, h^{-4}(\mathcal{C}_i) = \mathcal{C}_{(i+2 \bmod 4)}$ 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. Suppose 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, the minimum number of active S-boxes in the 6-round distinguisher is 10.*

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

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

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

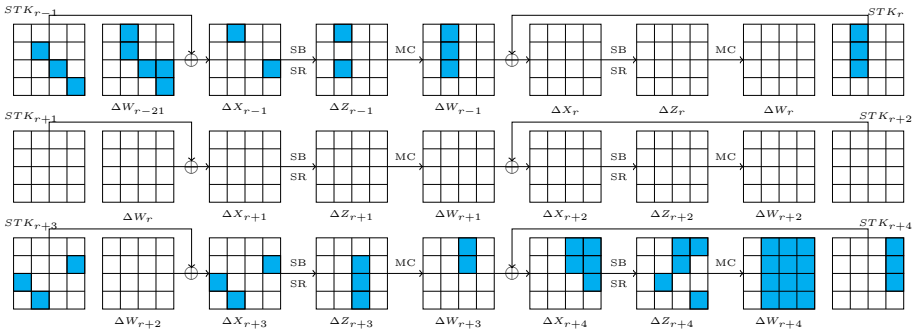

Figure 5: The 5-round and 6-round distinguishers of `Deoxys-BC-384`

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 $5-x$ active bytes in ΔZ_{r-1} at least 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

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

$I = \{i+1 \bmod 4, i+2 \bmod 4\}$. And 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 model, 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 acilitates 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 $P_f \leq 1$. Thus, the overall probability of the distinguisher 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 guessing solely 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 attack [11]. A slightly different version was reinvented for rectangle attacks in [43]. The first automatic MILP-based model was provided to enhance differential MITM attacks [5].

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. Our model incorporates several established approaches and improves them by integrating state-of-the-art techniques. Additionally, we offer three approaches to compensate for the deficiency in determining the optimal parameters of the new automated models.

Distinguisher and probabilistic extension. Justified by the principle that, under probabilistic propagation, a shorter distinguisher can effectively cover the longest one. Accordingly, we model the distinguisher by adapting classical methods with a reduced rounds to enhance flexibility, while the probabilistic extension is implemented following the approach introduced in [43].

New models for state test, key guessing strategy, ϵ calculation and key bridging. We introduce the first CP model for the state test and seamlessly integrate it with both key guessing strategy and ϵ calculation through a flexible architecture. For ϵ calculation, we present the first automatic model, which supports two approaches: guess-and-filter and pre-computation hash tables, to determine the complexity accurately. Extending the classical key bridging, we incorporate the property introduced in Section 3 into our model to maximize its utility.

For clarity, we take a toy cipher as an example to illustrate our model, as shown in Figure 6, where the equivalent subkeys are employed to replace the subkeys, *i.e.*, $Eqk = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(STK)$.According to the toy cipher, Table 5 summarizes the variables used in our model. The notation $\mathcal{V}$ denotes the set of state variables $\{X, Z, W\}$, which adhere to $X_r \xrightarrow[\mathtt{AEqk}]{\mathtt{SB}\circ\mathtt{SR}} Z_r \xrightarrow{\mathtt{MC}} W_r \xrightarrow{\mathtt{ART}} X_{r+1}$. Additionally, variables dedicated to individual components will be introduced in corresponding sections.

4.1 Model for the State Test

The state test is strategically integrated across three core components of our model: key guessing and ϵ calculation.When activated, it eliminates disjoint sets

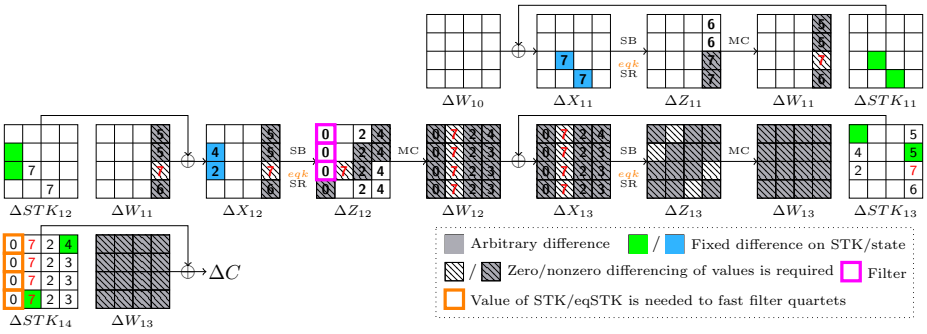

Figure 6: The toy example of E_f in the rectangle attack, the position of the round key label actually corresponds to the position of the equivalent key.

Table 5: Variables employed in our model

variables	domain	coverage	representation
$d\mathcal{V}$ $dSTK$	{0,1,2} {0,1}	$\{rE_b\} \cup \{rE_d\} \cup \{rE_f\}$	0: zero difference 1: determined non-zero difference 2: arbitrary difference
$v\mathcal{V}$, $vSTK$	{0,1}	$\{rE_b\} \cup \{rE_f\}$	0: value involved 1: value non-involved
$t\mathcal{V}$	{0,1,2}	$\{rE_b\} \cup \{rE_f\}$	0: untested state 1: tested state in key guessing strategy 2: tested state in ϵ calculation
$e\mathcal{V}$, $eSTK$	{0,1,2}	$\{rE_b\} \cup \{rE_f\}$	0: uneliminated state/subkey 1: elimination in key guessing strategy 2: elimination in ϵ calculation
$gSTK$	{0,1}	$\{rE_b\} \cup \{rE_f\}$	0: unguessing subkey 1: pre-guessed subkey
$det\mathcal{V}$	{0,1}	$\{rE_b\} \cup \{rE_f\}$	0: undetermined value 1: determined value
frSB,frMC	{0,1}	$\{rE_b\} \cup \{rE_f\}$	0: filter not obtained 1: obtained filter related to SB/MC
$s\mathcal{V}$, $sSTK$	{-1,...,S}	$\{rE_b\} \cup \{rE_f\}$	-1: non-involved in ϵ calculation 0: involved in key guessing strategy {1,...,S}: step in ϵ calculation

of 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 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 differential in our model.

Algorithm 1: Model for the state test

Input: variables in Table 5, model of the distinguisher and the extension.

1 **for** $r \in \{rE_f\} \setminus \{r.E_f\}$ **do**
2 **for** $c \in \{0, ..., 15\}$ **do**
  // determine the tested state
3 **if** $dW_r^c = 2$ *or* $vW_r^c = 0$ **then** $tW_r^c = 0$ **endif**
4 $eX_{r+1}^c = eW_r^c$
  // elimination propagation in SB, AEqk, SR
5 **if** $dZ_{r+1}^{\mathtt{SR}^{-1}(c)} \leq 1$ **then** $eEqk_{r+1}^c = eX_{r+1}^c$ **else** $eEqk_{r+1}^c = 0$ **endif**
6 $eZ_{r+1}^c = eEqk_{r+1}^{\mathtt{SR}(c)}$;
7 **for** $c \in \{0, 4, 8, 12\}$, $i \in \{0, ..., 3\}$ **do**
  // the generation of elimination
8 **if** $tW_r^{c+i} \geq 1$ **then** $eW_r^{c+i} = tW_r^{c+i}$
  // elimination propagation in MC
9 **elseif** $eZ_r^{c+i} \geq 1$ *and* $all((vZ_r^{c+j} = 0)$ *or* $(eZ_r^{c+j} = eZ_r^{c+i})$
10 *for* $j \in \{0, ..., 3\} \setminus \{i\})$
11 **then** $((eW_r^{c+j} = eZ_r^{c+i} \times vW_r^{c+j})$ *for* $j \in \{0, ..., 3\})$
12 **else** $eW_r^c = 0$ **endif**

Algorithm 1 specifies the conditions under which a state can be tested and (or) eliminated. As indicated in line 3, constraints are only applied to disable state testing when the required conditions are not satisfied, thereby allowing flexibility in selecting the tested state in key guessing strategy or ϵ calculation. Once a state is selected for testing, eliminations are generated, as shown in line 8. The rule of elimination propagation is outlined in lines $4 \sim 12$. When the eliminated state turns to hold an unknown differential or loses the disjointedness, the propagation will be terminated. In Figure 6, by testing W_{11}^{14}, $eqSTK_{13}^{14}$ and $eqSTK_{14}^{\{4,...,7\}}$ are all eliminated.

- For `SB`, lines $5 \sim 6$, elimination propagates only if the differential of the tested state after `SB` is determined, implying that probability is consumed when eliminating the active states, thereby influencing the probabilistic extension in a feedback manner.
- For `MC`, lines $8 \sim 12$, elimination propagation requires that all eliminated states within a column remain disjoint from other states, which ensures that the corresponding eliminated subkeys form a disjoint set. Thus, for block ciphers equipped with the MDS matrix elimination can propagate forward through `MC` only if all involved states in the column before `MC` are eliminated.

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 state testing is employed in key guessing strategy, the eliminated subkeys help reduce the number of subkeys

requiring guessing. For the first time, we integrate state testing into the model key guessing strategy, thus significantly improving its efficiency and flexibility. Specifically, add constraint **if** $tW^c_{r-1} = 1$ *or* $detX^c_r = 1$ **then** $detW^c_{r-1} = 1$ **else** $detW^c_{r-1} = 0$ **endif** to involve the influence of the state test into key guessing strategy, and the detailed model is given in Algorithm 5.

4.2 Model for ϵ Calculation

ϵ represents the complexity of processing quartets and extracting key information. 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 section automates ϵ calculation within our full-fledged model, ensuring optimal attack parameters while incorporating the state test impacts. The idea of the core part extends the step assignment method on integral attacks [21]. The core part of the model for ϵ calculation is given in Algorithm 2.

Algorithm 2 initializes the model with a sufficiently large constant S as the maximum step count, ensuring minimal complexity is achieved while allowing unused steps. Besides, there are some variables dedicated to this model that we need to introduce: $sEqk^s_\epsilon$, $fr\texttt{SB}^s_\epsilon$, and $fr\texttt{MC}^s_\epsilon$ represent the number of used equivalent subkeys, filters from `SB` and `MC`, respectively. To begin, lines $2 \sim 4$ determined the starting position of the calculation. During step propagation in lines $5 \sim 13$, the model incorporates the influence of the state test by inheriting relevant selections from preceding models. For instance, in Figure 6, starting points located at $Z_{12}^{\texttt{SR}^{-1}(3)}$ and $Z_{13}^{\texttt{SR}^{-1}(\{0,\ldots,15\}\backslash\{0,5,10,15\})}$; $eqSTK_{14}^{\{0,\ldots,3\}}$ are excluded to steps propagation. Lines $14 \sim 17$ count the number of subkeys used together with the tested state. The associated filters from `SB` and `MC` are gradually obtained only in the corresponding step, as shown in lines $18 \sim 21$.

Immediately following Algorithm 2, we present Algorithm 3 to evaluate complexities for ϵ calculation. Before diving into the algorithm, we introduce additional variables dedicated to this model. We define Tag for method selection (0: guess-and-determine or 1: pre-computation hash table), T^s_ϵ and M^s_ϵ for time and memory costs per step. For pre-computation hash tables, memory cost related to `SB` and `MC` are defined as $tabM\texttt{SB}^s_\epsilon$, $tabM\texttt{MC}v^s_\epsilon$, and $tabM\texttt{MC}d^s_\epsilon$, and $Flag^s_{PQ}$ represents the calculation objective (1: pairs or 2: quartets).

In Algorithm 3, lines $1 \sim 12$ evaluate memory costs for pre-computation hash tables, as illustrated by the toy example in Figure 6.

- For $tabM\texttt{SB}^s_\epsilon$, when a step begins after an `SB` operation, the state value at the starting positions must be stored. In step 2, the values of $Z_{13}^{\texttt{SR}^{-1}(\{2,5,8,15\})}$ is necessary to be stored in the hash table.

– For $tabM\texttt{MC}_\epsilon^s$, values after MC that are determined in prior steps are retained to compute pre-MC values. When more memory-efficient, pre-MC values may be stored as linear combinations of the determined post-MC values instead. In step 4, besides storing $Z_{13}^{\texttt{SR}(12)}$, $W_{12}^{\{13,14,15\}}$ are also required. However, storing $Z_{13}^{\texttt{SR}(12)}$ and $Z_{12}^{\{12,13\}}$ instead reduces memory cost.
– For $tabM\texttt{MC}_\epsilon^s$, when all arbitrary-differential cells in a column are value-determined, the corresponding filters are obtained, requiring storage of differentials from previous steps. For example, in step 6, storing $W_{11}^{\{12,13\}}$ differentials is necessary to derive filters for $Z_{11}^{\{12,13\}}$.

Finally, lines 12 $\sim$ 19 formulate the time and memory complexity for ϵ calculation using the two approaches.

Algorithm 2: Model for the core part of ϵ calculation

Input: variables in Table 5; distinguisher, extension, state test, and subkey guessing models; maximum steps S.

1 **for** $r \in \{rE_f\}, c \in \{0, ..., 15\}$ **do**
  // exclude uninvolved states, mark determined states
2 **if** $vZ^c_{r.E_f-1} = 0$ **then** $sZ^c_{r.E_f-1} = -1$
3 **elseif** $detZ^c_{r.E_f-1} = 1$ **then** $sZ^c_{r.E_f-1} = 0$
4 **else** $sZ^c_{r.E_f-1} \geq 1$ **endif**
5 **for** $r \in \{rE_f\} \setminus \{r.E_f\}$ **do**
6 **for** $c \in \{0, ..., 15\}$ **do**
    // exclude uninvolved subkeys, mark pre-guessed subkeys
7 **if** $vEqk^c_r = 0$ **then** $sEqk^c_r = -1$
8 **elseif** $gEqk^c_r = 1$ *or* $eEqk^c_r = 1$ **then** $sEqk^c_r = 0$
    // step propagation in SB,SR, AEqk and rounds
9 **else** $sEqk^c_r \geq sZ^{\texttt{SR}^{-1}(c)}_r$ **endif**
10 $sX^c_r = sEqk^c_r$
11 $sW^c_{r-1} = sX^c_r \times vW^c_{r-1}$
12 **for** $c \in \{0, 4, 8, 12\}, i \in \{0, ..., 3\}$ **do**
    // step propagation in MC
13 $sZ^{c+i}_{r-1} = max(sW^{c+j}_{r-1}\ for\ j \in \{0, ..., 3\}) \times vZ^{c+i}_{r-1}$
14 **for** $s \in \{1, ..., S\}$ **do**
  // count equivalent subkeys and tested states
15 $sEqk^s_\epsilon = sum((sEqk^c_r = 1\ and\ eEqk^c_r \neq 2) +$
16 $(2 \times (tW^c_r = 2\ and\ sW^c_r = s))$
17 $for\ r \in \{rE_f\} \setminus \{r.E_f\}, c \in \{0, ..., 15\})$
  // count filters from SB and MC
18 $fr\texttt{SB}^s_\epsilon = sum(sX^c_r \times dX^c_r = s\ for\ r \in \{rE_f\} \setminus \{r.E_f\}, c \in \{0, ..., 15\})$
19 $fr\texttt{MC}^s_\epsilon = sum($**if** $s = max(sW^{c+i}_r \times (dW^{c+i}_r = 2)\ for\ i \in \{0, ..., 3\})$
20 **then** $sum((dZ^{c+i}_r \leq 1)\ for\ i \in \{0, ..., 3\})$
21 **else** 0 **endif** $for\ r \in \{rE_f\} \setminus \{r.E_f\}, c \in \{0, ..., 15\})$

Algorithm 3: Model complexities for ϵ calculation

Input: variables inherited form Algorithm 2

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

// memory cost related to SB

2 $tabM\mathtt{SB}_\epsilon^s = sum((sZ_r^{\mathtt{SR}^{-1}(c)} < s \ and \ sX_r^c = s > \ and \ eX_r^c \neq 2)$

3 $for \ r \in \{rE_f\}, c \in \{0, ..., 15\})$

// memory cost for storing values related to MC

4 $tabM\mathtt{MC}v_\epsilon^s = sum($**if** $max(sW_r^{c+i} \ for \ i \in \{0, ..., 3\}) = s \ and$

5 $exist(sW_r^{c+i} < s \ for \ i \in \{0, ..., 3\})$

6 **then** $min(sum(sZ_r^{c+i} = s \ for \ i \in \{0, ..., 3\}),$

7 $sum(sW_r^{c+i} < s \ for \ i \in \{0, ..., 3\}))$

8 **else** 0 **endif** $for \ r \in \{rE_f\} \setminus \{r.E_f\}, c \in \{0, 4, 8, 12\})$

// memory cost for storing differential related to MC

9 $tabM\mathtt{MC}d_\epsilon^s = sum($**if** $max((sW_r^{c+i} \times (dW_r^{c+i} = 2) = s) \ for \ i \in \{0, ..., 3\})$

10 $and \ all(sZ_r^{c+i} \neq s \ for \ i \in \{0, ..., 3\})$

11 **then** $sum(sW_r^{c+i} < s \ and \ dW_r^{c+i} = 2 \ for \ i \in \{0, ..., 3\}),$

12 **else** 0 **endif** $for \ r \in \{rE_f\} \setminus \{r.E_f\}, c \in \{0, 4, 8, 12\})$

// initialize the time consumption of the first step

13 $T_\epsilon^1 = sEqk_\epsilon^1 - Tag \times Flag_{PQ}^1 \times (fr\mathtt{SB}_\epsilon^1 - fr\mathtt{MC}_\epsilon^1)$

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

// time consumption of steps 1 to S

15 **if** $Tag = 1$ **then** $T_\epsilon^s = T_\epsilon^{s-1} + sEqk_\epsilon^s - Flag_{PQ}^s \times (fr\mathtt{SB}_\epsilon^s + fr\mathtt{MC}_\epsilon^s)$

16 **else** $T_\epsilon^s = T_\epsilon^{s-1} - 2 \times (fr\mathtt{SB}_\epsilon^{s-1} + fr\mathtt{MC}_\epsilon^{s-1}) + sEqk_\epsilon^s$ **endif**

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

// memory consumption for pre-computation hash table

18 $M_\epsilon^s = sEqK_\epsilon^s + Flag_{PQ}^s \times (2 \times (tabM\mathtt{SB}_\epsilon^s + tabM\mathtt{MC}v_\epsilon^s) +$

19 $tabM\mathtt{MC}d_\epsilon^s - fr\mathtt{SB}_\epsilon^s - fr\mathtt{MC}_\epsilon^s)$

4.3 Model for Key Bridging

This section introduces a model that integrates the key bridging into our full-fledged model. Expect the classical approaches, our model incorporates a stronger key bridging. In Algorithm 4, we embed the properties of `Deoxys-BC`'s tweakey schedule, given in Section 3.1, into our model. Line 2 establishes the column-wise relationship between $vEqk_{15}$ and $vSTK_0$, to ensure that they remain in the same lane. The number of redundant subkey bits can then be counted by accessing Table 4.

By modifying line 2, we can make this model compatible with the key guessing strategy and ϵ calculation, and prevent redundant bits from being utilized multiple times.

– For subkey pre-guessing, using the constraint $all(gEqk_{15}^{\mathtt{SR}(col+i)} \times vEqk_{15}^{\mathtt{SR}(col+i)} = 1 \ for \ i \in \{0, ..., 3\})$ to ensures that the key bridging is only applied when all involved subkeys in a column have been guessed.

Algorithm 4: Model for key bridging of `Deoxys`

Input: variables in Table 5; models for the distinguisher, extension, state test, and ϵ calculation; subkey permutation table $hTable$; key bridging table $JTable$ (Table 4); the number of redundant subkey bits per column vRd^{col}

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

  `// counting for redundant subkey bits`

  if $exists(vEqk_{15}^{\mathtt{SR}(col+i)}\ for\ i \in \{0, ..., 3\})$

  then $vRd^{col} = JTable[sum(vSTK_0^{hTable^{-15}(col+i)}\ for\ i \in \{0, ..., 3\}),$

  $sum(vEqk_{15}^{col+i}\ for\ i \in \{0, ..., 15\})]$

  else $vRd^{col} = 0$ **endif**

$m_b + m_f = sum((vEqk_r^c + 2 \times tW_r^c)\ for\ r \in \{rE_b\} \cup \{rE_f\}, c \in \{0, ..., 15\}) - sum(Rd^{col}\ for\ col \in \{0, 4, 8, 12\})$

– For ϵ calculation, using the constraint $exists(sEqk_{15}^{col+i} \geq 1\ for\ i \in 0, ..., 3)\ and\ all(sEqk_{15}^{col+i} \leq max(sEqk_{15}^{col+j}\ for\ j \in 0, ..., 3)\ for\ i \in 0, ..., 3)$ to ensures that the key bridging is only utilized when all involved subkeys in a column have been processed.

Model the Subkey Relation for `SKINNY` The key bridging for `SKINNY-128-384` and `SKINNY-64-192` adhere to the conditions specified in Proposition 1, 2 and Proposition 4, 5. The method for incorporating corresponding key bridging into our model can be simply achieved by absorbing the logic presented in Algorithm 4.

4.4 Elasticizing the Full-Fledged Model

Our full-fledged model integrates all methods introduced in this section. When all components are activated, all potential complex relationships among them are satisfied, enabling the model to return a globally optimal solution without relying on any pre-assumptions.

However, solving such a comprehensive model demands substantial time and computational resources. To balance optimality and efficiency, each component of our model incorporates a configurable switch that enables or disables it. When deeper components are disabled to accelerate solving, we can obtain locally optimal solutions that are partial to the optimal attack, and help prune the search space in later phases. In this section, we present three styles commonly used to elasticize the full-fledged model and find locally optimal solutions, as well as provide guidance for achieving global optimal solutions.

– Enabling only the distinguisher, our model identifies the highest probability distinguisher for a given round. If the obtained probability approaches the boundary, it is recommended to reduce the distinguisher by one round.

– With key guessing and ϵ calculation disabled, we formulate a multi-objective function as minimizing $a \times Pr + b \times (m_b + m_f)$, where constants a and b represent objective weights, and $m_b + m_f$ is computed according to line 6 in Algorithm 4. This configuration helps identify differential patterns that are partial to optimal key recovery attacks.
– By disabling ϵ calculation and assuming $\epsilon = 0$, we can obtain a lower bound for the attack complexity. Then, by enabling ϵ calculation on it, the optimal attack may be discovered if the solution maintains $\epsilon = 0$.

5 Rectangle Attacks on Deoxys and SKINNY

In this section, applying our model to Deoxys-BC and SKINNY, we obtain a series of improved attack results. Utilizing the strong key bridges and the properties of h, we achieve a 15-round rectangle attack on Deoxys-BC-384 with reduced time complexity by a factor of 2^{40}, and the first 34-round SKINNY-64-192 rectangle attacks, which is 3 rounds more than the previous works. We also give improved attacks on Deoxys-BC-256, Deoxys-I, and SKINNY-128-384. Refer to Appendix C for more details.

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

For Deoxys-BC-384, exploiting the properties of the permutation h in the related-tweakey setting, as specified in Proposition 7, we get 5-round differential trails with 5 active S-boxes and 6-round ones with 10 active S-boxes. By restricting the two short differential trails to the activeness patterns defined in Proposition 7, flexibly selecting the number of rounds for E_0, E_1, and E_m, we obtain 32 activeness patterns for the rectangle attacks through our model. After instantiating the obtained patterns, we achieve an improved 15-round rectangle attack, depicted in Figure 7.

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

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.

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

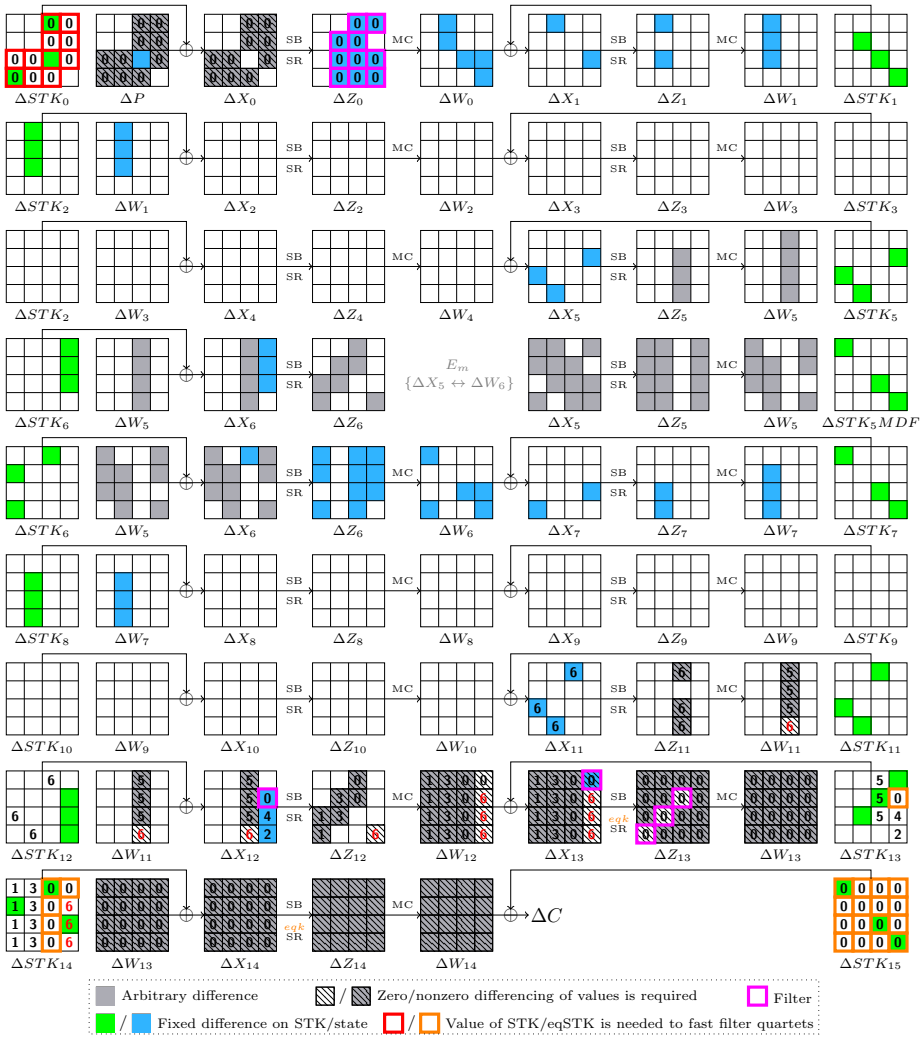

Figure 7: Rectangle attack on 15-round `Deoxys-BC-384`. To maintain symmetry across all rounds, the subteakeys shown for E_f in the figure are not equivalent keys. Nevertheless, the step number is placed according to the position of the equivalent keys.

and $T_4 = 2^{k-l}$. As detailed in Table 6, we obtain an $\epsilon = 1$ table look-up, which is calculated using the pre-computed hash table. In total, there are 6 steps to calculate ϵ. In Step 6, we employ the state-test technique. Throughout all steps, we take into account the independence of the subtweakeys, and utilize the strong key-bridge technique, which models the relationships between subtweakeys to obtain accurate key information. 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.

Table 6: 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 $s = 4, l = 68$, then the data, memory, and time complexities of our attack are $2^{115.7}, 2^{128}, 2^{331.7}$, respectively.

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}.

5.2 34-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]. Since E_m covers many rounds, it is hard to accurately calculate its probability. A common approach is to estimate the probability of E_m empirically, for example, the experimental treatments in [16, 20]. To demonstrate the full capabilities of our model in key recover attacks, we therefore search for optimal attacks on the middle part of the boomerang distinguishers from [16].

For `SKINNY`-64-192, our model shows that a 34-round attack is possible, as illustrated in Figure 8. The boomerang distinguisher used in the attack is displayed in Table 16 and has probability $P^2 = 2^{-53.56}$ when the clustering effect is considered. As can be seen in Figure 8, both the plaintext and the ciphertext are fully active, so $r_b = r_f = 64$; k_b contains 30 cells while k_f has 40 cells. However, $|k_b \cup k_f| = 164$ and $|k_b \cap k_f| = 116$.

We adopt the key recovery algorithm dedicated to the case $r_b = n$ (Sect. A.2 of [45]). We consider an enumeration method for the exhaustive search phase, where the expected number of right pairs is set to one. Then the data complexity

Figure 8: Rectangle attack on 34-round `SKINNY-64-192`

for one related key is $D = 2^{3n/4}/\sqrt{P} = 2^{48+13.39} = 2^{61.39}$ and the overall data complexity is $D_R = 2^{63.39}$. Let $\mu = n - \log_2 D$.

The best attack strategy is to guess 29 cells of k_b and 32 cells of k_f in advance, as marked with red squares in Figure 8. In particular, the 32-cell k'_f, *i.e.*, the last four subtweakeys, are the same as the first four subtweakeys, as the tweakey cells return after 30 rounds, leading to $|k'_b \cup k'_f| = 128$ and $|k'_b \cap k'_f| = 116$. The memory complexity is $M = D_R + D \cdot 2^{r_b^* - \mu} = 2^{64.12}$ to store the data and the pairs constructed. The time complexities of our attack are as follows.

$$
\begin{aligned}
T_0 &= D_R = 2^{63.39},\\
T_1 &= 2^{|k'_b \cup k'_f|} \cdot D_R \cdot \frac{8}{34} = 2^{128+63.39-2.09} = 2^{189.3},\\
T_2 &= 2^{|k'_b \cup k'_f|} \cdot D \cdot 2^{r_b^* - \mu} = 2^{128+61.39+4-2.61} = 2^{190.78},\\
T_3 &= 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r_b^* + 2r_f^* - 2n - 2\mu} \cdot \epsilon = 2^{128+61.39*2+8+48-128-2.61*2} \cdot \epsilon = 2^{173.56} \cdot \epsilon,\\
T_4 &= 2^{k+53.56-64} = 2^{181.56}.
\end{aligned}
$$

To extract the other nine key cells, the complexity ϵ is 1. Therefore, the overall time complexity is $2^{189.3}$ encryptions and $2^{190.78}$ memory accesses. The detailed computation of ϵ proceeds as follows.

1. Derive $STK_{29}[3,7]$. After guessing k'_b, k'_f, $\Delta X_{29}[15]$ is known. Since $\Delta X_{29}[3] = \Delta X_{29}[7] = \Delta X_{29}[15]$, we can derive $Y_{29}[3,7]$ from the known input and output differences of the S-boxes. Then $STK_{29}[3,7] = Y_{29}[3,7] \oplus Z_{29}[3,7]$. As the candidates for $STK_{29}[3,7]$ suggested by the two pairs in a quartet should coincide, there is a $2c$-bit filter and $2^{165.56}$ quartets remain.
2. Similarly, derive $STK_{29}[1]$. There is a c-bit filter and $2^{161.56}$ quartets remain.
3. Guess $STK_4[0]$, which leads to a $2c$-bit filter. The time complexity for this step is $2^{165.56}$ and $2^{157.56}$ quartets remain.
4. Guess $STK_{29}[2]$ and derive $STK_{28}[3]$ and $STK_{29}[5]$. From $STK_{29}[2]$ derive $\Delta X_{28}[15]$. Since $\Delta X_{28}[3] = \Delta X_{28}[11] = \Delta X_{28}[15]$, we can derive $Y_{28}[3,11]$ from the known input and output differences of the S-boxes. Further, $STK_{28}[3] = Y_{28}[3] \oplus Z_{28}[3]$, and $W_{28}[9] = Y_{28}[11]$ which can recover $STK_{29}[5]$. Again, the candidates for the derived key cells suggested by the two pairs in a quartet should coincide, which acts as $2c$-bit filter. The time complexity for this step is $2^{161.56}$ and $2^{153.56}$ quartets remain.
5. Guess $STK_{28}[7]$ and $STK_{29}[4]$ and verify the fixed differece $\Delta X_{27}[9]$. The time complexity for this step is $2^{161.56}$ and $2^{153.56}$ quartets are remaining.

Remark. If we use the same 22-round boomerang distinguisher in [16], 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 key bridges. Specifically, the data, memory, and time complexities of our attack are $2^{62.78}$, $2^{62.79}$, and $2^{154.07}$, respectively.

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 3 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.

References

1. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014-2019), https://competitions.cr.yp.to/caesar.html
2. Information security lightweight cryptography part 2: Block ciphers (2019), includes SKINNY among the standardized block ciphers
3. NIST: Lightweight cryptography competition (2019-2023), https://csrc.nist.gov/projects/lightweight-cryptography
4. Information security encryption algorithms part 7: Tweakable block ciphers (2022), includes Deoxys-TBC and SKINNY as specified TBCs
5. 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, https://doi.org/10.1007/978-3-031-58716-0_10
6. 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, https://doi.org/10.1007/978-3-031-30634-1_1
7. 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: Annual International Cryptology Conference. pp. 123–153. Springer (2016)
8. Biham, E., Dunkelman, O., Keller, N.: The rectangle attackrectangling the Serpent. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 340–357. Springer (2001)
9. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: International Workshop on Fast Software Encryption. pp. 1–16. Springer (2002)
10. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of CRYPTOLOGY **4**(1), 3–72 (1991)
11. 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, https://doi.org/10.1007/978-3-662-45611-8_10
12. 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, https://doi.org/10.13154/tosc.v2017.i3.73-107
13. 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)

14. 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, https://doi.org/10.1007/978-3-662-47989-6_9
15. 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, https://doi.org/10.46586/tosc.v2020.i4.104-129
16. 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, https://doi.org/10.1007/978-3-031-07082-2_1
17. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The SKEIN hash function family (2008)
18. 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, https://doi.org/10.1007/978-3-540-76900-2_21
19. 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, https://doi.org/10.13154/tosc.v2019.i3.43-80
20. 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, https://doi.org/10.46586/tosc.v2021.i2.140-198
21. Hadipour, H., Gerhalter, S., Sadeghi, S., Eichlseder, M.: Improved search for integral, impossible differential and zero-correlation attacks application to Ascon, ForkSKINNY, SKINNY, MANTIS, PRESENT and QARMAv2. IACR Trans. Symmetric Cryptol. **2024**(1), 234–325 (2024). https://doi.org/10.46586/TOSC.V2024.I1.234-325, https://doi.org/10.46586/tosc.v2024.i1.234-325
22. 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, https://doi.org/10.1007/978-3-031-30634-1_5
23. Hirose, S., Minematsu, K.: Compactly committing authenticated encryption using encryptment and tweakable block cipher. In: Carlet, C., Mandal, K., Ri-

jmen, 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, https://doi.org/10.1007/978-3-031-53368-6_12
24. 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, https://doi.org/10.1007/978-981-96-0947-5_11
25. 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, https://doi.org/10.1007/978-3-319-63697-9_2
26. 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, https://doi.org/10.1007/978-3-662-45608-8_15
27. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41. Submitted to CAESAR **124** (2016)
28. 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, https://doi.org/10.1007/978-3-642-32009-5_2
29. 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, https://doi.org/10.1007/3-540-45708-9_3
30. 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, https://doi.org/10.13154/tosc.v2017.i3.37-72
31. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: Santis, A.D. (ed.) Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Lecture Notes in Computer Science, vol. 950, pp. 366–375. Springer (1994). https://doi.org/10.1007/BFb0053451, https://doi.org/10.1007/BFb0053451

32. 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, https://doi.org/10.1007/978-3-662-48116-5_21
33. 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, https://doi.org/10.22042/isecure.2018.114245.405
34. 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, https://doi.org/10.1007/978-3-319-26059-4_9
35. 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, https://doi.org/10.13154/tches.v2020.i1.66-94
36. 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, https://doi.org/10.1007/978-3-662-53018-4_2
37. 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, https://doi.org/10.1007/978-3-540-30539-2_2
38. 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, https://doi.org/10.1007/978-3-319-89339-6_6
39. Schroeppel, R.: The Hasty pudding cipher (1998)
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, 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, 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, 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, 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, https://doi.org/10.1007/978-3-031-22963-3_14
45. 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. IACR Cryptol. ePrint Arch. p. 723 (2022), https://eprint.iacr.org/2022/723
46. Wagner, D.A.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999). https://doi.org/10.1007/3-540-48519-8_12, https://doi.org/10.1007/3-540-48519-8_12
47. 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, https://doi.org/10.13154/tosc.v2019.i1.142-169
48. 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, https://doi.org/10.1007/978-3-662-53887-6_17
49. 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. Journal of Cryptology **37**(2), 1–62 (2024)
50. Zhao, B., Dong, X., Jia, K.: New related-tweakey boomerang and rectangle attacks on Deoxys-BC including BDT effect. IACR Transactions on Symmetric Cryptology pp. 121–151 (2019)
51. 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, https://doi.org/10.1007/978-3-030-35423-7_7

52. 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, https://doi.org/10.1007/s10623-020-00730-1
53. 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, https://doi.org/10.1007/s11432-017-9382-2

Supplementary Materials

A Proof of Propositions 2

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

It can be reduced that

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

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

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

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

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

$$\texttt{Rank}(\boldsymbol{C}) = \texttt{Rank}\left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_{\boldsymbol{2}}^{m} + \boldsymbol{I} & \boldsymbol{L}_{\boldsymbol{2}}^{-m} + \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right),$$

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

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

  Due to the determinant of $\boldsymbol{C}$ is $det(\boldsymbol{C}) = det(\boldsymbol{L}_{\boldsymbol{2}}^{-m}(\boldsymbol{L}_{\boldsymbol{2}}^{m-n} + \boldsymbol{I})(\boldsymbol{L}_{\boldsymbol{2}}^{m} + \boldsymbol{I})(\boldsymbol{L}_{\boldsymbol{2}}^{n} + \boldsymbol{I})) \neq 0$, 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 Model for key pre-guessing allowing the state test

In Algorithm 5, tested states are treated as determined states, as shown in line 4, and contribute to filter generation when combined with other determined states. Since eliminated states always hold determined differentials, they remain independent of the filter acquisition process, and their corresponding subkeys are consequently excluded from guessing. The procedures for determination propagation and filter obtaining follow the established methodology in [44] and are not reiterated here. Finally, the costs of tested states and subkey pre-guessing are considered jointly.

Algorithm 5: Model for key pre-guessing allowing the state test

Input: variables in Table 5, model of distinguisher, extension, and state test.

// the values of ciphertext are determined

1 **for** $c \in \{0..15\}$ **do** $detZ^c_{r.E_f-1} = 1$

2 **for** $r \in \{rE_f\} \setminus \{r.E_f\}$ **do**

3 **for** $c \in \{0, ..., 15\}$ **do**

 // the determination generation, and backward propagation

4 **if** $tW^c_{r-1} = 1$ *or* $detX^c_r = 1$ **then** $detW^c_{r-1} = 1$ **else** $detW^c_{r-1} = 0$ **endif**

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

5 **if** $detZ^{\mathtt{SR}_c^{-1}}_r = 1$ *and* $gEqk^c_r = 1$

6 **then** $detX^c_r = 1$ *and* $fr\mathtt{SB}^c_r = dX^c_r - dZ^{\mathtt{SR}^{-1}}_r$

7 **else** $detX^c_r = 0$ *and* $fr\mathtt{SB}^c_r = 0$ **endif**

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

 // determination propagation (MC), filters from MC

9 **if** $all((detW^{c+i}_r = 1$ *or* $dW^{c+i}_r \leq 1)$ *for* $i \in \{0, ..., 3\})$

10 **then** $(detZ^{c+i}_r = min(detW^{c+j}_r$ *for* $j \in \{0, ..., 3\})$ *for* $i \in \{0, ..., 3\})$

11 *and* $(fr\mathtt{MC}^{c+i}_r = (dZ^{c+i}_r \leq 1)$ *for* $i \in \{0, ..., 3\})$

12 **else** $(fr\mathtt{MC}^{c+i}_r = 0$ *for* $i \in \{0, ..., 3\})$ **endif**

C Other Attacks

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

The longest known distinguisher on Deoxys-BC-256 spans 9 rounds [12], with a probability of $P^2 = 2^{-120.4}$ improved using the BDT technique [47]. 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 12. The whole attack is as shown in Figure 9, where E_b covers 1 round and E_f covers 2 rounds.

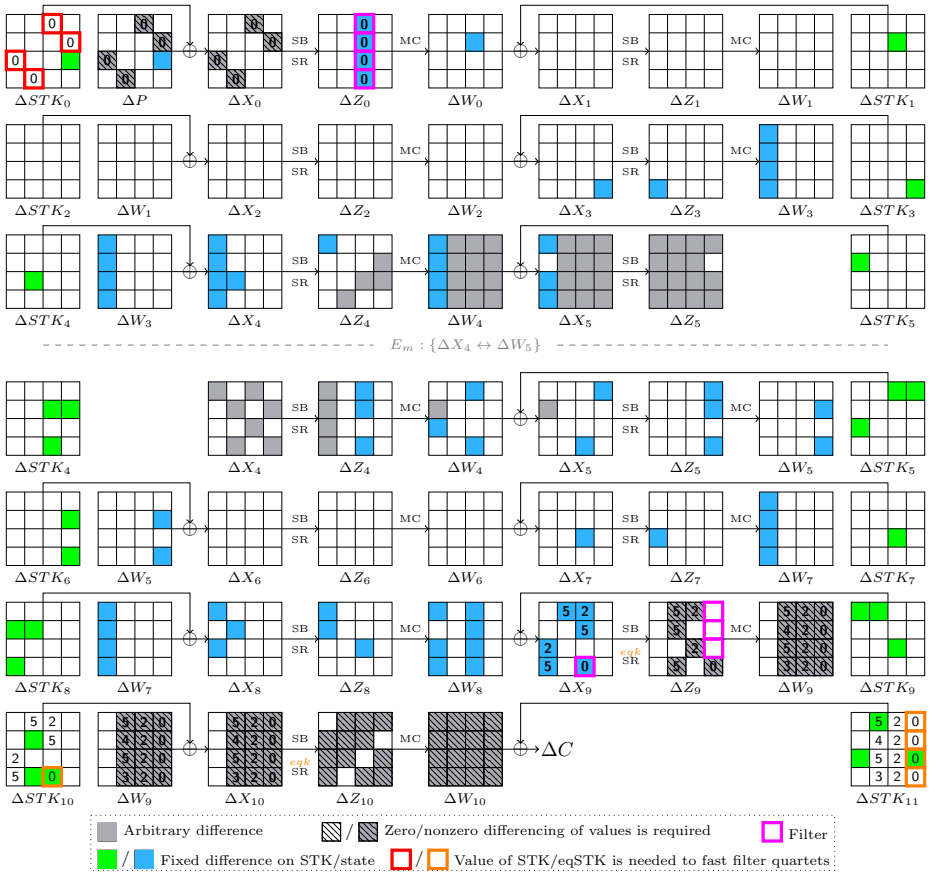

Figure 9: Rectangle attack on 11-round `Deoxys-BC-256`

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 7. 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:

$$T_0 = D_R = \sqrt{s} \cdot 2^{119},$$
$$T_1 = 2^{m'_b+m'_f} \cdot D_R \cdot \frac{2}{11} = \sqrt{s} \cdot 2^{188.5},$$

$$T_2 = 2^{m_b'+m_f'} \cdot D \cdot 2^{r_b^*} = \sqrt{s} \cdot 2^{189},$$
$$T_3 = 2^{m_b'+m_f'} \cdot D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{178},$$
$$T_4 = 2^{k-h}.$$

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 $l = 68$. Then, the data, memory, and time complexities of our attack are 2^{119}, 2^{119}, and $2^{189.8}$, respectively.

Table 7: 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-128-384

For SKINNY-128-384, we obtain a 33-round attack using our model, as shown in Figure 10.

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 10. As detailed in Table 8, 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.

$$T_1 = 2^{|k_b' \cup k_f'|} \cdot D_R = \sqrt{s} \cdot 2^{236+123.54} = \sqrt{s} \cdot 2^{359.54},$$
$$T_2 = 2^{|k_b' \cup k_f'|} \cdot D \cdot 2^{r_b^*} = \sqrt{s} \cdot 2^{336+121.54} = \sqrt{s} \cdot 2^{357.54},$$
$$T_3 = 2^{|k_b' \cup k_f'|} \cdot D^2 \cdot 2^{2r_b^*+2r_f^*-2n} \cdot \epsilon = s \cdot 2^{236+243.09+2\times 64-2\times 128} \cdot \epsilon = s \cdot 2^{351.09} \cdot \epsilon,$$
$$T_4 = 2^{384-l}.$$

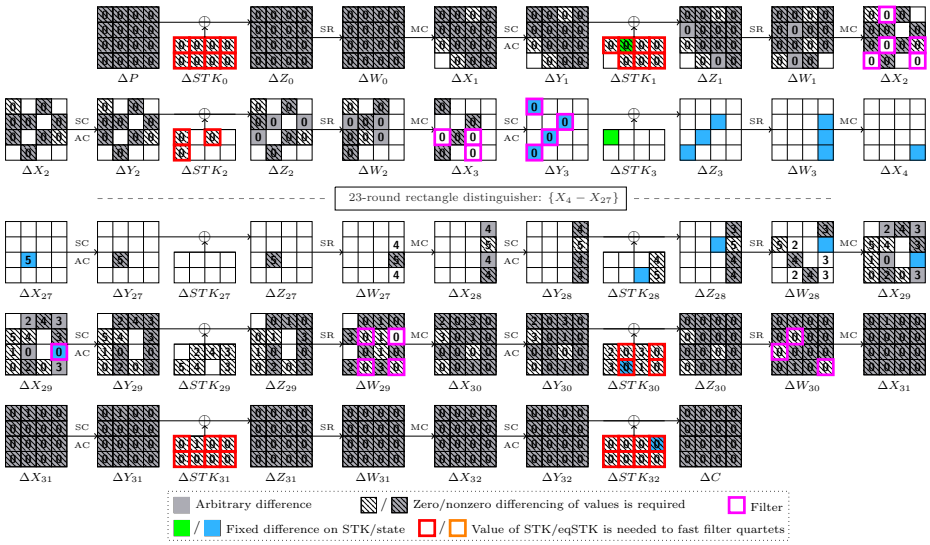

Figure 10: Rectangle attack on 33-round `SKINNY-128-384`

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 $s = 1, l = 40$, then the data, memory and time complexities of our attack are $2^{123.54}, 2^{123.54}, 2^{359.54}$ respectively.

Table 8: 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.3 Improved Attacks on `Deoxys` Authenticated Encryption

Improved 10-Round Attack on `Deoxys-I-128-128` Using our model described in Section 4, we obtain the improved attack on `Deoxys-I-128-128`, which extends the known best attack [12] by one round, as shown in Figure 11.

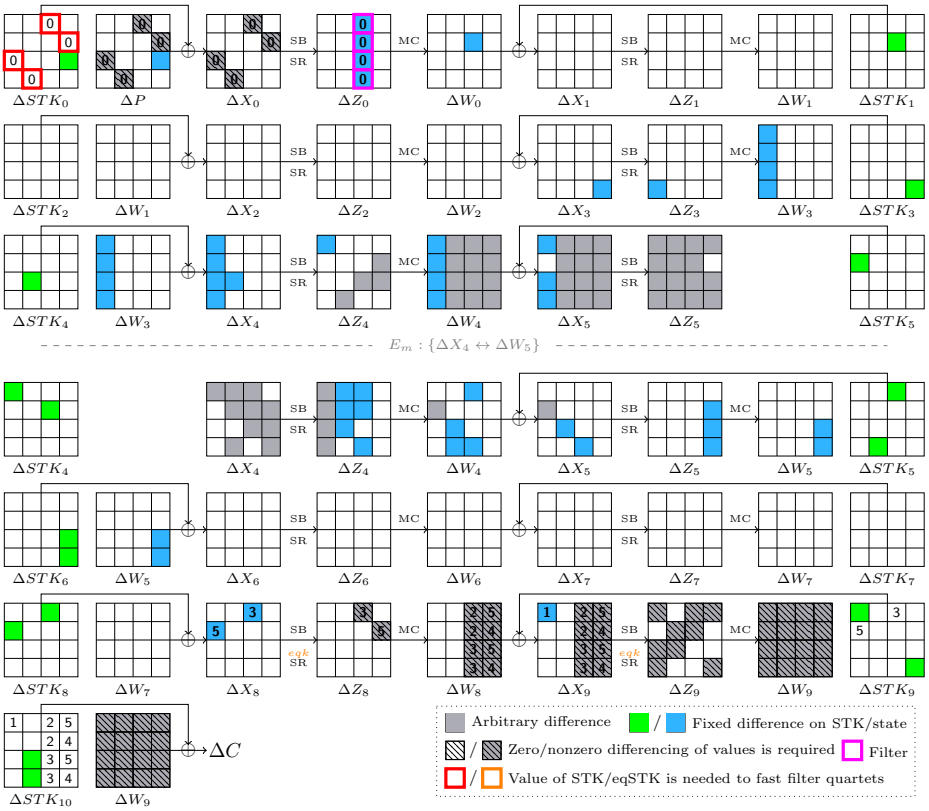

Figure 11: Rectangle attack on 11-round `Deoxys-I-128-128`

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 pre-computation hash table, with the setting of maximum step of calculation is 5, we obtain $\epsilon = 2^{16}$ table look-ups, as detailed in Table 9. Therefore, the time complexity of our attack is as follows:

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{91}, \\
T_1 &= 2^{m'_b+m'_f} \cdot D_R \cdot \frac{2}{10} = \sqrt{s} \cdot 2^{120.7}, \\
T_2 &= 2^{m'_b+m'_f} \cdot D \cdot 2^{r^*_b} = \sqrt{s} \cdot 2^{121}, \\
T_3 &= 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{114}, \\
T_4 &= 2^{k-h}.
\end{aligned}
$$

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 $h = 16$. 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%.

Table 9: Pre-compute hash table for 10-round rectangle attack on `Deoxys-I-128-128`, 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
$\bar{4}$	$Z_9[3,9]$	$eqSTK_{10}[13,15]$	-	$2^{6\times 8}$	$2^{2\times 8}$
$\bar{5}$	$Z_8[13]$, $Z_9[6,12]$	$eqSTK_9[1]$, $eqSTK_{10}[12,14]$	$\Delta X_8[1]$, $\Delta Z_8[12,14,15]$	$2^{5\times 8}$	$2^{1\times 8}$

Improved 14-Round Attack on `Deoxys-I-256-128` Using our full-fledged model described in Section 4, we obtain the improved attack on `Deoxys-I-256-128`, as is shown in Figure 12. 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 pre-computation hash table, with the number of steps set to 5, we obtain $\epsilon = 1$ table look-up, as detailed in Table 10. Therefore, the time complexity of our attack is as follows:

$$
\begin{aligned}
T_0 &= D_R = \sqrt{s} \cdot 2^{115.1}, \\
T_1 &= 2^{m_b'+m_f'} \cdot D_R \cdot \frac{3}{14} = \sqrt{s} \cdot 2^{232.9}, \\
T_2 &= 2^{m_b'+m_f'} \cdot D \cdot 2^{r_b^*} = \sqrt{s} \cdot 2^{233.1}, \\
T_3 &= 2^{m_b'+m_f'} \cdot D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{234.2}, \\
T_4 &= 2^{k-h}.
\end{aligned}
$$

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 $h = 32$. 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%.

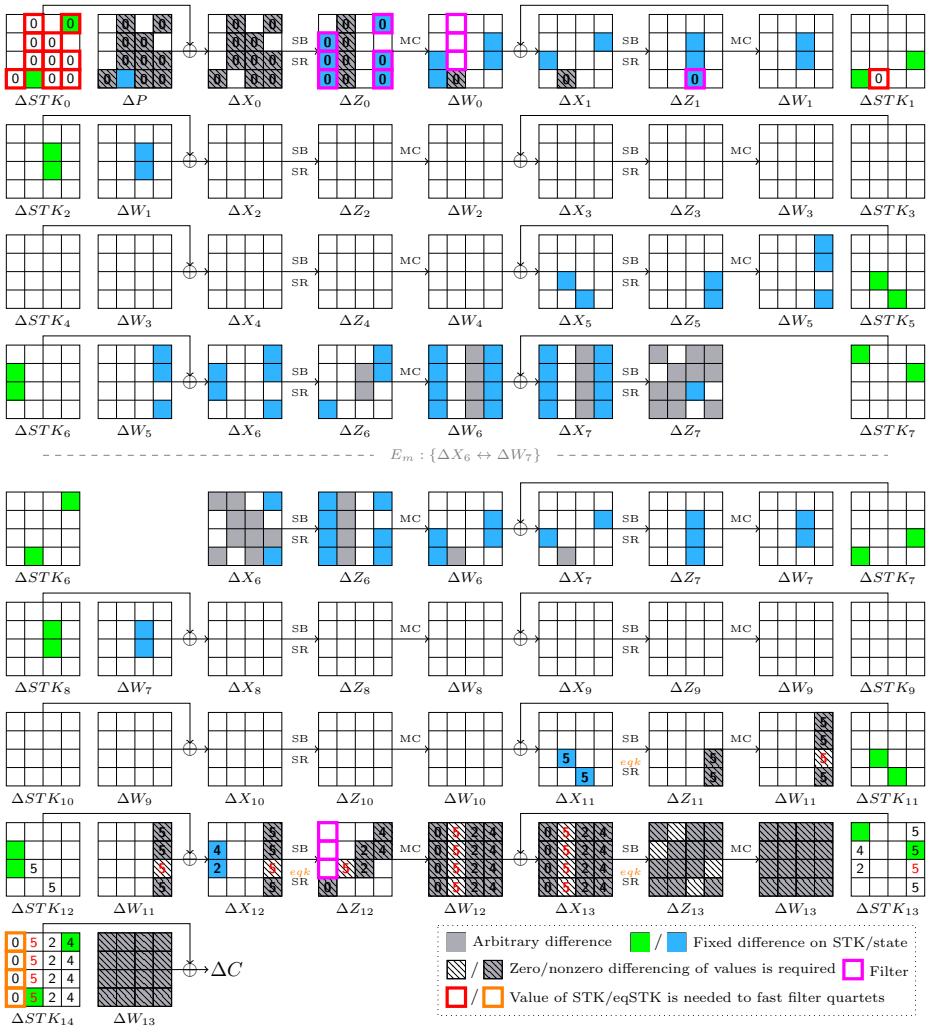

Figure 12: Rectangle attack on 14-round `Deoxys-I-256-128`

Table 10: Pre-compute hash table for 10-round rectangle attack on `Deoxys-I-256-128`.

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	$Z_{12}[3,9,12]$	$eqSTK_{13}[12,13,15]$, $eqSTK_{12}[6,11]$, $W_{11}[14]$	$\Delta X_{11}[6,11]$, $\Delta Z_{11}[14,15]$	$2^{11\times 8}$	$2^{-3\times 8}$

D Boomerang Distinguishers of Deoxys-BC and SKINNY

Table 11: 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 12: 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 00	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 ** 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 13: A 10-round related-tweakey boomerang distinguisher used to attack 15-round `Deoxys-BC`-384 in Section 5.1. 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 14: 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 15: A 23-round related-tweakey boomerang distinguisher for SKINNY-128-384 proposed in [16], 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 16: 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	*,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