Optimizing Key Recovery in Impossible Cryptanalysis and Its Automated Tool

Jianing Zhang and Haoyang Wang

Shanghai Jiao Tong University, Shanghai, China.

*Corresponding author(s). E-mail(s): haoyang.wang@sjtu.edu.cn;
Contributing authors: zhangjn@sjtu.edu.cn;

Abstract

Impossible differential (ID) cryptanalysis and impossible boomerang (IB) cryptanalysis are two methods of impossible cryptanalysis against block ciphers. Since the seminal work introduced by Boura *et al.* in 2014, there have been no substantial advancements in the key recovery process for impossible cryptanalysis, particularly for the IB attack. In this paper, we propose a generic key recovery framework for impossible cryptanalysis that supports arbitrary key-guessing strategies, enabling optimal key recovery attacks. Within the framework, we provide a formal analysis of probabilistic extensions in impossible cryptanalysis for the first time. Besides, for the construction of IB distinguishers, we propose a new method for finding contradictions in multiple rounds.

By incorporating these techniques, we propose an Mixed-Integer Linear Programming (MILP)-based tool for finding full ID and IB attacks. To demonstrate the power of our methods, we applied it to several block ciphers, including SKINNY, SKINNYee, Midori, and Deoxys-BC. Our approach yields a series of optimal results in impossible cryptanalysis, achieving significant improvements in time and memory complexities. Notably, our IB attack on SKINNYee is the first 30-round attack.

Keywords: Impossible differential cryptanalysis, Impossible boomerang cryptanalysis, SKINNY, SKINNYee, Midori, Deoxys-BC

1 Introduction

The impossible differential (ID) cryptanalysis was first independently introduced by Knudsen [1] and Biham [2]. As an important variant of differential cryptanalysis [3], the ID cryptanalysis uses a differential with a probability of zero to eliminate incorrect keys. Research on impossible differential attacks primarily focuses on two objectives: constructing ID distinguishers and mounting key recovery attacks.

The miss-in-the-middle technique [4] is one of the main methods for identifying ID distinguishers. This approach involves identifying two differences that propagate through the cipher, one forward and one backward, with certainty, but conflict at the meeting point. For automated tools, Cui *et al.* [5] first proposed a Mixed-Integer Linear Programming (MILP) model in 2016 to search for ID distinguishers. In 2017, Sasaki and Todo [6] proposed another MILP-based tool targeting a broader range of block ciphers. In [7], Sun *et al.* developed a constraint programming (CP)-based automatic model to search for related-key ID distinguishers in several SPN ciphers.

Regarding the key recovery attacks, Lu *et al.* introduced the early-abort technique in [8], which divides the key sieving phase into sequential steps. Boura *et al.* [9, 10] proposed a general framework for formalizing the key recovery process and provided a systematic complexity analysis within this framework. The previous automated tools for ID cryptanalysis only targeted the ID distinguisher, in 2016, Derbez and Fouque firstly [11] developed a computer-aided tool to search for a complete ID attack including the key recovery phase. More recently, Hadipour *et al.* [12, 13] developed and refined a CP-based tool for searching complete ID attacks. Additionally, while ID distinguishers are typically extended with probability 1, some works [8, 14–16] have explored probabilistic extension in their ID attacks. However, this approach has not been systematically studied. Recently, Song *et al.* [17] incorporated the meet-in-the-middle technique into the key recovery of ID cryptanalysis, which is typically useful when the key size is at least the twice the block size.

The impossible boomerang (IB) attack is another impossible cryptanalysis by combining the concepts of ID attacks and boomerang attacks, first introduced by Lu in his PhD thesis [18] and later published in 2011 [19]. The IB attack relies on an IB distinguisher, that is a boomerang distinguisher with probability zero. Since its introduction in 2008, the IB attack had not received sufficient attention until 2024, when several studies revisited this topic [20–22]. In [20, 21], the authors independently explored the construction of contradictions using boomerang tools. They also proposed two similar key recovery methods, along with a satisfiability modulo theories (SMT)-based tool and a mixed-integer quadratically-constrained programming (MIQCP)-based tool, respectively. In [22], Hu *et al.* provided a comprehensive theoretical analysis of the construction of IB distinguishers, and introduced a Boolean satisfiability problem (SAT)-based tool for searching IB distinguishers. Recently, the authors of [23] proposed a graph-based key-recovery technique and mounted the first full-round impossible boomerang attack on ARADI.

Contributions. We propose a new generic key recovery framework for ID and IB attacks that incorporates arbitrary key-guessing strategies. Notably, it is the first generic key recovery framework for IB attacks, covering the two specific key recovery methods introduced in [20, 21]. Our framework is proven to be effective in improving the time and memory complexities. Besides, we provide a first systematic analysis of probabilistic extensions in impossible key recovery attacks, which also proves effective in improving the time and memory complexities. On the other hand, to complete the construction of IB distinguishers, we introduce a new approach called `iUBCT`/`iLBCT` for identifying contradiction in multiple rounds, which can significantly reduce the computational complexity compared to the existing methods. With the help of this new approach, we obtained the upper bound of multi-rounds contradiction for the Sboxes of SKINNY and Deoxys-BC. Specifically, for SKINNY-128-384, we obtained a 19-round distinguisher based on `iLBCT`, which achieves one round more than the previously best impossible distinguisher.

Combining all the techniques introduced, we aim to achieve the optimal impossible cryptanalytic results for block ciphers. We propose a new generic MILP-based model to find full ID and IB attacks. To show the usefulness of our method, we apply it to SKINNY, SKINNYee, Midori, and Deoxys-BC. For SKINNYee, we obtained the first 30-round related-tweak IB attack, which is the best third-party cryptanalytic result reported on SKINNYee to date, improving upon the previous best 29-round attack in terms of data, time, and memory complexity. Tables 1 and 2 summarizes our results.

Table 1: Overview of our impossible differential cryptanalytic results. STK=Single-tweakey. SK=Single-key. RTK=Related-tweakey. Int.=Integral. MITM=Meet-in-the-Middle.

Target	Setting	#Rounds	Data	Time	Memory	Ref.
SKINNY-64-64	RTK	19	$2^{61.47}$	$2^{63.03}$	2^{56}	[24]
	RTK	19	$2^{61.47}$	$\mathbf{2^{62.76}}$	$\mathbf{2^{52}}$	H.2
SKINNY-128-128	RTK	19	$2^{122.47}$	$2^{124.60}$	2^{112}	[24]
	RTK	19	$2^{122.47}$	$\mathbf{2^{124.43}}$	$\mathbf{2^{104}}$	H.2
SKINNY-64-128	STK	19	$2^{60.86}$	$2^{110.34}$	2^{104}	[12]
	STK	19	$2^{65.05}$	$\mathbf{2^{104.90}}$	$\mathbf{2^{68.05}}$	H.3
	Int.	22	2^{58}	2^{106}	2^{104}	[13]
SKINNY-128-256	STK	19	$2^{117.86}$	$2^{219.23}$	2^{208}	[12]
	STK	19	$2^{126.05}$	$\mathbf{2^{209.45}}$	$\mathbf{2^{133.05}}$	H.3
	Int.	22	2^{114}	2^{213}	2^{208}	[13]
SKINNY-64-192	STK	21	$2^{62.43}$	$2^{174.42}$	2^{168}	[12]
	STK	21	$2^{64.99}$	$\mathbf{2^{169.38}}$	$\mathbf{2^{103.99}}$	H.4
	Int.	26	2^{62}	2^{166}	2^{164}	[13]
SKINNY-128-384	STK	21	$2^{122.89}$	$2^{347.35}$	2^{336}	[12]
	STK	21	$2^{125.99}$	$\mathbf{2^{338.65}}$	$\mathbf{2^{204.99}}$	H.4
	Int.	26	2^{122}	2^{331}	2^{328}	[13]
Midori64	SK	11	2^{60}	$2^{116.59}$	2^{92}	[25]
	SK	11	$2^{61.33}$	$\mathbf{2^{99.94}}$	$\mathbf{2^{56.33}}$	I.2
	MITM	12	$2^{55.5}$	$2^{125.5}$	2^{109}	[26]

Table 2: Overview of our impossible boomerang cryptanalytic results. RT=Related-tweak. Boom.=Boomerang. Rect.=Rectangle.

Target	Setting	#Rounds	Data	Time	Memory	Ref.
SKINNYee	RT	29	$2^{67.2}$	$2^{119.2}$		[20]
	RT	**30**	$2^{67.03}$	$2^{123.61}$	$2^{58.05}$	Section 6
Deoxys-BC-256	RTK	10	$2^{132.8}$	$2^{186.66}$	$2^{181.6}$	[21]
	RTK	10	$2^{132.9}$	$\mathbf{2^{177.42}}$	$\mathbf{2^{101.79}}$	J.2
	Boom.	11	$2^{122.4}$	$2^{218.65}$	2^{128}	[27]
Deoxys-BC-384	RTK	14	$2^{130.9}$	2^{368}	2^{320}	[21]
	RTK	14	$2^{132.41}$	$\mathbf{2^{343.05}}$	$\mathbf{2^{132.83}}$	J.3
	Rect.	15	$2^{115.7}$	$2^{371.7}$	2^{128}	[28]

- We improve the 19-round related-tweakey ID attack on SKINNY-n-n. We discover a new 12-round ID distinguisher and reduce the memory complexity of the attack with this new distinguisher. To the best of our knowledge, this is the best attack against SKINNY-n-n under the related-tweakey setting.
- We improve the single-tweakey ID attack on SKINNY-n-2n and SKINNY-n-3n. By using the same distinguisher as previously reported and thanks to our new generic key recovery framework, we improved the time and memory complexity of the attack.
- We improve the 11-round single-key impossible differential attack on Midori64 by leveraging the previously reported distinguisher and our generic key recovery framework, achieving improvements in both time and memory complexities.
- For Deoxys-BC, we improve the related-tweakey IB attacks on 10-round Deoxys-BC-256 and 14-round Deoxys-BC-384 in both time and memory complexities.

Outline. We introduce the background in Section 2, including the ID cryptanalysis and the IB cryptanalysis. In Section 3, we give an overview of the generic key guessing strategy, then propose the generic key recovery frameworks for IB and ID attacks, providing a detailed description of the attack procedure and complexity analysis. In Section 4, we provide a formal analysis of probabilistic extensions in impossible cryptanalysis for the first time, and introduce two dedicated tables `iUBCT` and `iLBCT` for IB cryptanalysis. An automated search tool is introduced in Section 5. Finally, we provide the applications in Section 6 and summarize this paper in Section 7.

2 Preliminaries

Impossible cryptanalysis, including ID and IB, exploits differential/boomerang distinguishers with probability 0 and discards the key candidates leading to such impossible distinguishers.

Impossible cryptanalysis consists mainly of two steps. The first step is to identify an impossible distinguisher (Δ_X, Δ_Y) such that the input difference Δ_X propagates to the output difference Δ_Y with probability 0. Then, the second step deals with the key recovery by extending the impossible distinguisher by some rounds backward and forward. Any key candidate involved in the extended rounds that allows a given pair/quartet of data to satisfy the input and output of the impossible distinguisher is eliminated.

The parameters of impossible cryptanalysis are illustrated in Figure 1. The impossible distinguisher is denoted by E_D, the input difference Δ_X represents a state difference and a pair of state differences for ID and IB, respectively (same for Δ_Y). The difference Δ_X (resp. Δ_Y) propagates backward (resp. forward) through E_B^{-1} (resp. E_F) to Δ_B (resp. Δ_F) with probability 1. The notation c_B (resp. c_F) is the number of bit-conditions that should be verified for the transition from Δ_B to Δ_X (resp. from Δ_F to Δ_Y). We use k_B and k_F to represent the subkey bits involved in E_B and E_F, respectively. Let r_B and r_F be the dimension of vector spaces Δ_B and Δ_F, respectively.

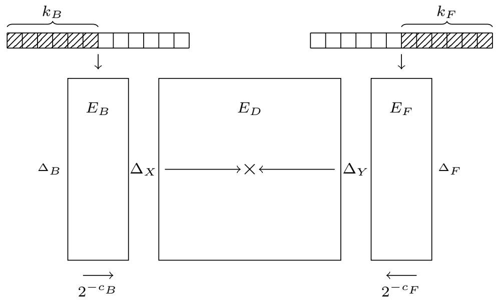


Fig. 1: Outline of impossible cryptanalysis.

2.1 Impossible Differential Cryptanalysis

Impossible differential cryptanalysis was independently proposed by Knudsen [1] and Biham [2]. The ID attack can be divided into three steps, and the complexity has been carefully analyzed in [9] and [10], which are summarized below.

Pairs Generation. Generate N pairs of data, for each of them its plaintext difference is in Δ_B and its ciphertext difference is in Δ_F. The data complexity as well as the time complexity of this step is given by

$$D = \max\left\{ \min_{r_x \in \{r_B, r_F\}} \left\{ \sqrt{N 2^{n+1-r_x}} \right\}, N 2^{n+1-r_B-r_F} \right\}.$$

Guess-and-Filter. For each of the N pairs, discard the subkeys in $k_B \cup k_F$ that generates the input difference Δ_X and the output difference Δ_Y. With the early abort technique [29], we can guess the subkeys step by step. Thus, the time complexity can be bounded by

$$T_1 + T_2 = N + 2^{|k_B \cup k_F|} \frac{N}{2^{c_B+c_F}}.$$

Note that this is the minimum number of partial encryptions/decryptions, which could not be achieved in practice.

Exhaustive Search. Exhaustively search the remaining key candidates after sieving. The probability that a trial key is kept in the candidate keys is

$$P = (1 - 2^{-(c_B+c_F)})^N.$$

Thus, the time complexity is $T_3 = P \cdot 2^{|k_B \cup k_F|} \cdot 2^{k-|k_B \cup k_F|} = 2^k \cdot P$.

The memory complexity is

$$M = \min\{N, 2^{|k_B \cup k_F|}\}$$

for storing the N pairs or the discarded key candidates.

In the rest of the paper, we will adopt another complexity representation introduced in [12]. Let g be the number of key bits we can retrieve through the guess-and-filter phase, i.e., $P = 2^{-g}$, $1 < g \le |k_B \cup k_F|$. With the approximation $(1 - 2^{-(c_B+c_F)})^N \approx e^{-N \cdot 2^{-(c_B+c_F)}}$, we have $N = 2^{c_B+c_F+\log_2(g)-0.53} = 2^{c_B+c_F+LG(g)}$. Thus, the complexity analysis of the ID attack can be reformulated as follows:

$$D = \max\left\{\min_{r_x \in \{r_B, r_F\}} \left\{\sqrt{2^{c_B+c_F+n+1+LG(g)-r_x}}\right\}, 2^{c_B+c_F+n+1+LG(g)-r_B-r_F}\right\}, \quad (1)$$

$$T_1 = 2^{c_B+c_F+LG(g)}, T_2 = 2^{|k_B \cup k_F|+LG(g)}, T_3 = 2^{k-g}, \quad (2)$$

$$M = \min\{2^{c_B+c_F+LG(g)}, 2^{|k_B \cup k_F|}\}. \quad (3)$$

2.2 Impossible Boomerang Cryptanalysis

The impossible boomerang (IB) cryptanalysis, first introduced by Lu in [18, 19], combines the concepts of boomerang attacks and ID attacks. It relies on a boomerang distinguisher with probability 0, referred to as the impossible boomerang distinguisher, which is defined as follows.

Definition 1. *Suppose $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ is a block cipher and $K \in \{0,1\}^k$ is a key for E. If there exists a quartet of n-bit blocks $(\alpha, \alpha', \delta, \delta')$ satisfying*

$$\forall X \in \mathbb{F}_2^n, \; Pr[E_K^{-1}(E_K(X) \oplus \delta) \oplus E_K^{-1}(E_K(X \oplus \alpha) \oplus \delta') = \alpha'] = 0,$$

then the combination of $(\alpha, \alpha', \delta, \delta')$ is called an impossible boomerang distinguisher for E, written $(\alpha, \alpha') \nrightarrow (\delta, \delta')$.

To build an IB distinguisher, following the notations in Figure 2, we need two differentials $\alpha_0 \to \alpha_1$ and $\alpha'_0 \to \alpha'_1$ with probability 1 over E_0, and two differentials $\beta_2 \to \beta_1$ and $\beta'_2 \to \beta'_1$ with probability 1 over E_1^{-1}. Then, the condition that the four differences in the middle satisfy $\alpha_1 \oplus \alpha'_1 \oplus \beta_1 \oplus \beta'_1 \neq 0$ establishes an IB distinguisher. In a very recent work [22], the authors use the same approach to identify IB distinguishers.

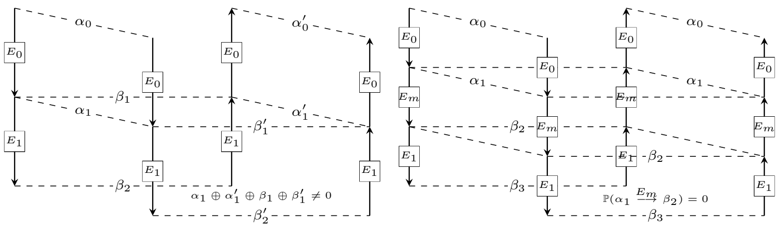


Fig. 2: Constructions of impossible boomerang distinguishers: using four different differentials introduced in [19, 22] (left) and a specific case using two different differentials adopted in [20, 21] (right).

On the other hand, an IB distinguisher can be built by two probability-1 differentials, where both sides of the boomerang employ the same differential (Figure 2). This approach is common in practice as demonstrated in the two recent works [20, 21]. Moreover, they also studied the possibility of extending contradictions to a middle layer E_m using boomerang tools, such as BCT [30] and DBCT [31, 32]. The most straightforward way is to use $\mathtt{BCT} = 0$ to ensure that the switching probability of single-round E_m is 0. As for the contradictions in multiple-rounds, [20, 21, 33] introduced the applications and limitations of the DBCT, which will be discussed in Section 4.2.

Similar to the ID attack, the IB key recovery attack also consists of three phases, where the difference is that it deals with quartets instead of pairs. Additionally, the two works [20, 21] independently proposed two methods for IB key recovery attacks. Details can be found in Appendix B.

3 Generic Key Recovery Framework for Impossible Cryptanalysis

In this section, we will introduce a more generic key guessing strategy into the key recovery attack for impossible cryptanalysis. Base on the new strategy, we propose the generic key recovery frameworks and complexity analysis for ID attacks and IB attacks separately.

3.1 Generic Key Guessing Strategy

In impossible cryptanalysis, the number of pair (quartet) candidates primarily determines the time complexity of the guess-and-filter phase and the memory complexity in most cases, as described in Section 2.1. Inspired by the optimized rectangle attack introduced in [27, 34], where most quartets can be filtered out before being constructed by guessing some subkey bits, we explore whether a similar approach could be applied to reduce the number of pair (quartet) candidates in impossible cryptanalysis.

Indeed, the recent works [20, 21] have applied similar ideas to IB attacks, independently proposing a key recovery method where the subkey bits k_B or k_F are guessed before the quartets generation, please refer to Appendix B for further details. This approach enables the construction of pairs satisfying the input or output conditions of the IB distinguisher, which are subsequently used to form quartets. However, rather than guessing full k_B or full k_F or both, a more generic key guessing strategy can be explored, involving partial guesses of k_B and k_F, denoted as k'_B or k'_F, respectively. Assuming that a c'_B-bit condition and a c'_F-bit condition can be verified in E_B and E_F under this partial guess, let $r^*_B = r_B - c'_B$ and $r^*_F = r_F - c'_F$. Consequently, the number of pairs (resp. quartets) to be processed becomes $N = 2^{c^*_B + c^*_F + LG(g)}$ (resp. $2^{2c^*_B + 2c^*_F + LG(g)}$), where $c^*_B = c_B - c'_B$ and $c^*_F = c_F - c'_F$. However, the time complexity for processing these candidates becomes $2^{|k'_B \cup k'_F|} \cdot N$, thus the time complexity is reduced only if $|k'_B \cup k'_F| < c'_B + c'_F$ for ID attacks (resp. $|k'_B \cup k'_F| < 2c'_B + 2c'_F$ for IB attacks). Additionally, this key guessing strategy could also affect memory and data complexities, which requires further analysis.

By incorporating the key guessing strategy, we propose a generic key recovery framework for impossible cryptanalysis and complexity analysis in the next two subsections.

3.2 Generic Key Recovery Framework for Impossible Differential Attacks

In the following, we propose the generic key recovery framework for ID attacks incorporating the generic key guessing strategy.

1. Prepare 2^y structures, each consists of 2^{r_B} plaintexts, and query their ciphertexts. The data complexity is $D = 2^{y + r_B}$.
2. Guess $|k'_B \cup k'_F|$ bits of the involved subkeys $k_B \cup k_F$.
   (a) For each data (P, C), partially encrypt P and partially decrypt C under the guessed subkey bits: $P^* = E_B(k'_B, P)$ and $C^* = E_F^{-1}(k'_F, C)$. For each structure, we will get $2^{c'_B}$ sub-structures, each of which contains $2^{r^*_B}$ plaintexts.
   (b) For each sub-structure, insert C^* into a hash table according to $n - r^*_F$ bits of C^*. We will get
   $$N = 2^y \cdot 2^{c'_B} \cdot 2^{2r^*_B - 1} \cdot 2^{r^*_F - n} = D \cdot 2^{r^*_B + r^*_F - n - 1}$$
   pairs satisfying the c'_B- and c'_F-bit conditions.
   (c) Guess the remaining subkey bits $|k^*_B \cup k^*_F| = |k_B \cup k_F| - |k'_B \cup k'_F|$ involved in E_B and E_F that encrypt/decrypt a data pair satisfying the c^*_B and c^*_F bit-conditions, respectively, and eliminate the guess of $k_B \cup k_F$.
3. Exhaustively search the remaining candidates of $k_B \cup k_F$ and the unknown subkey bits of $k \backslash (k_B \cup k_F)$.

The probability that a wrong key survives through step (c) is $P = (1 - 2^{-(c^*_B + c^*_F)})^N$. Let $P = 2^{-g}$, we have $N = 2^{c^*_B + c^*_F + log_2(g) - 0.53} = 2^{c^*_B + c^*_F + LG(g)}$.

Data Complexity. The data complexity can be divided into two cases.

- When multiple structures are used, the data complexity is

$$D = N \cdot 2^{n-r_B^*-r_F^*+1} = 2^{n+c_B+c_F-r_B-r_F+LG(g)+1}.$$

  Thus, the key guessing strategy does not affect the data complexity in this case.
- If less than one structure is used, let 2^{x_1} be the number of sub-structures, each contains 2^{x_2} data on average. Then $N = 2^{x_1} \cdot 2^{2x_2-1} \cdot 2^{r_F^*-n}$, which means that the data complexity is

$$\begin{aligned} D = 2^{x_1+x_2} &= \sqrt{N \cdot 2^{n-r_F^*+x_1+1}} \\ &= 2^{\frac{c_B+c_F+n+1-r_F+LG(g)+x_1-c'_B}{2}}. \end{aligned} \tag{4}$$

Similarly, we can also launch the attack from the ciphertext side, the complexity analysis is similar and is omitted here.

Time Complexity. The ID attacks can be divided into three phases:

- *Pairs Collection.* This phase consists of steps 1, 2(a) and 2(b). The time complexity includes D encryptions and $T_0 = 2^{|k'_B \cup k'_F|} \cdot D$ partial encryptions/decryptions.
- *Guess-and-Filter.* This phase consists of step 2(c). A pair satisfying the input difference Δ_X and output difference Δ_Y of the distinguisher needs to verify $c_B^* + c_F^*$ bit-conditions. The time complexity for this phase could be closely approximately by

$$\begin{aligned} T_1 + T_2 &= 2^{|k'_B \cup k'_F|} \cdot (N + 2^{|k_B^* \cup k_F^*|} \cdot \frac{N}{2^{c_B^*+c_F^*}}) \\ &= 2^{|k'_B \cup k'_F|+c_B^*+c_F^*+LG(g)} + 2^{|k_B \cup k_F|+LG(g)}. \end{aligned} \tag{5}$$

- *Exhaustive Search.* This phase consists of step 3 and the time complexity of this phase would be

$$T_3 = 2^{k-|k_B \cup k_F|} \cdot P \cdot 2^{|k_B \cup k_F|} = 2^{k-g}.$$

In summary, the time complexity of the ID attack is:

$$T = (D + (T_0 + T_1 + T_2)C_{E'} + T_3)C_E,$$

where C_E denotes the cost of one encryption, and $C_{E'}$ is the ratio of the cost for one partial encryption to the full encryption.

Memory Complexity. The memory complexity would be

$$M = \min\{2^{c_B^*+c_F^*+LG(g)}, 2^{|k_B \cup k_F|}\} \tag{6}$$

for storing the N pairs or the discarded key candidates.

For the ID attack in the related-(twea)key setting, we refer to Appendix C.

Advantages. We analyze the advantages of the generic key recovery framework as follows.

- *Time.* Note that T_1 corresponds directly to the total number of pairs used during the guess-and-filter phase, it can be observed that when $|k'_B \cup k'_F| < c'_B + c'_F$, the term T_1 in Equation (5) will be lower than the classical one in Equation (2), which has the potential to reduce the overall time complexity of ID attacks. However, the condition $|k'_B \cup k'_F| < c'_B + c'_F$ is rarely satisfied in most ID attacks, limiting its impact, though special cases may exist where it proves advantageous.
- *Data.* In the case when less one structure is used in ID attacks, we have $x_1 - c'_B \leq 0$ in Equation (4), which means the data complexity could be potentially reduced compared to the classical one in Equation (1).
- *Memory.* In practice, the inequality $N < 2^{|k_B \cup k_F|}$ holds in most cases. With the key guessing strategy, we have $c^*_B + c^*_F < c_B + c_F$, which means that the memory complexity can be effectively reduced. The effectiveness can be exemplified by the attacks on SKINNY and Midori64 in Appendices H.3, H.4 and I.2.

3.3 Generic Key Recovery Framework for Impossible Boomerang Attacks

In the single-key scenario, an IB attack can be transformed into an ID attack, as explained by [19], which renders it less interesting. On the other hand, in the related-key scenario, IB attacks might have advantages over ID attacks due to the flexibility in choosing related keys, as explained by [21]. Therefore, in this subsection, we present the generic key recovery method for SPN block ciphers with a linear key schedule under the related-key setting.

1. Prepare 2^y structures, each consists of 2^{r_B} plaintexts. Query their ciphertexts for each structure under the 4 related keys K_i, and the corresponding plaintext-ciphertext sets are L_i, $i \in \{1, 2, 3, 4\}$. Let $D' = 2^{y+r_B}$.
2. Guess $|k'_B \cup k'_F|$ bits of the involved subkeys $k_B \cup k_F$[1].
   (a) For each (P_i, C_i) in L_i, partially encrypt P_i and partially decrypt C_i under the guessed subkey bits: $P^*_i = E_B(k'_B, P_i)$ and $C^*_i = E_F^{-1}(k'_F, C_i)$. For each structure under K_i, we will get $2^{c'_B}$ sub-structures, each of which contains $2^{r^*_B}$ plaintexts.
   (b) If $2^{r^*_B} \leq D' \cdot 2^{r^*_F - n}$, go to step (i); else go to step (iv).
      i. Construct two sets as

$$S_1 = \{(P^*_1, C^*_1, P^*_2, C^*_2) \mid P^*_1 \text{ and } P^*_2 \text{ have difference in } r^*_B \text{ bits}\}.$$

$$S_2 = \{(P^*_3, C^*_3, P^*_4, C^*_4) \mid P^*_3 \text{ and } P^*_4 \text{ have difference in } r^*_B \text{ bits}\}.$$

The size of each set is $2^y \cdot 2^{c'_b} \cdot 2^{2r^*_B} = D' \cdot 2^{r^*_B}$.

[1]Since the key schedule is linear, we do not differentiate between the guessed subkeys of each key.

ii. Insert S_1 into a hash table indexed by the $n - r_F^*$ inactive bits of both C_1^* and C_2^*. Insert S_2 into a hash table indexed by the $n - r_F^*$ inactive bits of both C_3^* and C_4^*.

iii. For each $2(n - r_F^*)$-bits index, we pick two distinct entries $(P_1^*, C_1^*, P_2^*, C_2^*)$, $(P_3^*, C_3^*, P_4^*, C_4^*)$ to generate quartets. We will get

$$Q = D'^2 \cdot 2^{2r_B^* + 2r_F^* - 2n}$$

quartets. Then go to step (c).

iv. Construct two sets as

$$S_3 = \{(P_1^*, C_1^*, P_3^*, C_3^*) \mid C_1^* \text{ and } C_3^* \text{ are colliding in } n - r_F^* \text{ bits}\}.$$
$$S_4 = \{(P_2^*, C_2^*, P_4^*, C_4^*) \mid C_2^* \text{ and } C_4^* \text{ are colliding in } n - r_F^* \text{ bits}\}.$$

The size of each set is $D'^2 \cdot 2^{r_F^* - n}$.

v. Insert S_3 into a hash table indexed by the $n - r_B^*$ inactive bits of both P_1^* and P_3^*. Insert S_4 into a hash table indexed by the $n - r_B^*$ inactive bits of both P_2^* and P_4^*.

vi. Since the data are generated from $2^{y + c_B'}$ sub-structures of $2^{r_B^*}$ plaintexts each, the $2(n - r_B^*)$-bit index has $(D' \cdot 2^{-n})^2 \cdot 2^{2(n - r_B^*)}$ values at most. For each index, we pick two distinct entries $(P_1^*, C_1^*, P_3^*, C_3^*)$ and $(P_2^*, C_2^*, P_4^*, C_4^*)$ to construct quartets. The number of quartets is

$$Q = D'^2 \cdot 2^{2r_B^* + 2r_F^* - 2n}.$$

(c) Guess the remaining subkey bits $|k_B^* \cup k_F^*| = |k_B \cup k_F| - |k_B' \cup k_F'|$ involved in E_B and E_F that encrypt/decrypt a data quartet satisfying the $2c_B^*$ and $2c_F^*$ bit-conditions, respectively, and eliminate the guess of $k_B \cup k_F$.

3. Exhaustively search the remaining candidates of $k_B \cup k_F$ and the unknown subkey bits of $k \backslash (k_B \cup k_F)$.

In step (b), we can choose to first build pairs on either the plaintext side or the ciphertext side before constructing quartets, the size of set S_i determines which choice is preferable.

The probability that a wrong key survives through step (c) is $P = (1 - 2^{-2(c_B^* + c_F^*)})^Q$, thus we have $Q = 2^{2c_B^* + 2c_F^* + LG(g)}$ by letting $P = 2^{-g}$.

Data Complexity. The data complexity can be divided into two cases.

- When multiple structures are used, the data complexity is

$$D = 4 \cdot D' = 2^{n + c_B + c_F - r_B - r_F + LG(g)/2 + 2}$$

The key guessing strategy does not affect the data complexity in this case.

- If less than one structure is used, let 2^{x_1} be the number of sub-structures, each contains 2^{x_2} data on average. Then $Q = 2^{2x_1+4x_2} \cdot 2^{2r_F^*-2n}$, which means that the data complexity is

$$D = 4 \cdot 2^{x_1+x_2} = 4 \cdot 2^{\frac{c_B+c_F+n-r_F+LG(g)/2+x_1-c'_B}{2}}. \tag{7}$$

Similarly, we can also launch the attack from the ciphertext side, the complexity analysis is similar and is omitted here.

Time Complexity. The IB attack also consists of three phases:

- *Quartets Generation.* This phase consists of steps 1, 2(a) and 2(b). The time complexity of this phase includes
  1. Cost of data generation: D
  2. Partial encryption/decryption: $T_0 = 2^{|k'_B \cup k'_F|} \cdot D$
  3. The cost of producing sets: $T_1 = 2^{|k'_B \cup k'_F|} \cdot \min\{D' \cdot 2^{r_B^*}, D'^2 \cdot 2^{r_F^*-n}\}$ memory accesses (MAs).
- *Guess-and-Filter.* This phase consists of step 2(c). The time complexity of this phase can be closely approximately by

$$\begin{aligned} T_2 + T_3 &= 2^{|k'_B \cup k'_F|}\left(Q + 2^{|k_B^* \cup k_F^*|} \cdot \frac{Q}{2^{2(c_B^*+c_F^*)}}\right) \\ &= 2^{|k'_B \cup k'_F|+2c_B^*+2c_F^*+LG(g)} + 2^{|k_B \cup k_F|+LG(g)} \end{aligned}$$

- *Exhaustive Search.* This phase consists of step 3. $T_4 = 2^{k-g}$.

In summary, the time complexity of the IB attack is

$$T = (D + (T_0 + T_2 + T_3)C_{E'} + T_4)C_E + T_1 \text{ MAs}.$$

Memory Complexity. The memory complexity of IB attacks is given by

$$M = \min\{2^{2c_B^*+2c_F^*+LG(g)}, 2^{|k_B \cup k_F|}\}.$$

Advantages. We analyze the advantages of the generic key recovery framework as follows.

- Our framework considers all the key guessing strategies. The two key recovery methods presented in [20, 21] are two special cases of our framework, that is when $k'_B = k'_F = 0$ or $k'_B = k_B, k'_F = 0$. As a demonstration, the attack on SKINNYee in Section 6 uses the same distinguisher in [20], while we improve the attack by one round using the generic key guessing strategy.
- *Time.* Note that T_2 directly corresponds to the number of quartets used in the guess-and-filter phase. Compared to the method 1 in Appendix B.1, we can reduce T_2 when $|k'_B \cup k'_F| < 2c'_B + 2c'_F$, and then reduce the overall time complexity potentially.

- *Data.* The data complexity can only be reduced when less one structure is used, which occurs if $x_1 - c'_B < 0$ in Equation (7).
- *Memory.* With the key guessing strategy, we have $c^*_B + c^*_F < c_B + c_F$, which means that the memory complexity can be effectively reduced when only the quartets need to be stored, as exemplified by the attacks on SKINNYee and Deoxys-BC in Sections 6, J.2 and J.3.

In Section 5, we will introduce an MILP model that can find the best attacking parameters for the impossible cryptanalysis.

4 New Insights in Impossible Cryptanalysis

4.1 Probabilistic Extensions

In a classical key recovery attack for impossible cryptanalysis, the input and output differences of the impossible distinguisher are propagated backward and forward through E_B^{-1} and E_F both with probability 1. However, cases where the extensions propagate probabilistically have not been systematically analyzed. As a reference, Song et al. recently investigated such probabilistic extensions in rectangle attacks and differential attacks [28]. To address this gap, we provide a complexity analysis in this section, comparing the probabilistic extensions to deterministic extensions in ID attacks.

The complexity analysis for deterministic extensions has been provided in Section 2.1. To differentiate the notations, we add a bar to each symbol for the case of probabilistic extensions, e.g., $\bar{c}_F$.

Allowing probabilistic extensions is likely to lead to more filtering power on the plaintext/ciphertext side, which could affect the parameters $\bar{r}_B$, $\bar{r}_F$, $\bar{k}_B$, $\bar{k}_F$, etc. Thus, the complexities could be changed according to the above comparison. The comparison is as follows.

- *Data complexity.* $D/\bar{D} = 2^{(c_B+c_F-r_B-r_F)-(\bar{c}_B+\bar{c}_F-\bar{r}_B-\bar{r}_F)}$. The data complexity is likely to increase as we add constraints to the extensions, causing more bit-conditions to be satisfied and a sparser plaintext/ciphertext difference pattern. When applying probabilistic extensions in E_B (resp. E_F), the value of $\bar{c}_B - \bar{r}_B$ (resp. $\bar{c}_F - \bar{r}_F$) would be larger than the value of $c_B - r_B$ (resp. $c_F - r_F$) when using deterministic extensions. Examples in figures 3 and 4 will provide a direct demonstration of the increase in data complexity.
- *Time complexity.*
  - $T_1/\bar{T}_1 = 2^{c_B+c_F-\bar{c}_B-\bar{c}_F}$. With probabilistic extensions, the values of $\bar{r}_B$ and $\bar{r}_F$ are reduced, leading to a corresponding reduction in the bit-conditions ($\bar{c}_B$ and $\bar{c}_F$) that need to be verified. However, on the other hand, the addition of extra constraints to the extension will increase the size of bit-conditions. Thus, the affect on T_1 should be analyzed within the context of specific attacks.
  - $T_2/\bar{T}_2 = 2^{|k_B \cup k_F| - |\bar{k}_B \cup \bar{k}_F|}$. The involved subkey bits will be reduced in most cases of probabilistic extensions, leading to a corresponding reduction in T_2.

- *Memory complexity.* $\bar{M} = \min\{2^{\bar{c}_B+\bar{c}_F+LG(g)}, 2^{|\bar{k}_B\cup\bar{k}_F|}\}$. As we can reduce the involved subkey bits and potentially decrease the number of pair candidates, the memory complexity is also very likely to be reduced.

For the case of IB attacks, the complexity analysis is similar, so we omit it.

In the following, we use AES as an example to demonstrate the usefulness of probabilistic extensions. For simplicity, we focus solely on the extension in E_F.

Examples for Comparison. Figure 3 shows an example of E_F where three rounds are appended to the distinguisher with deterministic extensions, the corresponding steps of guess-and-filter and the complexities are listed in Table 3.

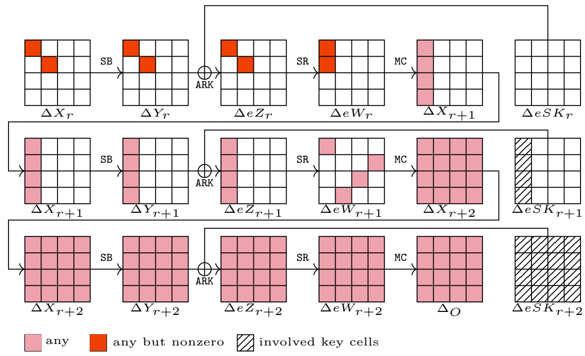


Fig. 3: Deterministic extensions. $eSK_r = \texttt{SR}^{-1} \circ \texttt{MC}^{-1}(SK_r)$. $r_F = 16c$, $c_F = 14c$, $|k_F| = 20c$, where $c = 8$ represents the cell size.

Table 3: Guess-and-Filter phase for Figure 3

Steps	Guess	Filter	Time
a)	$eSK_{r+2}[0-15]$	$\Delta eW_{r+1}[1-6, 8, 9, 11, 12, 14, 15] = 0$	$N_1 2^{4c} + N_1 2^{-3c+8c} + N_1 2^{-6c+12c} + N_1 2^{-9c+16c} \approx N_1 2^{7c}$
b)	$eSK_{r+1}[0-3]$	$\Delta eW_r[2, 3] = 0$	$N_1 2^{-12c+20c} = N_1 2^{8c}$
$N_1 = 2^{c_B+c_F+LG(g)} = 2^{c_B+14c+LG(g)}$ pairs satisfying the differences of plaintext and ciphertext.			
Data: $2^{n+1+LG(g)+c_B-r_B-2c}$.			
Time for this step: $2^{c_B+22c+LG(g)}$.			
Memory: $\min\{2^{c_B+14c+LG(g)}, 2^{|k_B|+20c}\}$			

When we apply probabilistic extensions, the updated trail is shown in Figure 4, where the MC transition in round r happens with probability 2^{-c}. The steps of guess-and filter and the complexities are listed in Table 4.

As a result, the example of probabilistic extensions has an increase of data complexity by 2^8, while its time complexity is reduced by 2^{39} and the memory complexity is reduced by 2^{24} at least. The effectiveness of probabilistic extensions can also be

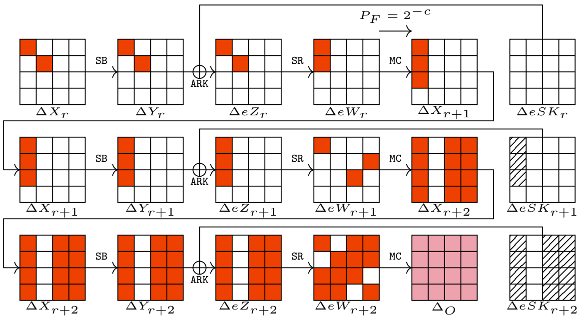


Fig. 4: Probabilistic extensions. The MC operation in round r happens with probability 2^{-c}. $\bar{r}_F = 12c$, $\bar{c}_F = 11c$, $|\bar{k}_F| = 15c$.

Table 4: Guess-and-Filter phase for Figure 4

Steps	Guess	Filter	Time
a)	$eSK_{r+2}[0-3, 8-15]$	$\Delta eW_{r+1}[1-3, 8, 9, 11, 12, 14, 15] = 0$	$N_2 2^{4c} + N_2 2^{-3c+8c} +$ $N_2 2^{-6c+12c} \approx N_2 2^{6c}$
b)	$eSK_{r+1}[0-2]$	$\Delta eW_r[2, 3] = 0$	$N_2 2^{-9c+15c} = N_2 2^{6c}$

$N_2 = 2^{c_B+11c+LG(g)}$ pairs satisfying the differences of plaintext and ciphertext.
Data: $2^{n+1+LG(g)+c_B-r_B-c}$.
Time for this step: $2^{c_B+17c+LG(g)+1}$.
Memory: $\min\{2^{c_B+11c+LG(g)}, 2^{|k_B|+15c}\}$

exemplified by the attacks on SKINNY-n-2n (see Appendix H.3) and SKINNY-n-3n (see Appendix H.4).

4.2 Complete Contradiction Detection in Impossible Boomerang Distinguishers

In this subsection, we first recall several boomerang tables and then introduce two new tables: iUBCT and iLBCT for identifying multiple-round contradiction.

The DBCT is defined by the combination of the UBCT and the LBCT, with the definitions of UBCT and LBCT provided in Appendix A. The definition of DBCT is as follows:

Definition 2 (DBCT [31, 32])**.** *Let S be a bijective function over $\mathbb{F}_2^n$. The Double Boomerang Connectivity Table (*DBCT*) of S is a two-dimensional table defined as*

$$\mathtt{DBCT}(\alpha_0, \beta_2) = \sum_{\alpha_1, \beta_1} \mathtt{UBCT}(\alpha_0, \alpha_1, \beta_1) \cdot \mathtt{LBCT}(\alpha_1, \beta_1, \beta_2).$$

In [35], Li *et al.* proposed the generalized boomerang connectivity table (GBCT) to characterize the asymmetric boomerang switch.

Definition 3 (GBCT [35])**.** *Let S be a bijective function over $\mathbb{F}_2^n$. The* GBCT *of S is a four-dimensional table defined as*

$$\texttt{GBCT}(\alpha_0, \alpha_0'; \beta_1, \beta_1') = \#\{x \in \mathbb{F}_2^n | S^{-1}(S(x) \oplus \beta_1) \oplus S^{-1}(S(x \oplus \alpha_0) \oplus \beta_1') = \alpha_0'\}.$$

Wang *et al.* [36] proposed a variant of DBCT, called DBCT*, aiming to calculate the two-round propagation probability more accurately by considering all possibilities of the middle differences.

Definition 4 (DBCT* [36])**.** *Let S be a bijective function over $\mathbb{F}_2^n$. The* DBCT* *of S is a two-dimensional table defined as*

$$\begin{aligned}\texttt{DBCT}^*(\alpha_0, \beta_2) = \sum_{\alpha_1, \alpha_1', \beta_1, \beta_1'} & \left(\# \left\{ x \in \mathbb{F}_2^n \,\middle|\, \begin{array}{l} S(x) \oplus S(x \oplus \alpha_0) = \alpha_1, \\ S(x) \oplus \beta_1 \oplus S(x \oplus \alpha_0) \oplus \beta_1' = \alpha_1', \\ S^{-1}(S(x) \oplus \beta_1) \oplus S^{-1}(S(x \oplus \alpha_0) \oplus \beta_1') = \alpha_0 \end{array} \right\}\right) \\ & \times \left(\# \left\{ x \in \mathbb{F}_2^n \,\middle|\, \begin{array}{l} x \oplus S^{-1}(S(x) \oplus \beta_2) = \beta_1, \\ S(x \oplus \alpha_1) \oplus S(x \oplus \alpha_1 \oplus \beta_1') = \beta_2, \\ S^{-1}(S(x) \oplus \beta_2) \oplus S^{-1}(S(x \oplus \alpha_1) \oplus \beta_2) = \alpha_1' \end{array} \right\}\right).\end{aligned}$$

Utilizing the BCT to enforce a contradiction in an IB distinguisher is straightforward and effective, as demonstrated by the attacks in [20, 21] . On the other hand, it is an intriguing challenge to exploit contradictions in multiple rounds. Zhang et al. attempted to use the DBCT to construct a 3-round IB distinguisher of SKINNY [21] based on zero entries of the DBCT. Nevertheless, this IB distinguisher was proven invalid, as the probability is not 0, according to [33]. The underlying reason [20, 33] is that the DBCT only covers quartets with equal differences on facing sides ($\alpha_1 = \alpha_1'$ and $\beta_1 = \beta_1'$), missing the admissible set of middle differences $(\alpha_1, \alpha_1', \beta_1, \alpha_1 \oplus \alpha_1' \oplus \beta_1)$ with $\alpha_1 \neq \alpha_1'$ that could potentially establish the connection. Therefore, it is suggested to use either the DBCT* [36] or the GDBCT [21] (both cover all the possibilities for the middle differences) to accurately analyze the switching probability in two rounds. Note that the DBCT* covers the cases where the input and output differences are the same on both sides, while the GDBCT covers the most generic case where even the input and output differences have no specific structure [33].

However, the computational complexity of the DBCT* is very high, which has been mentioned in [20, 36]. Specifically, the complexity would be $\mathcal{O}(2^{5n})$ for an n-bit Sbox. We conducted an experiment on an Intel Xeon Gold 5220R processor, and the results indicate that for an 8-bit Sbox, the computation takes approximately 4 hours and requires 33 GB of memory. This high resource demand could significantly impedes the integration of the DBCT* into an automated searching tool for IB attacks.

Realizing that the IB distinguisher only concerns the zero entries of the DBCT*, we propose two tables specifically designed for IB attacks, named **iUBCT** and **iLBCT**. We focus on the general configuration shown on the right in Figure 5, which considers the linear layer between S-boxes. The corresponding definitions are as follows.

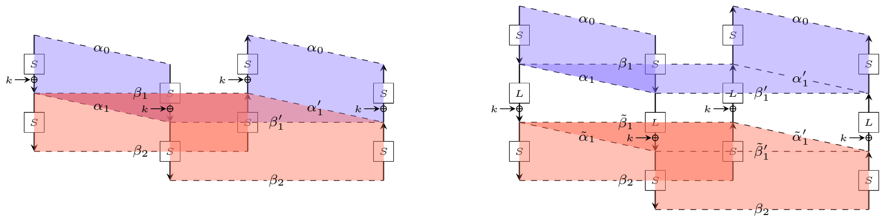


Fig. 5: The cases of two consecutive Sbox layers in a boomerang covering all admissible set of middle differences. The right one considers a linear layer in the middle, while it is omitted in the left one.

Definition 5 (`iUBCT`, `iLBCT`). *Let S be a bijective function over $\mathbb{F}_2^n$. The* `iUBCT` *and* `iLBCT` *of S are defined as (see Figure 5):*

$$\mathtt{iUBCT}(\alpha_0, \beta_2) = \#\left\{ (\alpha_1, \alpha_1') \in (\mathbb{F}_2^n)^2 \;\middle|\; \begin{array}{l} \mathtt{DDT}(\alpha_0, \alpha_1) > 0 \\ \mathtt{DDT}(\alpha_0, \alpha_1') > 0 \\ \mathtt{GBCT}(\alpha_1, \alpha_1'; \beta_2, \beta_2) > 0 \end{array} \right\},$$

$$\mathtt{iLBCT}(\alpha_0, \beta_2) = \#\left\{ (\beta_1, \beta_1') \in (\mathbb{F}_2^n)^2 \;\middle|\; \begin{array}{l} \mathtt{DDT}(\beta_1, \beta_2) > 0 \\ \mathtt{DDT}(\beta_1', \beta_2) > 0 \\ \mathtt{GBCT}(\alpha_0, \alpha_0; \beta_1, \beta_1') > 0 \end{array} \right\}.$$

Lemma 1. *For any nonzero α_0 and β_2, if either* $\mathtt{iUBCT}(\alpha_0, \beta_2) = 0$ *or* $\mathtt{iLBCT}(\alpha_0, \beta_2) = 0$*, or both entries are 0, then* $\mathtt{DBCT}^*(\alpha_0, \beta_2) = 0$.

Proof. Assume we have $\mathtt{iUBCT}(\alpha_0, \beta_2) = 0$. If $\mathtt{DBCT}^*(\alpha_0, \beta_2) > 0$, it means there exists at least a pair of values satisfy the boomerang transition of the two rounds. Thus, we can obtain the corresponding differences α_1 and α_1' for this pair, which contradicts the assumption. The proof is similar for `iLBCT`. □

From Definition 5 and the Lemma 1, it can be concluded that the `iUBCT` and the `iLBCT` are not as comprehensive as the `DBCT`* in capturing two-round contradictions, which means a non-zero entry in `iUBCT` and `iLBCT` could still be impossible. The advantage of the two new tables lies in helping encode two-round contradictions into the search model, enabling rapid identification of two-round zero-probability propagation. The distinguisher in figure 11 is obtained by our model in Section 5 based on `iLBCT`, while it is difficult to obtain this distinguisher using `DBCT`* directly. Note that we only concerns zero entries in the `iUBCT` and the `iLBCT`, the efficient procedures for identifying the zero entries and also the computational complexity are referred to Appendix D. Compared to using `DBCT`* or `GDBCT` to find contradictions through two consecutive rounds, we can significantly reduce the complexity from $\mathcal{O}(2^{5n})$ to $\mathcal{O}(2^{3n})$.

Analogous to the extension of `DBCT` to `3BCT` in [32], we can similarly define `i3UBCT`/`i3MBCT`/`i3LBCT` in Appendix F. Based on these newly proposed cryptanalytic tables and also the BCT, we can enforce

$$\texttt{BCT} = 0/\texttt{iUBCT} = 0 \vee \texttt{iLBCT} = 0/\texttt{i3UBCT} = 0 \vee \texttt{i3MBCT} = 0 \vee \texttt{i3LBCT} = 0$$

to construct contradictions for consecutive 1/2/3 rounds in IB distinguishers.

Determine the boundary of contradictions in IB distinguisher. In this work, we computed the iUBCT and iLBCT for all targeted ciphers. Based on iLBCT, we found a new 19-round related-tweakey impossible boomerang distinguisher for SKINNY-128-384 with two-round E_m for the first time (see Appendix E). The new distinguisher exceeds one more round than the 18-round one based on BCT (introduced in [21]), which illustrates the power of the new technique.

In addition, for the 4-bit Sbox of SKINNY-64 and the 8-bit Sbox of Deoxys-BC (same as AES), none of their iUBCT/iLBCT contains any entries with a value of 0. Consequently, for these ciphers, the maximum number of consecutive rounds that we can enforce a contradiction does not exceed one round.

5 Automated Tool for Searching the Full Impossible Cryptanalysis

In this section, we introduce an MILP-based tool for searching full ID and IB attacks. This tool supports: (1) searching for a full attack including the distinguisher and the key recovery; (2) searching for a standalone distinguisher; (3) optimizing attacks for a given distinguisher. (4) single-key and related-key settings[2]. The constraints on the components of the round function in this model are synthesized from the model proposed in [21, 27, 28, 37], and the inequalities for each part are listed in Appendix G. The source codes of this tool are publicly available at

https://github.com/ImpossibleCryptanalysis-2024/Tool

Suppose that the round function of the target cipher $E = E_F \circ E_D \circ E_B = E_F \circ E_1 \circ E_0 \circ E_B$ consists of

$$W_{r-1} \xrightarrow{\oplus K_r} X_r \xrightarrow{\texttt{SB}} Y_r \xrightarrow{\texttt{SR}} Z_r \xrightarrow{\texttt{MC}} W_r.$$

The upper trail includes E_0 and E_B, while the lower trail includes E_1 and E_F. We use $X^{up}_{r,i}$ to denote the i-th cell in the internal state before the SB operation of the r-th round in the upper trail, and $X^{lo}_{r,i}$ to denote the i-th cell in the internal state before the SB operation of the r-th round in the lower trail. Similarly, $Y^{up}_{r,i}$, $Y^{lo}_{r,i}$, $K^{up}_{r,i}$, $K^{lo}_{r,i}$, etc. represent the various internal states and key cells.

[2]In this work, we have searched for distinguishers and attacks separately on all targeted ciphers, ensuring that the distinguishers in all attacks are those with the longest rounds found. The distinguishers used in the attacks in Section 6 and Appendices H.3, H.4 and I.2 are the same as those proposed in previous works, which can be considered as the distinguishers that yield the optimal attacks. The distinguishers used in the optimal attacks described in Appendices H.2, J.2 and J.3 are reported for the first time.

5.1 Modeling the Difference Propagation in Distinguishers

With the miss-in-the-middle technique, we aim to search for one characteristic with probability 1 over E_0 and one characteristic with probability 1 over E_1^{-1} to construct the impossible distinguisher. We define four types of difference status for the state cells and key cells: 1) zero difference, 2) fixed nonzero difference, 3) any but nonzero difference, and 4) any difference, using two binary variables s_0, s_1 to characterize the different types. Let x^{s_0} and x^{s_1} denote the attributes s_0 and s_1 of a cell x, respectively. The corresponding relationship between the values and the difference states is as follows:

- $(x^{s_0}, x^{s_1}) = (1,1)$ zero difference
- $(x^{s_0}, x^{s_1}) = (0,1)$fixed nonzero difference
- $(x^{s_0}, x^{s_1}) = (1,0)$ any
- $(x^{s_0}, x^{s_1}) = (0,0)$ any but nonzero

`SubBytes`. In `SubBytes` operation, the effects of the Sbox on the deterministic propagation of each cell are as follows:

$$(1,1) \xrightarrow{S} (1,1) \qquad (0,1) \xrightarrow{S} (0,0) \qquad (1,0) \xrightarrow{S} (1,0) \qquad (0,0) \xrightarrow{S} (0,0)$$

The constraints for `SubBytes` are listed in Appendix G.1.

`MixColumns`. The `XOR` operation is a commonly used component in `MixColumns`, and the deterministic propagation of differences in the `XOR` operation can be categorized as [3]:

$$(a,b) \oplus (1,1) = (a,b) \qquad (a,b) \oplus (1,0) = (1,0) \qquad (0,1) \oplus (0,0) = \begin{matrix}(0,1)\\(1,1)\end{matrix} \qquad \begin{matrix}(0,1)\\(0,0)\end{matrix} \oplus (0,0) = (1,0)$$

The constraints for `MixColumns` are listed in Appendix G.2.

Enforcing Contradictions. To enforce contradictions in an ID distinguisher, we can set

$$\bigvee_{i=0}^{15} \begin{pmatrix} (X_{r_m,i}^{up,s_0} = 0 \wedge X_{r_m,i}^{up,s_1} = 1 \wedge Y_{r_m,i}^{lo,s_0} = 0 \wedge Y_{r_m,i}^{lo,s_1} = 1) \vee \\ (X_{r_m,i}^{up,s_0} = 0 \wedge X_{r_m,i}^{up,s_1} = 0 \wedge Y_{r_m,i}^{lo,s_0} = 1 \wedge Y_{r_m,i}^{lo,s_1} = 1) \vee \\ (X_{r_m,i}^{up,s_0} = 1 \wedge X_{r_m,i}^{up,s_1} = 1 \wedge Y_{r_m,i}^{lo,s_0} = 0 \wedge Y_{r_m,i}^{lo,s_1} = 0) \end{pmatrix} = 1,$$

where r_m is set before running the model.

Let L^E (resp. L^D) be a binary 16×16 matrix to describe the linear layer (combination of SR and MC (resp. $\mathtt{SR}^{-1}$ and $\mathtt{MC}^{-1}$)), $L^E(i)$ (resp. $L^D(i)$) be the set of the indexes j such that the coefficient $L^E_{i,j} = 1$ (resp. $L^D_{i,j} = 1$) in the matrix L^E (resp. L^D). We provide the matrices L^E and L^D for SKINNY and AES in Appendix G.6. Additionally, we denote ${}^tL^E$ (resp. ${}^tL^D$) as the transpose of L^E (resp. L^D).

For enforcing contradictions in an IB distinguisher (as introduced in Section 4.2), we can set

$$\mathtt{contr}_1 = \bigvee_{i=0}^{15} (X_{r_m,i}^{up,s_0} = 0 \wedge X_{r_m,i}^{up,s_1} = 1 \wedge Y_{r_m,i}^{lo,s_0} = 0 \wedge Y_{r_m,i}^{lo,s_1}) = 1$$

[3] (a,b) refers to any value including $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$.

for describing contradiction through single-round E_m ($\texttt{BCT} = 0$),

$$\texttt{contr}_{\texttt{2U}} = \bigvee_{i=0}^{15} \left(\begin{array}{l} X^{up,s_0}_{r_m,i} = 0 \wedge X^{up,s_1}_{r_m,i} = 1 \wedge \\ \left(\bigvee_{j \in L^E(i)} Y^{lo,s_0}_{r_m+1,j} = 0 \wedge Y^{lo,s_1}_{r_m+1,j} = 1 \wedge (\bigwedge_{k \in {}^tL^E(j) \setminus i} X^{up,s_0}_{r_m,k} = 1 \wedge X^{up,s_1}_{r_m,k} = 1) \right) \end{array} \right)$$

and

$$\texttt{contr}_{\texttt{2L}} = \bigvee_{i=0}^{15} \left(\begin{array}{l} Y^{lo,s_0}_{r_m+1,i} = 0 \wedge Y^{lo,s_1}_{r_m+1,i} = 1 \wedge \\ \left(\bigvee_{j \in L^D(i)} X^{up,s_0}_{r_m,j} = 0 \wedge X^{up,s_1}_{r_m,j} = 1 \wedge (\bigwedge_{k \in {}^tL^D(j) \setminus i} Y^{lo,s_0}_{r_m+1,k} = 1 \wedge Y^{lo,s_1}_{r_m+1,k} = 1) \right) \end{array} \right)$$

for describing contradictions through two-round E_m ($\texttt{contr}_{\texttt{2U}}$ for $\texttt{iUBCT} = 0$, $\texttt{contr}_{\texttt{2L}}$ for $\texttt{iLBCT} = 0$), then set $\texttt{contr}_2 = \texttt{contr}_{\texttt{2U}} \wedge \texttt{contr}_{\texttt{2L}}$. Figure 6 provides a toy example with AES-like linear layer of $\texttt{contr}_{\texttt{2U}}$ when $i = 0, j \in \{0, 1, 2, 3\}, k \in \{5, 10, 15\}$, and $\texttt{contr}_{\texttt{2L}}$ when $i = 0, j \in \{0, 5, 10, 15\}, k \in \{1, 2, 3\}$. Similarly, we can set constraints

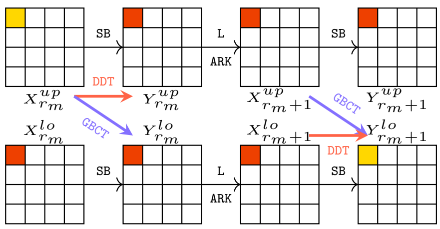


Fig. 6: A toy example of the contradiction $\texttt{contr}_2$ for two-round E_m

for describing contradiction through three-round E_m ($\texttt{i3UBCT}/\texttt{i3MBCT}/\texttt{i3LBCT} = 0$). Finally, we impose

$$\bigvee_{i=1}^{3} \texttt{contr}_{\texttt{i}} = 1$$

to specify that at least one contradiction must exist in the impossible boomerang distinguisher.

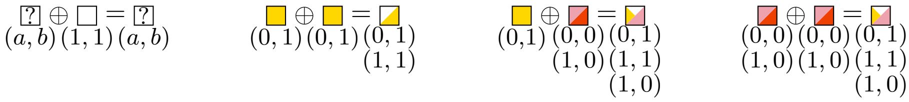
5.2 Modeling the Difference Propagation in Key Recovery

In this model, we introduce probabilistic extensions into the search of key recovery phase for E_B and E_F.

`SubBytes`. In `SubBytes` operation, the effects of the Sbox on the probabilistic propagation of each cell are as follows.

The constraints for `SubBytes` are listed in Appendix G.3.

`MixColumns`. The `XOR` operation is a commonly used component in `MixColumns`, and the probabilistic propagation of differences in the `XOR` operation can be categorized as:



The constraints for `MixColumns` are listed in Appendix G.4.

Identifying cell conditions. The cell conditions that need to be verified in E_B will only exist in states Y and W, while the cell conditions that need to be verified in E_F will only exist in states X and Z. Taking E_B as an example, the conditions in Y can be identified by

$$Y_{r,i}^{up,c} = \neg Y_{r,i}^{up,s0} \wedge Y_{r,i}^{up,s1},$$

and the conditions in W can be identified by

$$W_{r,i}^{up,c} = W_{r,i}^{up,s1} \wedge \Big(\bigvee_{j \in {}^tL^D(i)} \neg X_{r,j}^{up,s1} \Big),$$

thus we can obtain $c_B = \sum_r \sum_i (Y_{r,i}^{up,c} + W_{r,i}^{up,c})$ for the complexity calculations. The constraints in E_F are similar.

Identifying involved key cells and pre-guessed key cells. For the key cell $K_{r,i}$, we set a binary variable $K_{r,i}^{g} = 1$ indicates that this key cell is involved in E_B or E_F, and $K_{r,i}^{g} = 0$ indicates it is involved in E_B or E_F. We also set a binary variable $K_{r,i}^{p} = 1$ indicates that this key cell would be pre-guessed to construct pairs (in ID attacks) or quartets (in IB attacks), while $K_{r,i}^{p} = 0$ indicates that this key cell is not used for data collection phase. Thus, there is a clear constraint

$$\texttt{if } K_{r,i}^{p} = 1, \texttt{ then } K_{r,i}^{g} = 1$$

to describe the relationship between the two variables.

Continuing with E_B as an example, the subkey cells involved in verifying the conditions of Y_r can be calculated by

$$\texttt{if } Y_{r,i}^{up,c} = 1, \texttt{ then } K_{r,i}^{up,g} = 1,$$

and the subkey cells involved in verifying the conditions of W_r can be calculated by

$$\texttt{if } \big((W_{r,i}^{up,c} = 1) \wedge (X_{r,j}^{up,s0} + X_{r,j}^{up,s1} \le 1) \big) = 1, \texttt{ then } K_{r,j}^{up,g} = 1 \texttt{ for } \forall j \in {}^tL^D(i).$$

In addition, we can backtrack to calculate all the involved subkey cells from round 0 using

$$\texttt{if } K_{r,i}^{up,g} = 1, \texttt{ then } K_{r-1,j}^{up,g} = 1 \texttt{ for } \forall j \in {}^tL^D(i).$$

Thus, we can obtain $k_B = \sum_r \sum_i K_{r,i}^{up,g}$.

To calculate the subkey cells used for pairs (quartets) generation phase, we introduce two variables Y_r^v and W_r^v to indicate that the conditions are verified under the pre-guessed keys K_r^p. The constraints among Y_r^v, W_r^v, and K_r^p are similar to those constraints among Y_r^c, W_r^c, and K_r^g mentioned above, with the specific constraints listed in Appendix G.5. The constraints in E_F are similar and thus we can obtain r'_B, r'_F, c'_B, c'_F, k'_B, and k'_F.

Furthermore, we can add a characterization of the key-bridging technique to calculate $|k_B \cup k_F|$ and $|k'_B \cup k'_F|$. The detailed description of the key-bridging is referred to [12, 27, 38].

Objective function. For the complexity calculation formulas introduced in Section 3, all parameters r'_B, r'_F, c'_B, c'_F, k'_B, and k'_F can be represented. Therefore, we set the objective function of the model as min T to calculate the attack with the lowest time complexity. Since all parameters are known, we can also define the objective function with the data complexity or the memory complexity.

Instantiation and verification. In this model, the outputs of distinguishers and attacks are truncated ones, i.e., the active status of each cell is known, while the difference values of cells with fixed difference are unknown. This requires instantiation with the specification of the targeted cipher. It is particularly important to consider the properties of the Sbox when enforcing contradictions to find instantiations. The optimization and verification procedure of this model is illustrated in Algorithm 3.

6 Applications

To demonstrate the advancements in impossible cryptanalysis achieved this work, we apply our methods to SKINNY, SKINNYee, Midori, and Deoxys-BC in this paper. Due to the page limit, we only present the attack on SKINNYee in this section, the remaining attacks are referred to Appendices H.2, H.3, H.4, I.2, J.2 and J.3.

6.1 Specification of SKINNYee

SKINNYee is a tweakable block cipher derived from the SKINNY family [39], proposed by Naito *et al.* at CRYPTO 2022 [40]. SKINNYee conforms to the new security scheme HOMA proposed in [40], featuring a 64-bit block size, a 128-bit key size, a 259-bit tweak size, and a total of 56 rounds. The design of SKINNYee is based on SKINNYe (TK4 setting) [41]. The round function is quite similar to that of the SKINNY family, but there are some differences in the operation `AddRoundKey` and the round constant generation. Figure 7 provides the round transformation of SKINNYee, and for more detailed specification please refer to the original design document.

The $LFSR_2$, $LFSR_3$, $LFSR_4$ and the cell permutation P_T are defined as:

$$\begin{aligned}
&LFSR_2 : (x_3 \| x_2 \| x_1 \| x_0) \to (x_2 \| x_1 \| x_0 \| x_3 \oplus x_2) \\
&LFSR_3 : (x_3 \| x_2 \| x_1 \| x_0) \to (x_0 \oplus x_3 \| x_3 \| x_2 \| x_1) \\
&LFSR_4 : (x_3 \| x_2 \| x_1 \| x_0) \to (x_1 \| x_0 \| x_3 \oplus x_2 \| x_2 \oplus x_1) \\
&P_T : (0, ..., 15) \to (9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7)
\end{aligned}$$

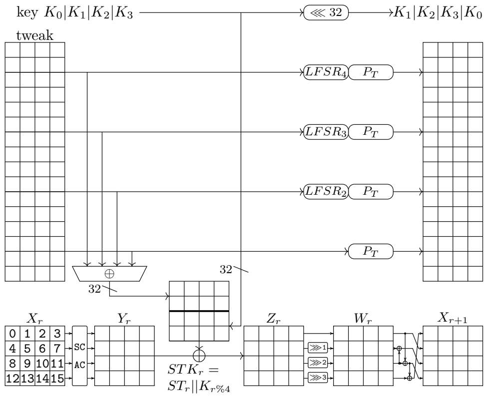


Fig. 7: Round Transformation of SKINNYee

6.2 30-Round Impossible Boomerang Attack on SKINNYee

We propose a 30-round related-tweak IB attack against SKINNYee, which is obtained by our automated tool. In this attack, we use a 21-round related-tweak IB distinguisher, which is the same distinguisher used in [20]:

$$(\Delta Y_5, \Delta ST_5) = ((0000|0a00|0000|0000), (0000|0a00))$$
$$\not\rightarrow$$
$$(\nabla Z_{26}, \nabla ST_{26}) = ((0000|000b|0000|0000), (0000|000b)),$$

where Δ represents the difference in the upper characteristic and ∇ represents the difference in the lower characteristic of the boomerang trail. We prefix 6 rounds at the beginning and append 3 rounds at the end of the distinguisher to mount the attack, as shown in Figure 8. From the figure, we can get the parameters used for this attack: $r_B = 12c$, $c_B = 12c$, $r_F = 9c$, $c_F = 9c$, $|k_B \cup k_F| = 26c$, $c'_B = 5c$, $c'_F = 9c$, $|k'_B \cup k'_F| = 15c$. In the key-recovery extensions of this attack, it adopts deterministic extensions.

Quartets Collection. In this phase, we pre-guess 2^{15c} possible values of $K_0[0-7]$, $K_1[0,3,4,5,6,7]$ and $K_3[0]$ to generate two sets as:

$$S_1 = \{(P_1^*, C_1^*, P_3^*, C_3^*) \mid C_1^* \text{ and } C_3^* \text{ are colliding in } n - r_F^* = 16c \text{ bits}\},$$
$$S_2 = \{(P_2^*, C_2^*, P_4^*, C_4^*) \mid C_2^* \text{ and } C_4^* \text{ are colliding in } n - r_F^* = 16c \text{ bits}\}.$$

According to the framework introduced in Section 3.3, the size of each set is $D'^2 \cdot 2^{r_F^*-n} = 2^{n+c_F^*+LG(g)} = 2^{16c+LG(g)}$. From the two sets, we can construct $Q = 2^{2c_B*+2c_F^*+LG(g)} = 2^{14c+LG(g)}$ quartets for performing guess-and-filter phase.

Guess-and-Filter. For Q quartets under each pre-guessed subkey bits:

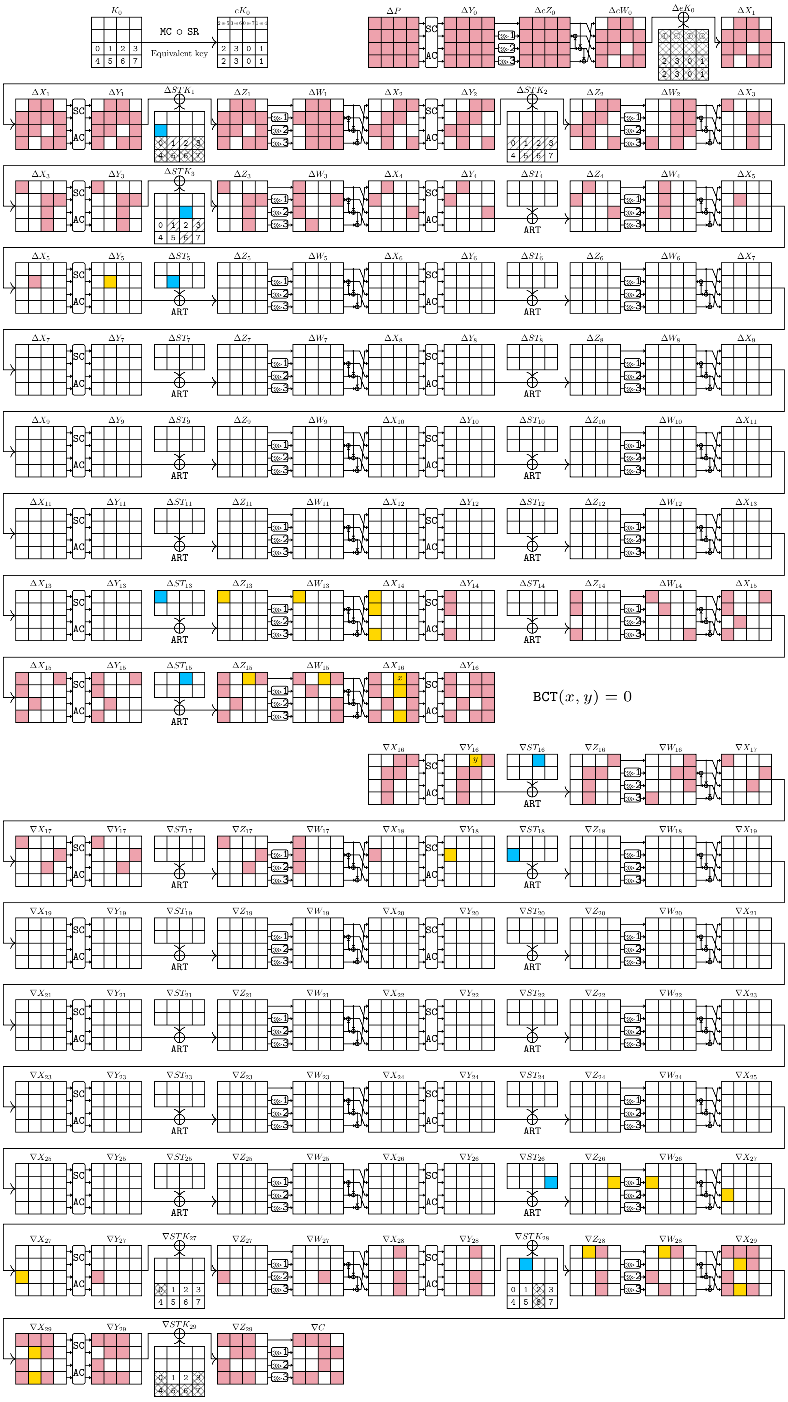


Fig. 8: The related-tweak impossible boomerang attack against 30-round SKINNYee

1. *Satisfying the cell conditions in* ΔX_3. Guess $K_1[1]$ and we can compute $\Delta W_2[3]$. The condition $\Delta W_2[3] = \Delta W_2[11]$ will lead to two c-bit filters on both sides of the boomerang. The time complexity of this step is $2^{16c} \cdot Q \cdot 4 \cdot \frac{1}{30 \cdot 16} = Q2^{16c-6.91}$.
2. *Satisfying the cell conditions in* ΔX_3. Guess $K_1[2]$ and we can compute $\Delta W_2[10]$. The condition $\Delta W_2[2] \oplus \Delta W_2[10] \oplus \Delta W_2[14] = 0$ will lead to two c-bit filters on both sides of the boomerang. The time complexity of this step is $2^{17c} \cdot Q \cdot 2^{-2c} \cdot 4 \cdot \frac{1}{30 \cdot 16} = Q2^{15c-6.91}$.
3. *Satisfying the cell conditions in* ΔX_4. Guess $K_2[0]$ and compute $\Delta W_3[8]$. The condition $\Delta W_3[4] = \Delta W_3[8]$ will lead to two c-bit filters. Guess $K_2[2,5]$ and compute $\Delta W_3[0]$. The condition $\Delta W_3[0] = \Delta W_3[8]$ will also lead to two c-bit filters. The time complexity of this step is $2^{18c} \cdot Q \cdot 2^{-4c} \cdot 4 \cdot \frac{1}{30 \cdot 16} + 2^{20c} \cdot Q \cdot 2^{-6c} \cdot 4 \cdot \frac{2}{30 \cdot 16} = Q2^{14c-5.32}$.
4. *Satisfying the cell conditions in* ΔX_5. Guess $K_2[3]$ and $K_3[1]$. The condition $\Delta W_4[5] = \Delta W_4[9]$ will lead to two c-bit filters. The time complexity of this step is $2^{22c} \cdot Q \cdot 2^{-8c} \cdot 4 \cdot \frac{2}{30 \cdot 16} = Q2^{14c-5.91}$.
5. *Satisfying the cell conditions in* ΔX_5 *and* ΔY_5. Guess $K_2[1,6]$ and $K_3[3,6]$. The conditions with $\Delta W_4[1] = \Delta W_4[9]$ and the known difference value of $Y_5[5]$ will lead to four c-bit filters. The time complexity of this step is $2^{26c} \cdot Q \cdot 2^{-10c} \cdot 4 \cdot \frac{4}{30 \cdot 16} = Q2^{16c-4.91}$.

Complexity. The data complexity is $D = 2^{16c + \frac{LG(g)}{2} + 2}$. The time complexity primarily consists of partial encryption/decryption, generating sets in quartets collection, guess-and-filter and exhaustive search, represented as $2^{31c + \frac{LG(g)}{2} + 2} \cdot \frac{15}{30 \cdot 16} + 2^{31c + LG(g) + 1} \cdot \frac{15}{30 \cdot 16} + 2^{30c + LG(g) - 5.12} + 2^{32c - g}$. We set $g = 6$, then $D = 2^{67.03}$, $T = 2^{123.61}$, $M = 2^{58.05}$.

7 Conclusion

We proposed a generic key recovery frameworks impossible cryptanalysis, supporting any form of key-guessing strategy. To further enhance the key recovery attacks, we provided the first formal analysis of probabilistic extensions in impossible cryptanalysis. Additionally, we also introduced a new approach for identifying multi-round contradictions in IB distinguishers. Finally, we integrated all the techniques into a new MILP-based model. This model enables us to discover the optimal impossible cryptanalysis for block ciphers: SKINNY, SKINNYee, Midori, and Deoxys-BC. Our results show significant improvements in both time and memory complexity compared to previous results, demonstrating the effectiveness of the new techniques.

Currently, our framework only targets the SPN ciphers, we believe it is potential to apply it to other types of ciphers.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (Grant No. 62302293).

Declarations

The authors declare that they have no competing interests.

References

[1] Knudsen, L.: DEAL-a 128-bit block cipher. complexity **258**(2), 216 (1998)

[2] Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) Advances in Cryptology – EUROCRYPT'99. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer, Prague, Czech Republic (1999). https://doi.org/10.1007/3-540-48910-X_2

[3] Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) Advances in Cryptology – CRYPTO'90. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer, Santa Barbara, CA, USA (1991). https://doi.org/10.1007/3-540-38424-3_1

[4] Biham, E., Biryukov, A., Shamir, A.: Miss in the middle attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, vol. 1636, pp. 124–138. Springer, Rome, Italy (1999). https://doi.org/10.1007/3-540-48519-8_10

[5] Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. Cryptology ePrint Archive, Report 2016/689. https://eprint.iacr.org/2016/689 (2016)

[6] Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J.-S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part III. Lecture Notes in Computer Science, vol. 10212, pp. 185–215. Springer, Paris, France (2017). https://doi.org/10.1007/978-3-319-56617-7_7

[7] Sun, L., Gerault, D., Wang, W., Wang, M.: On the usage of deterministic (RK) TDs and MDLAs for SPN ciphers. IACR Transactions on Symmetric Cryptology **2020**(3), 262–287 (2020) https://doi.org/10.13154/tosc.v2020.i3.262-287

[8] Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In: Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5365, pp. 279–293. Springer, ??? (2008). https://doi.org/10.1007/978-3-540-89754-5_22 . https://doi.org/10.1007/978-3-540-89754-5_22

[9] Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 179–199. Springer, Kaoshiung, Taiwan, R.O.C. (2014). https://doi.org/10.1007/978-3-662-45611-8_10

[10] Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. Journal of Cryptology **31**(1), 101–133 (2018) https://doi.org/10.1007/s00145-016-9251-7

[11] Derbez, P., Fouque, P.-A.: Automatic search of meet-in-the-middle and impossible differential attacks. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 157–184. Springer, Santa Barbara, CA, USA (2016). https://doi.org/10.1007/978-3-662-53008-5_6

[12] 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, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 128–157. Springer, Lyon, France (2023). https://doi.org/10.1007/978-3-031-30634-1_5

[13] 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 Transactions on Symmetric Cryptology **2024**(1), 234–325 (2024) https://doi.org/10.46586/tosc.v2024.i1.234-325

[14] Phan, R.C.: Impossible differential cryptanalysis of 7-round advanced encryption standard (AES). Inf. Process. Lett. **91**(1), 33–38 (2004) https://doi.org/10.1016/J.IPL.2004.02.018

[15] Bahrak, B., Aref, M.R.: A novel impossible differential cryptanalysis of AES. In: Proceedings of the Western European Workshop on Research in Cryptology, vol. 2007 (2007). Citeseer

[16] Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6498, pp. 282–291. Springer, ??? (2010). https://doi.org/10.1007/978-3-642-17401-8_20 . https://doi.org/10.1007/978-3-642-17401-8_20

[17] Song, L., Fu, Q., Yang, Q., Lv, Y., Hu, L.: Generalized Impossible Differential Attacks on Block Ciphers: Application to SKINNY and ForkSKINNY. Cryptology ePrint Archive, Paper 2024/1908 (2024). https://eprint.iacr.org/2024/1908

[18] Lu, J.: Cryptanalysis of block ciphers. PhD thesis, University of London UK (2008)

[19] Lu, J.: The (related-key) impossible boomerang attack and its application to the AES block cipher. Des. Codes Cryptogr. **60**(2), 123–143 (2011) https://doi.org/10.1007/s10623-010-9421-9

[20] Bonnetain, X., Cordero, M., Lallemand, V., Minier, M., Naya-Plasencia, M.: On impossible boomerang attacks application to Simon and SKINNYee. IACR Trans. Symmetric Cryptol. **2024**(2), 222–253 (2024) https://doi.org/10.46586/TOSC.V2024.I2.222-253

[21] Zhang, J., Wang, H., Tang, D.: Impossible boomerang attacks revisited applications to Deoxys-BC, Joltik-BC and SKINNY. IACR Trans. Symmetric Cryptol. **2024**(2), 254–295 (2024) https://doi.org/10.46586/TOSC.V2024.I2.254-295

[22] Hu, X., Feng, D., Jiao, L., Hao, Y., Gong, X., Li, Y.: A deep study of the impossible boomerang distinguishers: New construction theory and automatic search methods. IACR Cryptol. ePrint Arch., 1008 (2024)

[23] Hu, X., Jiao, L.: Pre-sieve, Partial-guess, and Accurate estimation: Full-round Related-key Impossible Boomerang Attack on ARADI. Cryptology ePrint Archive, Paper 2025/056 (2025). https://eprint.iacr.org/2025/056

[24] Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings (long paper). IACR Transactions on Symmetric Cryptology **2017**(3), 37–72 (2017) https://doi.org/10.13154/tosc.v2017.i3.37-72

[25] Liu, Y., Xiang, Z., Chen, S., Zhang, S., Zeng, X.: A novel automatic technique based on MILP to search for impossible differentials. In: Tibouchi, M., Wang, X. (eds.) ACNS 23: 21st International Conference on Applied Cryptography and Network Security, Part I. Lecture Notes in Computer Science, vol. 13905, pp. 119–148. Springer, Kyoto, Japan (2023). https://doi.org/10.1007/978-3-031-33488-7_5

[26] Lin, L., Wu, W.: Meet-in-the-middle attacks on reduced-round Midori64. IACR Transactions on Symmetric Cryptology **2017**(1), 215–239 (2017) https://doi.org/10.13154/tosc.v2017.i1.215-239

[27] 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, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 410–440. Springer, Taipei, Taiwan (2022). https://doi.org/10.1007/978-3-031-22963-3_14

[28] 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, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 339–367. Springer, Zurich, Switzerland (2024). https://doi.org/10.1007/978-3-031-58716-0_12

[29] Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In: Malkin, T. (ed.) Topics in Cryptology – CT-RSA 2008. Lecture Notes in Computer Science, vol.

4964, pp. 370–386. Springer, San Francisco, CA, USA (2008). https://doi.org/10.1007/978-3-540-79263-5_24

[30] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer, Tel Aviv, Israel (2018). https://doi.org/10.1007/978-3-319-78375-8_22

[31] Hadipour, H., Bagheri, N., Song, L.: Improved rectangle attacks on SKINNY and CRAFT. IACR Transactions on Symmetric Cryptology **2021**(2), 140–198 (2021) https://doi.org/10.46586/tosc.v2021.i2.140-198

[32] Yang, Q., Song, L., Sun, S., Shi, D., Hu, L.: New properties of the double boomerang connectivity table. IACR Transactions on Symmetric Cryptology **2022**(4), 208–242 (2022) https://doi.org/10.46586/tosc.v2022.i4.208-242

[33] Bonnetain, X., Lallemand, V.: A note on the use of the double boomerang connectivity table (DBCT) for spotting impossibilities. IACR Cryptol. ePrint Arch., 1218 (2024)

[34] Yang, Q., Song, L., Zhang, N., Shi, D., Wang, L., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. J. Cryptol. **37**(2), 19 (2024) https://doi.org/10.1007/S00145-024-09499-1

[35] Li, C., Wu, B., Lin, D.: Generalized boomerang connectivity table and improved cryptanalysis of GIFT. In: Inscrypt 2022, Beijing, China, December 11-13, 2022. Lecture Notes in Computer Science, vol. 13837, pp. 213–233. Springer, ??? (2022). https://doi.org/10.1007/978-3-031-26553-2_11 . https://doi.org/10.1007/978-3-031-26553-2_11

[36] Wang, L., Song, L., Wu, B., Rahman, M., Isobe, T.: Revisiting the boomerang attack from a perspective of 3-differential. IEEE Trans. Inf. Theory **70**(7), 5343–5357 (2024) https://doi.org/10.1109/TIT.2023.3324738

[37] Derbez, P., Euler, M., Fouque, P.-A., Nguyen, P.H.: Revisiting related-key boomerang attacks on AES using computer-aided tool. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology – ASIACRYPT 2022, Part III. Lecture Notes in Computer Science, vol. 13793, pp. 68–88. Springer, Taipei, Taiwan (2022). https://doi.org/10.1007/978-3-031-22969-5_3

[38] Dunkelman, O., Keller, N., Shamir, A.: Improved single-key attacks on 8-round AES-192 and AES-256. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 158–176. Springer, Singapore (2010). https://doi.org/10.1007/978-3-642-17373-8_10

[39] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer, Santa Barbara, CA, USA (2016). https://doi.org/10.1007/978-3-662-53008-5_5

[40] Naito, Y., Sasaki, Y., Sugawara, T.: Secret can be public: Low-memory AEAD mode for high-order masking. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part III. Lecture Notes in Computer Science, vol. 13509, pp. 315–345. Springer, Santa Barbara, CA, USA (2022). https://doi.org/10.1007/978-3-031-15982-4_11

[41] Naito, Y., Sasaki, Y., Sugawara, T.: Lightweight authenticated encryption mode suitable for threshold implementation. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 705–735. Springer, Zagreb, Croatia (2020). https://doi.org/10.1007/978-3-030-45724-2_24

[42] Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer, Rome, Italy (1999). https://doi.org/10.1007/3-540-48519-8_12

[43] Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier, B. (ed.) Fast Software Encryption – FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 75–93. Springer, New York, NY, USA (2001). https://doi.org/10.1007/3-540-44706-7_6

[44] Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 340–357. Springer, Innsbruck, Austria (2001). https://doi.org/10.1007/3-540-44987-6_21

[45] Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 393–410. Springer, Santa Barbara, CA, USA (2010). https://doi.org/10.1007/978-3-642-14623-7_21

[46] Wang, H., Peyrin, T.: Boomerang switch in multiple rounds. IACR Transactions on Symmetric Cryptology **2019**(1), 142–169 (2019) https://doi.org/10.13154/tosc.v2019.i1.142-169

[47] Delaune, S., Derbez, P., Vavrille, M.: Catching the fastest boomerangs application to SKINNY. IACR Transactions on Symmetric Cryptology **2020**(4), 104–129 (2020) https://doi.org/10.46586/tosc.v2020.i4.104-129

[48] Boukerrou, H., Huynh, P., Lallemand, V., Mandal, B., Minier, M.: On the Feistel counterpart of the boomerang connectivity table (long paper). IACR Transactions on Symmetric Cryptology **2020**(1), 331–362 (2020) https://doi.org/10.13154/tosc.v2020.i1.331-362

[49] Dunkelman, O.: Efficient Construction of the Boomerang Connection Table. Cryptology ePrint Archive, Paper 2018/631 (2018). https://eprint.iacr.org/2018/631

[50] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 411–436. Springer, Auckland, New Zealand (2015). https://doi.org/10.1007/978-3-662-48800-3_17

[51] Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41. Submitted to CAESAR **124** (2016)

[52] 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, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer, Kaoshiung, Taiwan, R.O.C. (2014). https://doi.org/10.1007/978-3-662-45608-8_15

A Overview of Boomerang Attacks and Boomerang Tables

The Boomerang attack is another major variant of differential cryptanalysis, first introduced by Wagner [42]. From the Boomerang attack, some similar attack techniques such as the amplified boomerang attack [43], the rectangle attack [44] and the sandwich attack [45] have evolved. The main idea of the boomerang attack and its variants is to concatenate an r_0-round differential characteristic and an r_1-round differential characteristic to obtain an $(r_0 + r_1)$-round boomerang trail. The compatibility of two short differential characteristics is an important topic of research in the study of boomerang attacks. Here, we introduce the dependency of two differential characteristics and some boomerang tables based on the framework of the sandwich attack.

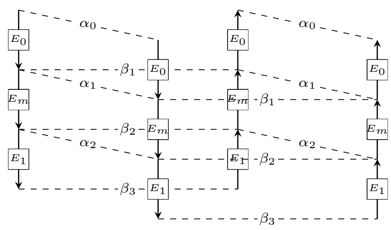


Fig. 9: Framework of sandwich attacks

Suppose that a block cipher E is treated as the composition of three sub-ciphers: $E = E_1 \circ E_m \circ E_0$, where there exists a short differential $\alpha_0 \to \alpha_1$ with probability p for E_0 and a short differential $\beta_2 \to \beta_3$ with probability q for E_1. Then the probability of an boomerang distinguisher is defined as:

$$\mathbb{P}\left(E^{-1}\left(E(P) \oplus \beta_3\right) \oplus E^{-1}\left(E\left(P \oplus \alpha_0\right) \oplus \beta_3\right) = \alpha_0\right) = \mathbb{P}_{E_0} \cdot \mathbb{P}_{E_m} \cdot \mathbb{P}_{E_1} = p^2 \cdot r \cdot q^2,$$

in which r denotes the probability of a boomerang returns over E_m for random input x:

$$r = \mathbb{P}\left(E_m^{-1}\left(E_m(x) \oplus \beta_2\right) \oplus E_m^{-1}\left(E_m\left(x \oplus \alpha_1\right) \oplus \beta_2\right) = \alpha_1\right).$$

Boomerang Tables. When E_m is composed of a single S-box layer, the Boomerang Connectivity Table (`BCT`) would become an effective tool for calculating probability r.

Definition 6 (`BCT` [30])**.** *Let S be a bijective function over $\mathbb{F}_2^n$, and $\alpha_0, \beta_1 \in \mathbb{F}_2^n$. The* `BCT` *of S is a two-dimensional table defined as:*

$$\texttt{BCT}(\alpha_0, \beta_1) = \#\{x \in \mathbb{F}_2^n | S^{-1}(S(x) \oplus \beta_1) \oplus S^{-1}(S(x \oplus \alpha_0) \oplus \beta_1) = \alpha_0\}.$$

Later, based on the `BCT`, a series of tables such as `UBCT` were proposed to more accurately calculate the probability r of the middle layer E_m. These tables go beyond the limitation of E_m being composed of a single S-box layer.

Definition 7 (UBCT, LBCT [46, 47]). *Let S be a bijective function over $\mathbb{F}_2^n$, and $\alpha_0, \alpha_1, \beta_0, \beta_1 \in \mathbb{F}_2^n$. The Upper* BCT *(*UBCT*) and the Lower* BCT *(*LBCT*) of S are three-dimensional tables defined as:*

$$\mathtt{UBCT}(\alpha_0, \alpha_1, \beta_1) = \# \left\{ x \in \mathbb{F}_2^n \middle| \begin{array}{l} S(x) \oplus S(x \oplus \alpha_0) = \alpha_1 \\ S^{-1}(S(x) \oplus \beta_1) \oplus S^{-1}(S(x \oplus \alpha_0) \oplus \beta_1) = \alpha_0 \end{array} \right\},$$

$$\mathtt{LBCT}(\alpha_0, \beta_0, \beta_1) = \# \left\{ x \in \mathbb{F}_2^n \middle| \begin{array}{l} S(x) \oplus S(x \oplus \beta_0) = \beta_1 \\ S^{-1}(S(x) \oplus \beta_1) \oplus S^{-1}(S(x \oplus \alpha_0) \oplus \beta_1) = \alpha_0 \end{array} \right\}.$$

In recent works, the authors of [21, 35, 48] respectively proposed a series of boomerang tables, including FBCT [48], GBDT[35], and GDBCT[21], to more accurately characterize the probability of the boomerang switch.

B Related-Key Impossible Boomerang Key Recovery Attack proposed in [20, 21]

In Figure 10, we provide an outline of the impossible boomerang attack. Suppose that $\Delta_X \nrightarrow \Delta_Y$ over E_D is an impossible boomerang distinguisher and E_B and E_F are added by extending the distinguisher backward and forward. Suppose the input difference of the distinguisher Δ_X propagates backward with probability 1 to Δ_B over E_B^{-1}, the output difference of the distinguisher Δ_Y propagates forward with probability 1 to Δ_F over E_F. Let V_B be the space spanned by all possible Δ_B where $r_B = \log_2 |V_B|$ denotes the dimension of the space. Let V_F be the space spanned by all possible Δ_F where $r_F = \log_2 |V_F|$ denotes the dimension of the space. Let c_B and c_F denote the number of bit conditions satisfying $\Delta_B \to \Delta_X$ and $\Delta_F \to \Delta_Y$, respectively. Let k_B and k_F be the subkey bits involved in E_B and E_F to verify $\Delta_B \to \Delta_X$ and $\Delta_F \to \Delta_Y$, respectively.

Here, we briefly recall the works in [20] and [21]; for detailed descriptions, please refer to the original papers. Method 1 refers to the Key Recovery with Quartet Filtering in [20] and the Impossible Differential Style in [21]. Method 2 refers to the Key Recovery with Pair Filtering in [20] and the Boomerang Style in [21]. Both methods are under related-key settings.

B.1 Method 1

Choose 2^y structures of 2^{r_B} plaintexts each, we can construct 2^{y+2r_B} pairs satisfying Δ_B. To go further, there would be $Q = 2^{2y+4r_B+2r_F-2n}$ quartets satisfying Δ_B and Δ_F constructed.

Thus the data complexity of the related-key IB attack is given by

$$D = 2^{y+r_B+2}.$$

For the time complexity, it is composed of the time used for collecting quartets, guess-and-filter, and exhaustive search. In the phase of quartets collection, the time

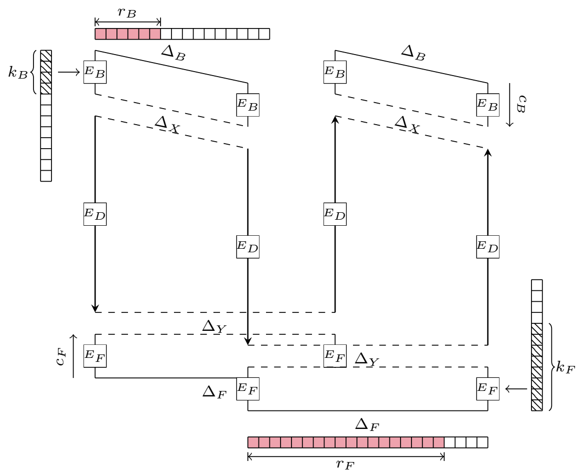


Fig. 10: Outline of the impossible boomerang attack

complexity is:

$$T_0 + T_1 = D + Q.$$

With the early abort technique [29], the time complexity for guess-and-filter phase would be bounded by

$$T_2 = 2^{|k_B \cup k_F|} \frac{Q}{2^{2c_B+2c_F}}.$$

The time complexity of the final phase, exhaustive search, is:

$$T_3 = 2^k P,$$

where P denotes the probability that a trial key is kept in the candidate keys:

$$P = (1 - 2^{-(2c_B+2c_F)})^Q.$$

Let g be the number of key bits we can retrieve through the guess-and-filter phase, i.e., $P = 2^{-g}$, $1 < g \leq |k_B \cup k_F|$. Due to $(1 - 2^{-2(c_B+c_F)})^Q \approx e^{-Q2^{-2(c_B+c_F)}}$. Then we would have $Q = 2^{2c_B+2c_F+\log_2(g)-0.53}$. Let $LG(g) = \log_2(g) - 0.53$, thus the time complexity of the IB attack can be reformulated as follows:

$$\begin{aligned} T_0 &= D, \\ T_1 &= 2^{2c_B+2c_F+LG(g)}, \\ T_2 &= 2^{|k_B \cup k_F|+LG(g)}, \\ T_3 &= 2^{k-g}. \end{aligned}$$

Then we would have the total time complexity of the IB attack:

$$T = (T_0 + (T_1 + T_2)C_{E'} + T_3)C_E,$$

where C_E denotes the cost of one encryption and $C_{E'}$ is the ratio of the cost of partial encryption to the full encryption.

B.2 Method 2

Choose 2^y structures of 2^{r_B} plaintexts each, thus the data complexity of theattack is given by

$$D = 2^{y+r_B+2}.$$

Guess $2^{|k_B|}$ possible values of k_B, we can obtain 2^{y+r_B} pairs, which satisfy the difference Δ_X after encryption over E_B. To go further, there would be $Q = 2^{2y+2r_B+2r_F-2n}$ quartets under each guessed key satisfying Δ_X and Δ_F constructed. These Q quartets will be used in the guess-and-filter phase to eliminate incorrect keys.

For the time complexity, it is composed of the time used for collecting quartets, guess-and-filter, and exhaustive search. In the phase of quartets collection, the time complexity is:

$$T_0 = 2^{|k_B|} 2D$$

and

$$T_1 = 2^{|k_B|} Q.$$

With the early abort technique [29], the time complexity for guess-and-filter phase would be bounded by

$$T_2 = 2^{|k_B \cup k_F|} \frac{Q}{2^{2c_F}}.$$

The time complexity of the final phase, exhaustive search, is:

$$T_3 = 2^k P,$$

where P denotes the probability that a trial key is kept in the candidate keys:

$$P = (1 - 2^{-2c_F})^Q.$$

Let g be the number of key bits we can retrieve through the guess-and-filter phase, i.e., $P = 2^{-g}$, $1 < g \leq |k_B \cup k_F|$. Due to $(1 - 2^{-2c_F})^Q \approx e^{-Q 2^{-2c_F}}$. Then we would have $Q = 2^{2c_F + \log_2(g) - 0.53}$. Let $LG(g) = \log_2(g) - 0.53$, thus the time complexity of the IB attack can be reformulated as follows:

$$\begin{aligned} T_0 &= 2^{|k_B|} \cdot 2D, \\ T_1 &= 2^{|k_B| + 2c_F + LG(g)}, \\ T_2 &= 2^{|k_B \cup k_F| + LG(g)}, \\ T_3 &= 2^{k-g}. \end{aligned}$$

And the memory complexity of the IB attack would be determined by $M = \min\{Q, 2^{|k_B \cup k_F|}\}$.

Then we would have the total time complexity of the IB attack:

$$T = (D + (T_0 + T_1 + T_2) C_{E'} + T_3) C_E,$$

where C_E denotes the cost of one encryption and $C_{E'}$ is the ratio of the cost of partial encryption to the full encryption.

C Complexity Analysis of the Related-(Twea)key ID Attack within the Generic Framework

In the related-(twea)key setting, the adversary has the ability to control the key differences and perform encryption/decryption of plaintext/ciphertext data using two keys, K_0 and $K_1 = K_0 \oplus \Delta K$. For the related-(twea)key impossible differential attack, due to the presence of key difference, plaintext data P in each plaintext structure can generate two distinct pairs $((P, K_0), (P \oplus \Delta_B, K_1))$ and $((P \oplus \Delta_B, K_0), (P, K_1))$ (distinguishing it from the single-(twea)key impossible differential attack). Under the related-(twea)key setting, the data complexity will include the ciphertext obtained from plaintext under two related keys. For the complexity analysis of related-(twea)key ID attacks, the data complexity and the time complexity of T_0 will change, while other terms remain.

The data complexity of the related-(twea)key ID attack would be:

$$D = \max\left\{2^{\frac{c_B+c_F+n-r_F+LG(g)+x_1-c'_B}{2}+1}, 2^{n+c_B+c_F-r_B-r_F+LG(g)+1}\right\}.$$

The time complexity of the related-(twea)key ID attack would be:

$$T_0 = 2^{|k'_B \cup k'_F|} \cdot D,\ T_1 = 2^{|k'_B \cup k'_F| + c^*_B + c^*_F + LG(g)},\ T_2 = 2^{|k_B \cup k_F| + LG(g)},\ T_3 = 2^{k-g}.$$

The memory complexity of the related-(twea)key ID attack would be:

$$M = \min\{2^{c^*_B + c^*_F + LG(g)}, 2^{|k_B \cup k_F|}\}$$

for storing the $N = 2^{c^*_B + c^*_F + LG(g)}$ pairs or the discarded key candidates.

D Algorithms for Fast Identification of Zero Entries in `iU/LBCT`

In Algorithm 1, for a specific α_0, a pair (α_1, α'_1) that simultaneously satisfy $\mathtt{DDT}(\alpha_0, \alpha'_1) > 0$ and $\mathtt{DDT}(\alpha_0, \alpha_1) > 0$ would hold a probability of 2^{-2} approximately under the assumption of an S-box with differential uniformity 2. And for a specific β_2, a pair (α_1, α'_1) satisfying $\mathtt{GBCT}(\alpha_1, \alpha'_1; \beta_2, \beta_2) > 0$ would hold a probability of 2^{-1} approximately with an experimental result for the target ciphers in this work. Then the computational complexity of the Algorithm 1 would be approximated by 2^{3n}. The computation of `iLBCT` in Algorithm 2 is similar to Algorithm 1. We have experimentally verified it for 4-bit and 8-bit Sboxes.

Algorithm 1: The algorithm for fast identification of zero entries in `iUBCT`

Construct the `DDT` with complexity of $\mathcal{O}(2^{2n})$ and the `GBCT`$(\alpha, \alpha'; \beta, \beta)$ with complexity of $\mathcal{O}(2^{3n})$ (refer to the construction for `BCT` in [49]);
Initialize an empty table `iUBCT` with all 0 entries.
for *all values of* $\alpha_0 \in \mathbb{F}_2^n$ **do**
  for *all values of* $\beta_2 \in \mathbb{F}_2^n$ **do**
    flag $=$ *False*;
    for *all values of* $\alpha_1 \in \mathbb{F}_2^n$ **do**
      for *all values of* $\alpha'_1 \in \mathbb{F}_2^n$ **do**
        if `DDT`$(\alpha_0, \alpha'_1) > 0 \wedge$ `DDT`$(\alpha_0, \alpha_1) > 0 \wedge$ `GBCT`$(\alpha_1, \alpha'_1; \beta_2, \beta_2) > 0$ **then**
          `iUBCT`$(\alpha_0, \beta_2) = 1$;
          flag$=$*True*;
          break;
      if *flag* **then**
        break;

Algorithm 2: The algorithm for fast identification of zero entries in `iLBCT`

Construct the `DDT` with complexity of $\mathcal{O}(2^{2n})$ and the `GBCT`$(\alpha, \alpha; \beta, \beta')$ with complexity of $\mathcal{O}(2^{3n})$;
Initialize an empty table `iLBCT` with all 0 entries.
for *all values of* $\alpha_0 \in \mathbb{F}_2^n$ **do**
  for *all values of* $\beta_2 \in \mathbb{F}_2^n$ **do**
    flag $=$ *False*;
    for *all values of* $\beta_1 \in \mathbb{F}_2^n$ **do**
      for *all values of* $\beta'_1 \in \mathbb{F}_2^n$ **do**
        if `DDT`$(\beta_1, \beta_2) > 0 \wedge$ `DDT`$(\beta'_1, \beta_2) > 0 \wedge$ `GBCT`$(\alpha_0, \alpha_0; \beta_1, \beta'_1) > 0$ **then**
          `iLBCT`$(\alpha_0, \beta_2) = 1$;
          flag$=$*True*;
          break;
      if *flag* **then**
        break;

E New 19-Round Related-Tweakey Impossible Boomerang Distinguisher for SKINNY-128-384

In this section, we provide a 19-Round related-tweakey impossible boomerang distinguisher for SKINNY-128-384 based on iLBCT in Figure 11. All of the zero entries in the iU/LBCT of the SKINNY-128-384 S-box are listed in Appendix E.1.

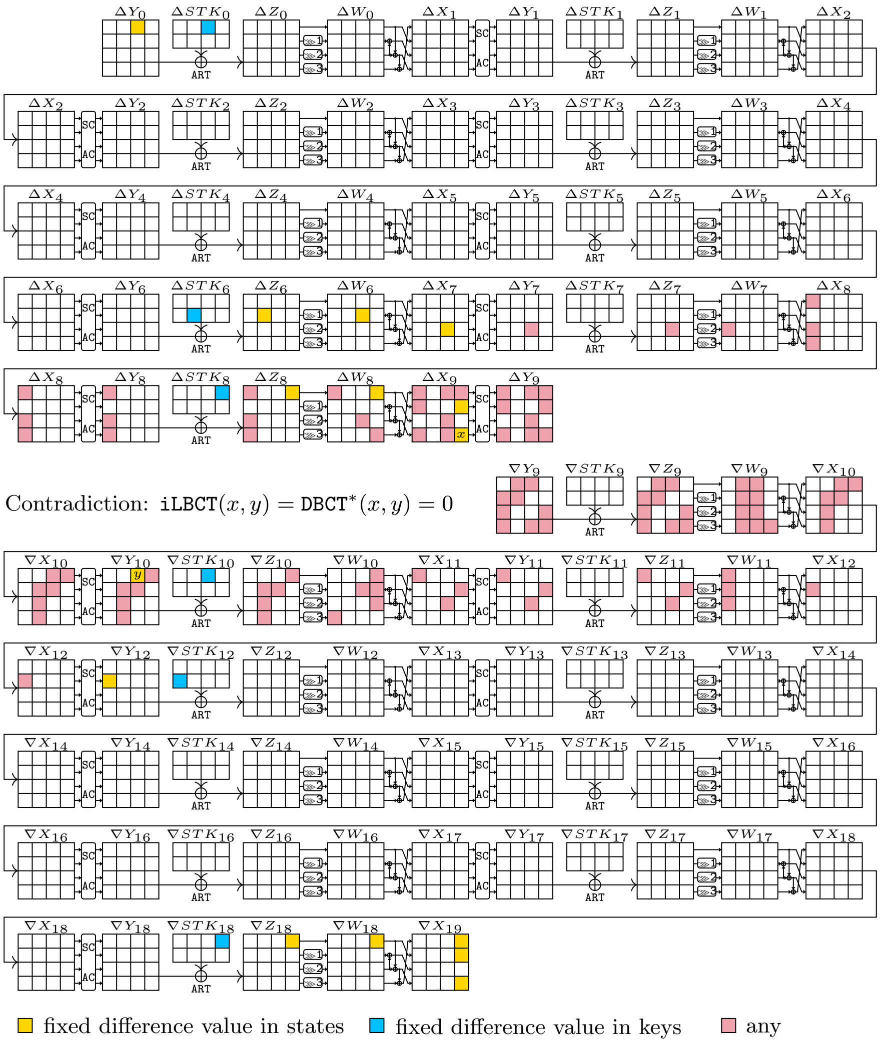


Fig. 11: 19-Round Related-Tweakey Impossible Boomerang Distinguisher for SKINNY-128-384

E.1 The entries with zero value in the `iUBCT` and `iLBCT` of the SKINNY-128 S-box

Table 5: The entries with zero value in the `iUBCT` and `iLBCT` of the SKINNY-128 S-box

$\texttt{iUBCT}(a, b) = 0$ (Hex)
$(a, 24)$, $(a, 34)$, $(a, a4)$, $(a, b4)$, $(20, 88)$, $(20, 89)$, $(20, 8c)$, $(20, 8d)$, $(20, a8)$, $(20, a9)$, $(20, ac)$, $(20, ad)$, $(40, 12)$, $(40, 13)$, $(40, 16)$, $(40, 17)$, $(40, 32)$, $(40, 33)$, $(40, 36)$, $(40, 37)$, $(50, 12)$, $(50, 13)$, $(50, 16)$, $(50, 17)$, $(50, 32)$, $(50, 33)$, $(50, 36)$, $(50, 37)$, $(60, 12)$, $(60, 13)$, $(60, 16)$, $(60, 17)$, $(60, 32)$, $(60, 33)$, $(60, 36)$, $(60, 37)$, $(70, 12)$, $(70, 13)$, $(70, 16)$, $(70, 17)$, $(70, 32)$, $(70, 33)$, $(70, 36)$, $(70, 37)$, $(80, c2)$, $(80, c3)$, $(80, d2)$, $(80, d3)$, $(90, c2)$, $(90, c3)$, $(90, d2)$, $(90, d3)$, $(c0, 12)$, $(c0, 13)$, $(c0, 16)$, $(c0, 17)$, $(c0, 32)$, $(c0, 33)$, $(c0, 36)$, $(c0, 37)$, $(d0, 12)$, $(d0, 13)$, $(d0, 16)$, $(d0, 17)$, $(d0, 32)$, $(d0, 33)$, $(d0, 36)$, $(d0, 37)$, $(e0, 12)$, $(e0, 13)$, $(e0, 16)$, $(e0, 17)$, $(e0, 32)$, $(e0, 33)$, $(e0, 36)$, $(e0, 37)$, $(f0, 12)$, $(f0, 13)$, $(f0, 16)$, $(f0, 17)$, $(f0, 32)$, $(f0, 33)$, $(f0, 36)$, $(f0, 37)$
$\texttt{iLBCT}(a, b) = 0$ (Hex)
$(2, 20)$, $(2, 21)$, $(2, 24)$, $(2, 25)$, $(4, 10)$, $(4, 11)$, $(4, 14)$, $(4, 15)$, $(5, 10)$, $(5, 11)$, $(5, 14)$, $(5, 15)$, $(12, 20)$, $(12, 21)$, $(12, 24)$, $(12, 25)$, $(14, 10)$, $(14, 11)$, $(14, 14)$, $(14, 15)$, $(15, 10)$, $(15, 11)$, $(15, 14)$, $(15, 15)$, $(17, 20)$, $(20, 1)$, $(20, 5)$, $(20, 41)$, $(20, 45)$, $(20, 51)$, $(20, 55)$, $(20, c1)$, $(20, c5)$, $(20, d1)$, $(20, d5)$, $(22, 1)$, $(22, 20)$, $(22, 21)$, $(22, 24)$, $(22, 25)$, $(22, 41)$, $(22, 51)$, $(24, 10)$, $(24, 11)$, $(24, 14)$, $(24, 15)$, $(25, 10)$, $(25, 11)$, $(25, 14)$, $(25, 15)$, $(32, 20)$, $(32, 21)$, $(32, 24)$, $(32, 25)$, $(34, 10)$, $(34, 11)$, $(34, 14)$, $(34, 15)$, $(35, 10)$, $(35, 11)$, $(35, 14)$, $(35, 15)$, $(37, 20)$, $(42, 20)$, $(42, 21)$, $(42, 24)$, $(42, 25)$, $(44, 10)$, $(44, 11)$, $(44, 14)$, $(44, 15)$, $(45, 10)$, $(45, 11)$, $(45, 14)$, $(45, 15)$, $(4c, 10)$, $(4c, 30)$, $(4d, 10)$, $(4d, 30)$, $(4e, 10)$, $(4f, 10)$, $(52, 20)$, $(52, 21)$, $(52, 24)$, $(52, 25)$, $(54, 10)$, $(54, 11)$, $(54, 14)$, $(54, 15)$, $(55, 10)$, $(55, 11)$, $(55, 14)$, $(55, 15)$, $(5c, 10)$, $(5c, 30)$, $(5d, 10)$, $(5d, 30)$, $(5e, 10)$, $(5f, 10)$, $(62, 20)$, $(62, 21)$, $(62, 24)$, $(62, 25)$, $(64, 10)$, $(64, 11)$, $(64, 14)$, $(64, 15)$, $(65, 10)$, $(65, 11)$, $(65, 14)$, $(65, 15)$, $(6c, 10)$, $(6c, 30)$, $(6d, 10)$, $(6d, 30)$, $(6e, 10)$, $(6f, 10)$, $(72, 20)$, $(72, 21)$, $(72, 24)$, $(72, 25)$, $(74, 10)$, $(74, 11)$, $(74, 14)$, $(74, 15)$, $(75, 10)$, $(75, 11)$, $(75, 14)$, $(75, 15)$, $(7c, 10)$, $(7c, 30)$, $(7d, 10)$, $(7d, 30)$, $(7e, 10)$, $(7f, 10)$, $(82, 20)$, $(82, 21)$, $(82, 24)$, $(82, 25)$, $(84, 10)$, $(84, 11)$, $(84, 14)$, $(84, 15)$, $(85, 10)$, $(85, 11)$, $(85, 14)$, $(85, 15)$, $(92, 20)$, $(92, 21)$, $(92, 24)$, $(92, 25)$, $(94, 10)$, $(94, 11)$, $(94, 14)$, $(94, 15)$, $(95, 10)$, $(95, 11)$, $(95, 14)$, $(95, 15)$, $(a2, 20)$, $(a2, 21)$, $(a2, 24)$, $(a2, 25)$, $(a4, 10)$, $(a4, 11)$, $(a4, 14)$, $(a4, 15)$, $(a5, 10)$, $(a5, 11)$, $(a5, 14)$, $(a5, 15)$, $(b2, 20)$, $(b2, 21)$, $(b2, 24)$, $(b2, 25)$, $(b4, 10)$, $(b4, 11)$, $(b4, 14)$, $(b4, 15)$, $(b5, 10)$, $(b5, 11)$, $(b5, 14)$, $(b5, 15)$, $(c2, 20)$, $(c2, 21)$, $(c2, 24)$, $(c2, 25)$, $(c4, 10)$, $(c4, 11)$, $(c4, 14)$, $(c4, 15)$, $(c5, 10)$, $(c5, 11)$, $(c5, 14)$, $(c5, 15)$, $(d2, 20)$, $(d2, 21)$, $(d2, 24)$, $(d2, 25)$, $(d4, 10)$, $(d4, 11)$, $(d4, 14)$, $(d4, 15)$, $(d5, 10)$, $(d5, 11)$, $(d5, 14)$, $(d5, 15)$, $(e2, 20)$, $(e2, 21)$, $(e2, 24)$, $(e2, 25)$, $(e4, 10)$, $(e4, 11)$, $(e4, 14)$, $(e4, 15)$, $(e5, 10)$, $(e5, 11)$, $(e5, 14)$, $(e5, 15)$, $(f2, 20)$, $(f2, 21)$, $(f2, 24)$, $(f2, 25)$, $(f4, 10)$, $(f4, 11)$, $(f4, 14)$, $(f4, 15)$, $(f5, 10)$, $(f5, 11)$, $(f5, 14)$, $(f5, 15)$

F `i3UBCT/i3MBCT/i3LBCT`: Extending `iUBCT/iLBCT` to 3 rounds

Definition 8 (`i3UBCT/i3MBCT/i3LBCT`)**.** *Let S be a bijective function over $\mathbb{F}_2^n$, and $\alpha_0, \beta_3 \in \mathbb{F}_2^n$. The* `i3UBCT`*,* `i3MBCT` *and* `i3LBCT`*, with specific definitions provided as:*

$$\texttt{i3UBCT}(\alpha_0, \beta_3) = \# \left\{ (\alpha_1, \alpha_1', \alpha_2, \alpha_2') \in (\mathbb{F}_2^n)^4 \;\middle|\; \begin{array}{l} \texttt{DDT}(\alpha_0, \alpha_1) > 0, \texttt{DDT}(\alpha_1, \alpha_2) > 0, \\ \texttt{DDT}(\alpha_0, \alpha_1') > 0, \texttt{DDT}(\alpha_1', \alpha_2') > 0, \\ \texttt{GBCT}(\alpha_2, \alpha_2'; \beta_3, \beta_3) > 0 \end{array} \right\},$$

$$\texttt{i3MBCT}(\alpha_0, \beta_3) = \# \left\{ (\alpha_1, \alpha_1', \beta_2, \beta_2') \in (\mathbb{F}_2^n)^4 \;\middle|\; \begin{array}{l} \texttt{DDT}(\alpha_0, \alpha_1) > 0, \texttt{DDT}(\beta_2, \beta_3) > 0, \\ \texttt{DDT}(\alpha_0, \alpha_1') > 0, \texttt{DDT}(\beta_2', \beta_3) > 0, \\ \texttt{GBCT}(\alpha_1, \alpha_1'; \beta_2, \beta_2') > 0 \end{array} \right\},$$

$$\texttt{i3LBCT}(\alpha_0, \beta_3) = \# \left\{ (\beta_1, \beta_1', \beta_2, \beta_2') \in (\mathbb{F}_2^n)^4 \;\middle|\; \begin{array}{l} \texttt{DDT}(\beta_1, \beta_2) > 0, \texttt{DDT}(\beta_2, \beta_3) > 0, \\ \texttt{DDT}(\beta_1', \beta_2') > 0, \texttt{DDT}(\beta_2', \beta_3) > 0, \\ \texttt{GBCT}(\alpha_0, \alpha_0; \beta_1, \beta_1') > 0 \end{array} \right\},$$

G Supplementary Materials for the MILP Model Introduced in Section 5

G.1 Constraints for Deterministic Propagations in SubBytes

Let $y = S(x)$, then the constraints of deterministic propagation in the S-box can be written as:

$$\begin{cases} x^{s_0} - y^{s_0} = 0 \\ x^{s_1} - y^{s_1} \geq 0 \\ x^{s_0} - y^{s_1} \geq 0 \\ -x^{s_0} - x^{s_1} + y^{s_1} \geq -1 \end{cases}$$

G.2 Constraints for Deterministic Propagations in MixColumns

Let $z = x \oplus y$, then the constraints of deterministic propagation for the XOR operation can be written as:

$$\begin{cases} -x^{s_1} - y^{s_1} + z^{s_1} \geq -1 \\ x^{s_1} - z^{s_1} \geq 0 \\ y^{s_1} - z^{s_1} \geq 0 \\ -x^{s_0} - y^{s_0} + z^{s_0} \geq -1 \\ x^{s_1} + y^{s_0} + z^{s_0} \geq 1 \\ x^{s_0} + y^{s_1} + z^{s_0} \geq 1 \\ -x^{s_0} - x^{s_1} + y^{s_0} - z^{s_0} \geq -2 \\ x^{s_0} - y^{s_0} - y^{s_1} - z^{s_0} \geq -2 \end{cases}$$

G.3 Constraints for Probabilistic Propagations in SubBytes

Let $y = S(x)$, then the constraints of probabilistic propagation in the S-box can be written as:

$$\begin{cases} x^{s_0} - y^{s_0} = 0 \\ -x^{s_1} + y^{s_1} \geq 0 \\ x^{s_1} - y^{s_0} - y^{s_1} \geq -1 \end{cases}$$

G.4 Constraints for Probabilistic Propagations in `MixColumns`

Let $z = x \oplus y$, then the constraints of probabilistic propagation for the `XOR` operation can be written as:

$$\begin{cases} -2x^{s_0} - x^{s_1} - 2y^{s_0} - y^{s_1} + 2z^{s_0} + z^{s_1} \geq -3 \\ -x^{s_1} - y^{s_1} + z^{s_1} \geq -1 \\ x^{s_0} + y^{s_1} + z^{s_0} + z^{s_1} - 1 \geq 0 \\ x^{s_1} + y^{s_0} + z^{s_0} + z^{s_1} - 1 \geq 0 \\ -x^{s_0} - x^{s_1} + y^{s_0} - z^{s_0} \geq -2 \\ -x^{s_0} - x^{s_1} + y^{s_1} - z^{s_1} \geq -2 \\ x^{s_0} - y^{s_0} - y^{s_1} - z^{s_0} \geq -2 \\ x^{s_1} - y^{s_0} - y^{s_1} - z^{s_1} \geq -2 \end{cases}$$

G.5 Constraints for Identifying Pre-guessed Keys in the Data Collection Phase

The subkey cells need to be pre-guessed for verifing the conditions of Y_r can be calculated by

$$\texttt{if } Y_{r,i}^{up,v} = 1, \texttt{ then } K_{r,i}^{up,p} = 1,$$

and the subkey cells need to be pre-guessed for verifing the conditions of W_r can be calculated by

$$\texttt{if } \big((W_{r,i}^{up,v} = 1) \wedge (X_{r,j}^{up,s0} + X_{r,j}^{up,s1} \leq 1)\big) = 1, \texttt{ then } K_{r,j}^{up,p} = 1 \texttt{ for } \forall j \in {}^tL^D(i).$$

In addition, we can backtrack to calculate all the pre-guessed subkey cells from round 0 using

$$\texttt{if } K_{r,i}^{up,p} = 1, \texttt{ then } K_{r-1,j}^{up,p} = 1 \texttt{ for } \forall j \in {}^tL^D(i).$$

G.6 Linear Layer Matrix $\boldsymbol{L_E}$ and $\boldsymbol{L_D}$ used in Section 5.1

For SKINNY-family,

$$L^E = \begin{pmatrix}
1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1\\
1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

$$L^D = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0
\end{pmatrix}$$

For AES and Deoxys-BC,

$$L^E = \begin{pmatrix}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}$$

$$L^D = \begin{pmatrix}
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0
\end{pmatrix}$$

G.7 Algorithm for the optimization and verification procedure of the model

The algorithm for the optimization and verification procedure of the model is provided in algorithm 3.

Algorithm 3: Optimizing Model and Verifying Contradictions

Input: model $\mathcal{M}$, number of target rounds r, state variables $X_0...X_{r-1}, Y_0...Y_{r-1}, Z_0...Z_{r-1}, W_0...W_{r-1}$, key variables $K_0...K_{r-1}$, contradiction variables $\mathtt{contr}_{1,2,3}$
Output: minimum time complexity T
Initialize: $flag = 0$
while $flag = 0$ **do**
  Add constraints on the round functions to model $\mathcal{M}$
  Add constraints on the involved subkey cells to model $\mathcal{M}$
  Add constraints on the contradictions to model $\mathcal{M}$
  Compute r'_B, r'_F, c'_B, c'_F, k'_B, k'_F and add to model $\mathcal{M}$
  $\mathcal{M} \leftarrow \bigvee_{i=1}^{3} \mathtt{contr_i} = 1$
  $T \leftarrow \mathcal{M}.sol()$
  for each i **do**
    if $\mathtt{contr_i} = 1$ **then**
      Verify the contradiction with $\mathtt{BCT}(\mathtt{contr_1} = 1)$, $\mathtt{iUBCT}/\mathtt{iLBCT}(\mathtt{contr_2} = 1)$, and $\mathtt{i3UBCT}/\mathtt{i3MBCT}/\mathtt{i3LBCT}(\mathtt{contr_3} = 1)$
      if an instance exists **then**
        $flag = 1$
        break
      else
        $\mathcal{M} \leftarrow \mathtt{contr_i} = 0$
      end if
    end if
  end for
end while
return T

H Applications to SKINNY-family

H.1 Specification

SKINNY is a tweakable block cipher family following the TWEAKEY framework, first proposed by Beierle *et al.* at CRYPTO 2016 [39]. SKINNY family has 6 versions, denoted by SKINNY-n-t: $n \in \{64, 128\}$ is the block size and $t \in \{n, 2n, 3n\}$ is the tweakey size. The cell size c is 4 for $n = 64$ and 8 for $n = 128$.

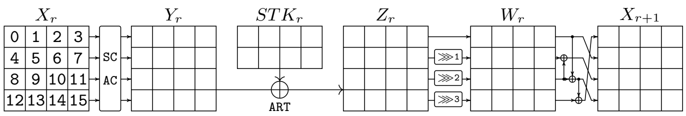


Fig. 12: Round function of SKINNY

The SKINNY round function (see Figure 12) applies five transformations: `SubCells` (`SC`), `AddConstants` (`AC`), `AddRoundTweakey` (`ART`), `ShiftRows` (`SR`), `MixColumns` (`MC`). The `SC` operation applies a 4-bit (resp. 8-bit) S-box on each cell for SKINNY-64 (resp. SKINNY-128). The `AC` operation XORs the round constant to the internal state. The `ART` operation XORs the first and second rows of subtweakey with the corresponding cells in the internal state. The `SR` operation rotates the 4-cell i-th row right by i positions, $i = 0, 1, 2, 3$. The `MC` operation multiplies the internal state by a binary matrix $\mathrm{M} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$.

The tweakey schedule of SKINNY is a linear algorithm, which divides the master tweakey into z tweakey arrays $(TK1, ..., TKz)$ with n-bit length each, where $z = \frac{t}{n} \in \{1, 2, 3\}$. $TK1$, $TK2$ and $TK3$ follow three independent update functions. The subtweakey used in r-th round STK_r is generated from:

- $STK_r = TK1_r$ if $z = 1$,
- $STK_r = TK1_r \oplus TK2_r$ if $z = 2$,
- $STK_r = TK1_r \oplus TK2_r \oplus TK3_r$ if $z = 3$,

where $TK1_r, TK2_r, TK3_r$ denote the tweakey arrays in round r and are generated as follows. First, a permutation h is applied to each tweakey array as $TKz_{r+1}[i] \leftarrow TKz_r[h[i]]$. Next, each cell of the first and second rows of $TK2_r$ and $TK3_r$ are individually updated with an LFSR. For more details on the specification of SKINNY, please refer to [39].

H.2 Related-Tweakey Impossible Differential Attack on SKINNY-n-n

This section provides a 19-round related-tweakey impossible differential attack against SKINNY-n-n. In this attack, we use a 12-round related-tweakey impossible differential as:

$$\begin{aligned}(\Delta Y_3, \Delta STK_3) &= ((00a0|0000|a000|00?0), (0000|0a00|0000|0000)) \\ &\nrightarrow (\Delta X_{16}) = (0000|0000|0a00|0000),\end{aligned}$$

where a denotes a fixed non-zero difference and ? denotes any difference. We prefix 4 rounds at the beginning and append 3 rounds at the end of the distinguisher to mount the attack, as shown in Figure 13. From the figure, we can get the parameters used for this attack: $r_B = 8c$, $c_B = 7c$, $r_F = 6c$, $c_F = 6c$, $|k_B \cup k_F| = 13c$, $c'_B = 0$, $c'_F = 0$, $|k'_B \cup k'_F| = 0$. In the key-recovery extensions of this attack, it adopts deterministic extensions.

We use Lemma 2 in the complexity analysis of our attacks.

Lemma 2. *For a given S-box and any nonzero input-output difference pair (δ_i, δ_o), there would exist one solution x on average for the equation $S(x) \oplus S(x \oplus \delta_i) = \delta_o$ holds true.*

Fig. 13: The related-tweakey impossible differential attack against 19-round SKINNY-n-n

Pairs Collection. In this phase, we need to collect $N = 2^{c_B+c_F+LG(g)} = 2^{13c+LG(g)}$ pairs to eliminate the wrong keys.

Guess-and-Filter. For N pairs:

1. *Satisfying the cell conditions in* ΔW_{17}. From the ciphertexts, we can know the value of $X_{18}[8-15]$. With the condition $\Delta W_{17}[13] = \Delta X_{18}[1] \oplus \Delta X_{18}[13] = 0$, we can deduce the difference value $\Delta X_{18}[1]$. With known $\Delta Y_{18}[1]$ and Lemma 2, we can compute $X_{18}[1]$, $Y_{18}[1]$ and $\underline{STK_{18}[1]}$. Similarly, we can also compute

$X_{18}[3,7]$, $Y_{18}[3,7]$ and $\underline{STK_{18}[3,7]}$. The time complexity of this step is $N \cdot 2 \cdot \frac{3}{19 \cdot 16} = N2^{-5.66}$.

2. *Satisfying the cell conditions in* ΔW_{16}. Guess 2^c possible values of $\underline{STK_{18}[2]}$ and compute $X_{17}[15]$ and $Y_{17}[15]$. With the condition $\Delta W_{16}[15] = \Delta X_{17}[3] \oplus \Delta X_{17}[15] = 0$, we can use Lemma 2 and known $\Delta Y_{17}[3]$ to deduce $\underline{STK_{17}[3]}$. Similarly, from the condition $\Delta W_{16}[7] = \Delta X_{17}[11] \oplus \Delta X_{17}[15] = 0$, we can deduce $\underline{STK_{18}[5]}$. The time complexity of this step is $2^c \cdot N \cdot 2 \cdot \frac{3}{19 \cdot 16} = N2^{c-5.66}$.
3. *Satisfying the cell conditions in* ΔX_2. Due to $eSTK_0[4] = STK_{18}[2]$, $eSTK_0[11] = STK_{18}[5]$, we can compute $\Delta W_1[5,9]$. Checking whether $\Delta W_1[5] = \Delta W_1[9]$ or not would be a c-bit filter. With the condition $\Delta X_2[13] = \Delta W_1[1] \oplus \Delta W_1[9] = 0$, we can compute the value of $\Delta W_1[1]$ and then compute $X_1[1]$, $Y_1[1]$ and $\underline{eSTK_0[1]}$ by Lemma 2. Time of this step is $2^c \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{1}{19 \cdot 16} \approx N2^{-7.25}$.
4. *Satisfying the cell conditions in* ΔX_2. Due to $eSTK_0[3] = STK_{18}[7]$, $eSTK_0[12] = STK_{18}[2]$, we can compute $\Delta W_1[3,15]$. With the condition $\Delta X_2[3] = \Delta W_1[3] \oplus \Delta W_1[11] \oplus \Delta W_1[15] = 0$, we can compute $\Delta W_1[11]$ and deduce $\underline{eSTK_0[9]}$ by Lemma 2. Similarly, we can also deduce $\underline{eSTK_0[6]}$ from the condition $\Delta X_2[11] = \Delta W_1[7] \oplus \Delta W_1[11] = 0$. The time complexity of this step is $2^c \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{4}{19 \cdot 16} = N2^{-5.25}$.
5. *Satisfying the cell conditions in* ΔX_{16}. From the previous steps, we can get all the values of $STK_{18}[1-5,7], STK_{17}[3]$ and thus $Y_{17}[15], \Delta Y_{17}[7], Z_{17}[7,15]$. Due to $\Delta Y_{16}[9] = \Delta X_{17}[15]$, we can deduce $Y_{16}[9]$ by Lemma 2, and then compute $\underline{STK_{17}[7]}$. The time complexity of this step is $2^c \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{1}{19 \cdot 16} = N2^{-7.25}$.
6. *Satisfying the cell conditions in* ΔX_3. Guess $\underline{eSTK_0[1]}$. With $STK_1[3] = STK_{17}[3]$ and $STK_1[7] = STK_{17}[7]$, we can compute $\underline{X_2[8]}$ and $X_2[3,7,15]$. From the condition $\Delta X_3[10] = \Delta W_2[6] \oplus \Delta W_2[10] = 0$, we would have one solution $X_2[5]$ from the equation of S-box by Lemma 2 and deduce $\underline{STK_1[1]}$. The time complexity of this step is $2^{2c} \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{2}{19 \cdot 16} = N2^{c-6.25}$.
7. *Satisfying the cell conditions in* ΔY_3. From the previous steps, we can get the values of $W_2[4]$ and $\Delta X_3[8]$. With $\Delta Y_3[8] = a$, one solution $X_3[8]$ could be derived from Lemma 2. Then we can compute $W_2[8]$ and $X_2[10]$. Due to $X_2[10] = Z_1[5] \oplus Z_1[8]$, we would have the actual value of $Z_1[5]$ and then compute $\underline{STK_1[5]}$. Similarly, $\underline{STK_1[2]}$ could also be computed. The time complexity of this step is $2^{2c} \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{2}{19 \cdot 16} = N2^{c-6.25}$.

Complexity. The data complexity is $D = 2^{15c+LG(g)+1}$. The time complexity is $2^{15c+LG(g)+1} + 2^{14c-4.44+LG(g)} + 2^{16c-g}$. Memory complexity is 2^{13c}. For SKINNY-64-64 with $c = 4$, we set $g = 2$, then $D = 2^{61.47}$, $T = 2^{62.76}$, $M = 2^{52}$. For SKINNY-128-128 with $c = 8$, we set $g = 4$, then $D = 2^{122.47}$, $T = 2^{124.43}$, $M = 2^{104}$.

H.3 Single-Tweakey Impossible Differential Attack on SKINNY-n-2n

In this section, we propose a 19-round single-tweakey impossible differential attack against SKINNY-n-2n. In this attack, we use a 12-round single-tweakey impossible differential as:

$$\Delta X_3 = (0000|0000|0000|000?) \nrightarrow \Delta W_{14} = (0000|0000|0?00|0000).$$

We prefix 3 rounds at the beginning and append 4 rounds at the end of the distinguisher to mount the attack, as shown in Figure 14. From the figure, we can get the parameters used for this attack: $r_B = 7c$, $c_B = 6c$, $r_F = 13c$, $c_F = 13c$, $|k_B \cup k_F| = 25c$, $c'_B = 0$, $c'_F = 3c$, $|k'_B \cup k'_F| = 8c$. In the key-recovery extensions of this attack, it adopts probabilistic extensions.

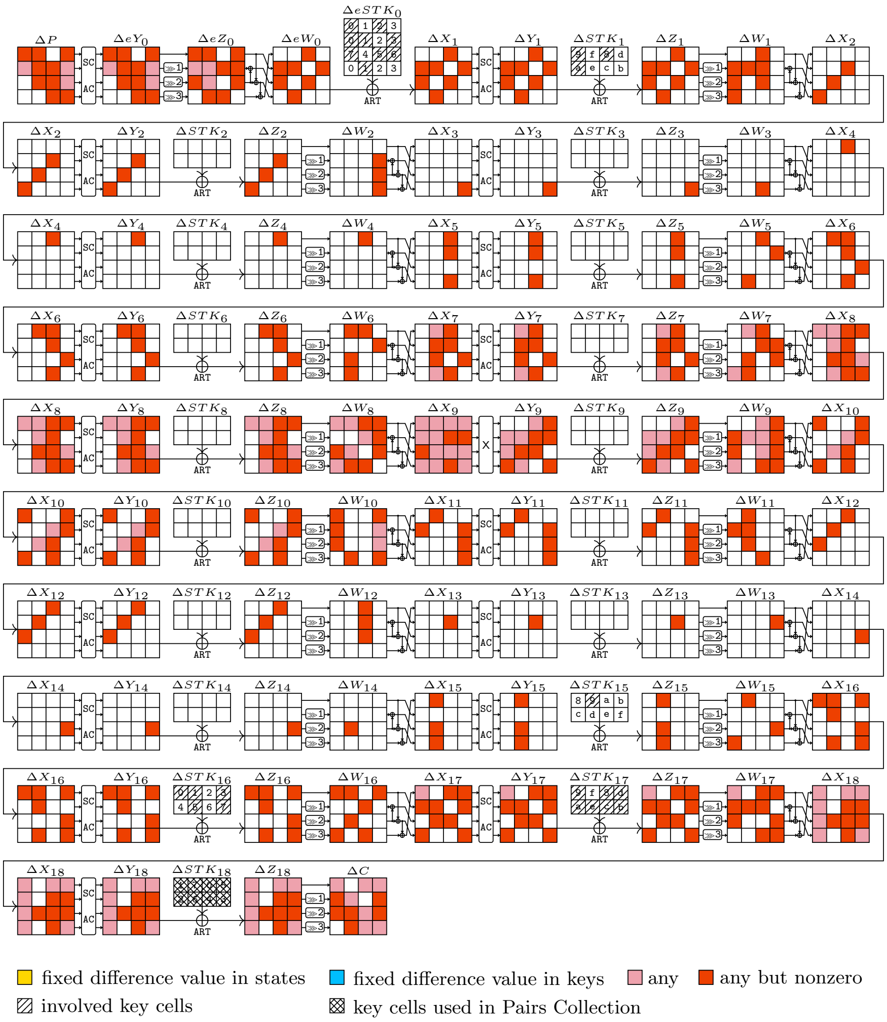


Fig. 14: The single-tweakey impossible differential attack against 19-round SKINNY-n-2n

Pairs Collection. In this phase, we need to collect $N = 2^{c^*_B + c^*_F + LG(g)} = 2^{16c+LG(g)}$ pairs under 2^{8c} pre-guessed subkey bits to eliminate the wrong keys.

Guess-and-Filter. For N pairs under each pre-guessed subkey bits:

1. *Satisfying the cell conditions in* ΔW_{16}. With condition $\Delta W_{16}[4] = \Delta X_{17}[4] \oplus \Delta X_{17}[12] = 0$, we can compute $\underline{STK_{17}[4]}$ and $X_{17}[4]$ by Lemma 2. Similarly, we could deduce the value of $\underline{STK_{17}[3,5,7]}$. Time complexity of this step is $2^{8c} \cdot N \cdot 2 \cdot \frac{4}{19 \cdot 16} = N2^{8c-5.25}$.
2. *Satisfying the cell conditions in* ΔW_{15}. Guess $\underline{STK_{17}[2]}$, we will have a c-bit filter for the condition $\Delta W_{15}[7] = \Delta X_{16}[11] \oplus \Delta X_{16}[15] = 0$. Then with the condition $\Delta W_{15}[15] = \Delta X_{16}[3] \oplus \Delta X_{16}[15] = 0$ and Lemma 2, we would derive the value of $\underline{STK_{16}[3]}$. Similarly, we would compute the value of $\underline{STK_{16}[1,5]}$ by guessing $\underline{STK_{17}[0,6]}$ and using Lemma 2. Time complexity of this step is $2^{9c} \cdot N \cdot 2 \cdot \frac{1}{19 \cdot 16} + 2^{11c} \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{4}{19 \cdot 16} \approx N2^{10c-5.25}$.
3. *Satisfying the cell conditions in* ΔX_2. From previous steps, we have known the value of $STK_{16}[1,3,5]$ and $STK_{18}[0,3,7]$. With the tweakey schedule of SKINNY-n-2n, we can compute $STK_0[1,3,5] = eSTK_0[13,10,7]$. Then we can compute $Y_1[7,10,13]$ and the condition $\Delta W_1[4] = \Delta W_1[8] = \Delta W_1[12]$ will lead to 2 c-bit filters. Time cost of this step would be a negligible one compared with previous steps.
4. *Satisfying the cell conditions in* ΔW_{14}. Guess $\underline{STK_{16}[0,7]}$, and we will know the values of cells in Z_{15}. The condition $\Delta W_{14}[5] = \Delta X_{15}[9] \oplus \Delta X_{15}[13] = 0$ will lead to a c-bit filter. Meanwhile, we would determine $\underline{STK_{15}[1]}$ by applying Lemma 2. Time complexity of this step is $2^{13c} \cdot N \cdot 2^{-3c} \cdot 2 \cdot \frac{2}{19 \cdot 16} = N2^{10c-6.25}$.
5. *Satisfying the cell conditions in* ΔX_2. We can compute $eSTK_0[5,8]$ from the tweakey schedule and known subtweakeys in previous steps, thus the condition $\Delta X_2[10] = \Delta Z_1[5] \oplus \Delta Z_1[8]$ will lead to a c-bit filter. Meanwhile, we would determine $\underline{eSTK_0[2]}$ by applying Lemma 2. Time complexity of this step is $2^{13c} \cdot N \cdot 2^{-4c} \cdot 2 \cdot \frac{1}{19 \cdot 16} = N2^{9c-7.25}$.
6. *Satisfying the cell conditions in* ΔX_3. Guess $\underline{eSTK_0[11]}$ and $\underline{STK_1[4]}$. From previous steps, we have known $STK_{17}[0]$ and $STK_{15}[1]$, thus we can compute $STK_1[1]$ by tweakey schedule and $Y_2[12]$. The condition $\Delta X_3[3] = \Delta Y_2[9] \oplus \Delta Y_2[12] = 0$ will lead to a c-bit filter. With condition $\Delta Y_2[6] = \Delta Y_2[12]$ and $\Delta X_2[6] = \Delta W_1[2]$, we can deduce $X_2[6]$ and $\underline{STK_1[2]}$ by applying Lemma 2. Time complexity of this step is $2^{15c} \cdot N \cdot 2^{-5c} \cdot 2 \cdot \frac{2}{19 \cdot 16} = N2^{10c-6.25}$.

Complexity. The data complexity is $D = 2^{15c+LG(g)+1}$. The time complexity of this whole attack is $2^{8c+15c+LG(g)+1} \cdot \frac{8}{19 \cdot 16} + 2^{16c+10c+LG(g)-4.25} + 2^{32c-g}$. For SKINNY-64-128 with $c = 4$, we set $g = 24$, then $D = 2^{65.05}$, $T = 2^{104.90}$, $M = 2^{68.05}$. For SKINNY-128-256 with $c = 8$, we set $g = 48$, then $D = 2^{126.05}$, $T = 2^{209.45}$, $M = 2^{133.05}$.

H.4 Single-Tweakey Impossible Differential Attack on SKINNY-n-3n

In this section, we propose a 21-round single-tweakey impossible differential attack against SKINNY-n-3n. In this attack, we use a 12-round single-tweakey impossible differential as:

$$\Delta X_5 = (0000|0000|0000|000?) \nrightarrow \Delta W_{16} = (0000|0000|0?00|0000).$$

We prefix 5 rounds at the beginning and append 4 rounds at the end of the distinguisher to mount the attack, as shown in Figure 15. From the figure, we can get the parameters used for this attack: $r_B = 16c$, $c_B = 15c$, $r_F = 13c$, $c_F = 13c$, $|k_B \cup k_F| = 41c$, $c'_B = 0$, $c'_F = 3c$, $|k'_B \cup k'_F| = 8c$. In the key-recovery extensions of this attack, it adopts probabilistic extensions.

Pairs Collection. In this phase, we need to collect $N = 2^{c_B^* + c_F^* + LG(g)} = 2^{25c+LG(g)}$ pairs under 2^{8c} pre-guessed subkey bits.

Guess-and-Filter. For N pairs under each pre-guessed subkey bits:

1. *Satisfying the cell conditions in* ΔW_{18}. With condition $\Delta W_{18}[4] = \Delta X_{19}[4] \oplus \Delta X_{19}[12]$, we can compute $\underline{STK_{19}[4]}$ and $X_{19}[4]$ by Lemma 2. Similarly, we could deduce the value of $\underline{STK_{19}[3,5,7]}$. Time complexity of this step is $2^{8c} \cdot N \cdot 2 \cdot \frac{4}{21 \cdot 16} = N2^{8c-5.39}$.
2. *Satisfying the cell conditions in* ΔX_2. Guess $\underline{eSTK_0[0-3, 8-11]}$ and compute ΔW_1. The conditions $\Delta W_1[0] = \Delta W_1[4] = \Delta W_1[8]$, $\Delta W_1[1] = \Delta W_1[9]$ and $\Delta W_1[2] \oplus \Delta W_1[10] = \Delta W_1[14]$ will lead to four c-bit filters. Time complexity of this step is $2^{16c} \cdot N \cdot 2 \cdot \frac{8}{21 \cdot 16} = N2^{16c-4.39}$.
3. *Satisfying the cell conditions in* ΔX_3. Guess $\underline{STK_1[0,1,2,6]}$ and compute $Y_2[1,4,11,14]$, and so $\Delta Y_2[1,4,11,14]$. The conditions $\Delta Y_2[4] = \Delta Y_2[11]$ and $\Delta Y_2[1] = \Delta Y_2[11] \oplus \Delta Y_2[14]$ will lead to two c-bit filters. Guess $\underline{STK_1[3,4,5]}$ and compute $Y_2[0,3,6,9,10]$. The conditions $\Delta Y_2[0] = \Delta Y_2[10]$ and $\Delta Y_2[3] = \Delta Y_2[6] = \Delta Y_2[9]$ will lead to three c-bit filters. Guess $\underline{STK_1[7]}$. Time complexity of this step could be approximated by $2^{20c} \cdot N \cdot 2^{-4c} \cdot 2 \cdot \frac{4}{21 \cdot 16} + 2^{23c} \cdot N \cdot 2^{-6c} \cdot 2 \cdot \frac{3}{21 \cdot 16} + 2^{24c} \cdot N \cdot 2^{-9c} \cdot 2 \cdot \frac{1}{21 \cdot 16} \approx N2^{17c-5.81}$.
4. *Satisfying the cell conditions in* ΔW_{17}. Guess $\underline{STK_{19}[2]}$ and compute $\Delta X_8[11,15]$, which will lead to a c-bit filter. Guess $\underline{STK_{19}[0,6]}$. With condition $\Delta W_{17}[15] = \Delta X_{18}[3] \oplus \Delta X_{18}[15] = 0$, we can determine $\underline{STK_{18}[3]}$ by applying Lemma 2. Similarly, we can also derive $\underline{STK_{18}[1,5]}$. Time complexity of this step could be approximated by $2^{25c} \cdot N \cdot 2^{-9c} \cdot 2 \cdot \frac{1}{21 \cdot 16} + 2^{27c} \cdot N \cdot 2^{-10c} \cdot 2 \cdot \frac{5}{21 \cdot 16} \approx N2^{17c-5.07}$.
5. *Satisfying the cell conditions in* ΔX_4. From the previous steps, we have known $STK_{20}[0,3,7]$, $STK_{18}[1,3,5]$ and $STK_0[5,6,7]$. With the tweakey schedule of SKINNY-n-3n, we can determine $STK_2[1,3,5]$ and thus compute $\Delta W_3[4,8,12]$. The conditions $\Delta W_3[4] = \Delta W_3[8] = \Delta W_3[12]$ will lead to two c-bit filters.
6. *Satisfying the cell conditions in* ΔW_{16}. Guess $\underline{STK_{18}[0,7]}$ and compute $\Delta X_{17}[9,13]$. The condition $\Delta X_{17}[9] = \Delta X_{17}[13]$ would act as a c-bit filter. Also, we can derive $\underline{STK_{17}[1]}$ with another condition $\Delta X_{17}[1] = \Delta X_{17}[13]$ by applying Lemma 2. Time complexity of this step is $2^{29c} \cdot N \cdot 2^{-12c} \cdot 2 \cdot \frac{2}{21 \cdot 16} = N2^{17c-6.39}$.
7. *Satisfying the cell conditions in* ΔX_4. Deduce $STK_2[1,7]$ from known $STK_{20}[0,1]$, $STK_{18}[1,7]$ and $STK_0[3,7]$. Guess $\underline{STK_2[2]}$. Then we can compute $\Delta W_3[2,6,10]$.

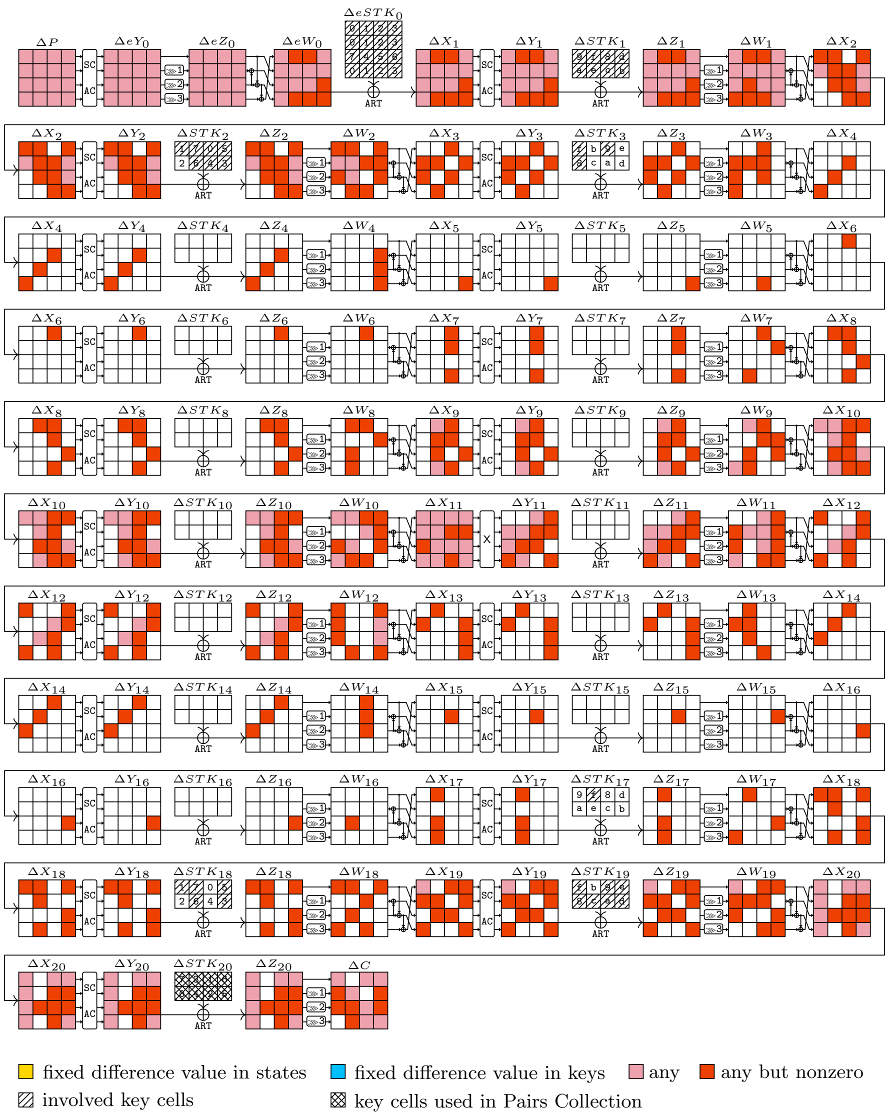


Fig. 15: The single-tweakey impossible differential attack against 21-round SKINNY-n-3n

The conditions $\Delta W_3[2] = \Delta W_3[6] = \Delta W_3[10]$ will lead to two c-bit filters. Time complexity of this step is $2^{30c} \cdot N \cdot 2^{-13c} \cdot 2 \cdot \frac{1}{21 \cdot 16} = N2^{17c-8.39}$.

8. *Satisfying the cell conditions in* ΔX_5. Determine $STK_2[0]$ from $STK_0[1]$, $STK_{18}[0]$ and $STK_{20}[2]$. Guess $\underline{STK_2[6]}$ and $\underline{STK_3[4]}$. With the condition $\Delta X_5[11] = \Delta W_4[7] \oplus \Delta W_4[11] = \overline{\Delta Y_4[6]} \oplus \overline{\Delta Y_4[9]} = 0$, we can determine $X_4[6]$ and thus $\underline{STK_3[2]}$ due to $X_4[6] = Y_3[2] \oplus STK_3[2]$. We can also determine $STK_3[0]$ from

the knowledge of $STK_{19}[0]$, $STK_{17}[1]$ and $STK_1[1]$. The condition $\Delta W_4[11] = \Delta W_4[15]$ would act as a c-bit filter. Time complexity of this step is $2^{32c} \cdot N \cdot 2^{-15c} \cdot 2 \cdot \frac{3}{21 \cdot 16} = N2^{17c-6.81}$.

Complexity. The data complexity is $D = 2^{15c+LG(g)+1}$. The time complexity of this whole attack is $2^{8c+15c+LG(g)+1} \cdot \frac{8}{21 \cdot 16} + 2^{25c+17c+LG(g)-3.81} + 2^{48c-g}$. For SKINNY-64-192 with $c = 4$, we set $g = 23$, then $D = 2^{64.99}$, $T = 2^{169.38}$, $M = 2^{103.99}$. For SKINNY-128-384 with $c = 8$, we set $g = 46$, then $D = 2^{125.99}$, $T = 2^{338.65}$, $M = 2^{204.99}$.

I Application to Midori64

I.1 Specification

Midori is a lightweight block cipher proposed by Banik *et al.* at ASIACRYPT 2015 [50]. Midori-family includes two ciphers: Midori64 with 64-bit block size (4-bit cell size) and 128-bit key size, Midori128 with 128-bit block size (8-bit cell size) and 128-bit key size.

Midori is a variant of substitution-permutation network (SPN) and the number of rounds is 16 for Midori64. The round function of Midori consists of four transformations: `SubCell` (`SB`), `ShuffleCell` (`SC`), `MixColumn` (`MC`) and `KeyAdd` (`AK`). In `SB` operation, a 4-bit S-box is applied to every 4-bit cell of the internal state S of Midori64. In `SC` operation, each 4-bit cell of the state S is permuted as follows:

$$(s_0, s_1, \ldots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8).$$

The `MC` operation applies an involutory matrix

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

to every column of the state. In `AK` operation, the n-bit round key is XORed to the state S. The round function of Midori is shown in Figure 16.

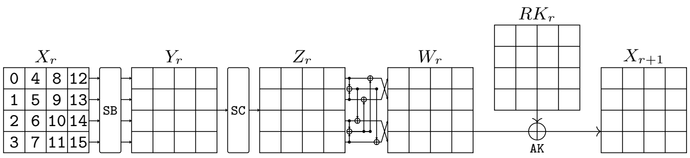


Fig. 16: Round function of Midori

Round Key Generation. For Midori64, a 128-bit secret key K is denoted as two 64-bit keys K_0 and K_1 as $K = K_0 \| K_1$. Then, $WK = K_0 \oplus K_1$ and $RK_i = K_{(i \bmod 2)} \oplus \alpha_i$, where $0 \leq i \leq 14$. The WK is used as the whitening key between the plaintext and round 0, and as the round key in the final round (where there is no `SC` and `MC` operations).

I.2 Single-Key Impossible Differential Attack on Midori64

This section provides a 11-round single-key impossible differential attack against Midori64. In this attack, we use the same 6-round single-key impossible differential as in [25]:

$$\Delta W_1 = (0000|0000|00?0|0000) \nrightarrow \Delta Z_7 = (0?00|0000|0000|0?00).$$

We prefix 2 rounds at the beginning and append 3 rounds at the end of the distinguisher to mount the attack, as shown in Figure 17. From the figure, we can get the parameters used for this attack: $r_B = 9c$, $c_B = 8c$, $r_F = 14c$, $c_F = 13c$, $|k_B \cup k_F| = 23c$, $c'_B = 6c$, $c'_F = 2c$, $|k'_B \cup k'_F| = 9c$. In the key-recovery extensions of this attack, it adopts deterministic extensions.

Pairs Collection. In this phase, we guess 2^{9c} possible values of $WK[0, 1, 4, 5, 6, 9, 10, 12, 14]$ to generate pairs. There will be $N = 2^{c^*_B + c^*_F + LG(g)} = 2^{13c+LG(g)}$ pairs need to be prepared.

Guess-and-Filter. For N pairs under each pre-guessed subkey bits:

1. *Satisfying the cell conditions in* ΔeW_9. With condition $\Delta eW_9[0] = \Delta X_{10}[1] \oplus \Delta X_{10}[2] = 0$, we can get the value of $\Delta X_{10}[2]$. By applying Lemma 2, we can deduce the value of $Y_{10}[2]$ and thus $WK[2]$. Similarly, we can derive $WK[7, 8, 13, 15]$. Time complexity of this step would be approximated by $2^{9c} \cdot N \cdot 2 \cdot \frac{5}{11 \cdot 16} = N 2^{9c-4.14}$.
2. *Satisfying the cell conditions in* ΔeW_8. Guess $eRK_9[5, 11]$. The condition $\Delta X_9[4] = \Delta X_9[6]$ will lead to one c-bit filter. From the condition $\Delta X_9[4] = \Delta X_9[7]$, we can determine the value of $eRK_9[12]$. Guess $eRK_9[4, 13]$. The condition $\Delta X_9[13] = \Delta X_9[14]$ will lead to one c-bit filter. From the condition $\Delta X_9[13] = \Delta X_9[15]$, we can determine the value of $eRK_9[3]$. Time complexity of this step is $2^{11c} \cdot N \cdot 2 \cdot \frac{2}{11 \cdot 16} + 2^{13c} \cdot N \cdot 2^{-c} \cdot 2 \cdot \frac{2}{11 \cdot 16} = N 2^{12c-5.46}$.
3. *Satisfying the cell conditions in* ΔZ_7. Due to $eRK_9[5] = K_1[4] \oplus K_1[6] \oplus K_1[7]$, $WK[4] = K_0[4] \oplus K_1[4]$,$WK[6] = K_0[6] \oplus K_1[6]$ and $WK[7] = K_0[7] \oplus K_1[7]$, we can compute $eRK_8[5] = K_0[4] \oplus K_0[6] \oplus K_0[7]$. Similarly, we can compute $eRK_8[12]$. The condition $\Delta X_8[4] = \Delta X_8[7]$ will lead to one c-bit filter. Time complexity of this step is $2^{13c} \cdot N \cdot 2^{-2c} \cdot 2 \cdot \frac{2}{11 \cdot 16} = N 2^{11c-5.46}$.
4. *Satisfying the cell conditions in* ΔW_1. Guess $RK_0[3]$. We can derive $RK_0[6, 9]$ by applying Lemma 2 with the condition $\Delta Y_1[3] = \Delta Y_1[6] = \Delta Y_1[9]$. Time complexity of this step is $2^{14c} \cdot N \cdot 2^{-3c} \cdot 2 \cdot \frac{3}{11 \cdot 16} = N 2^{11c-4.87}$.

Complexity. The data complexity is $D = 2^{14c+LG(g)+1}$. The time complexity is $2^{9c+14c+LG(g)+1} \cdot \frac{9}{11 \cdot 16} + 2^{25c+LG(g)-5.46} + 2^{32c-g}$. We set $g = 29$, then $D = 2^{61.33}$, $T = 2^{99.94}$, $M = 2^{56.33}$.

Fig. 17: The single-key impossible differential attack against 11-round Midori64

J Applications to Deoxys-BC

J.1 Specification

Deoxys-BC [51], the core primitive of authenticated encryption scheme Deoxys (winner of the CAESAR competition), is an 128-bit tweakable block cipher conforming to the TWEAKEY framework [52]. Deoxys-BC has two main versions: Deoxys-BC-256 with 256-bit tweakey size and Deoxys-BC-384 with 384-bit tweakey size.

Deoxys-BC takes an AES-like design and adopts a SPN structure that transforms the internal states through a round function similar to that of AES. Deoxys-BC-256 has 14 rounds, while Deoxys-BC-384 has 16 rounds.

The round function of Deoxys-BC consists of the four transformations in the order specified below:

- AddRoundTweakey (ART): XOR the 128-bit round subtweakey to the internal state.

- SubBytes (SB): Apply the 8-bit AES S-box $\mathcal{S}$ to the 16 bytes of the internal state.
- ShiftRows (SR): Rotate the 4-byte i-th row left by i positions, $i = 0, 1, 2, 3$.
- MixColumns (MC): Multiply the internal state by the 4×4 MDS matrix of AES.

At the end of the last round, a final AddRoundTweakey operation is applied to the internal state to produce the ciphertext. Figure 18 provides an overview of the round function of Deoxys-BC.

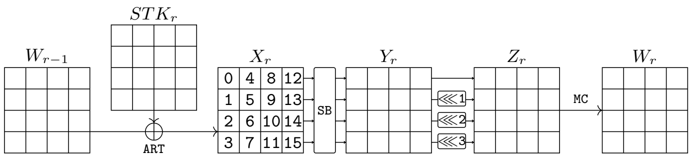


Fig. 18: Round function of Deoxys-BC

Tweakey Schedule. Different from the key schedule of AES, Deoxys-BC used a linear tweakey schedule under the TWEAKEY framework. We denote the concatenation of the key K and the tweak K as KT, i.e. $KT = K||T$. For Deoxys-BC-256, the size of KT is 256 bits with the first (most significant) 128 bits denoted as W_1, the second W_2, while the 384 bits tweakey of Deoxys-BC-384 is divided into W_1, W_2 and W_3 per 128 bits sequentially. For Deoxys-BC-256, a subtweakey of i-th round is defined as $STK_i = TK_i^1 \oplus TK_i^2 \oplus RC_i$ while for the case of Deoxys-BC-384 it is defined as $STK_i = TK_i^1 \oplus TK_i^2 \oplus TK_i^3 \oplus RC_i$.

The 128-bit words TK_i^1, TK_i^2, TK_i^3 are outputs produced by tweakey schedule algorithm, initialized with $TK_0^1 = W_1$ and $TK_0^2 = W_2$ for Deoxys-BC-256 and with $TK_0^1 = W_1, TK_0^2 = W_2$ and $TK_0^3 = W_3$ for Deoxys-BC-384. The tweakey schedule algorithm is defined as

$$TK_{i+1}^1 = h(TK_i^1), TK_{i+1}^2 = h(LFSR_2(TK_i^2)), TK_{i+1}^3 = h(LFSR_3(TK_i^3)),$$

where the byte permutation h is defined as:

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

The $LFSR_2$ and $LFSR_3$ functions are the application of an LFSR to each of the 16 bytes of a tweakey 128-bit word. The two LFSRs used are given in Table 6.

Table 6: Two LFSRs used in Deoxys-BC tweakey schedule

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

For more details on the specification of Deoxys-BC, please refer to [51].

J.2 Related-Tweakey Impossible Boomerang Attack on Deoxys-BC-256

In this section, we propose a new 10-round related-tweakey impossible boomerang attack against Deoxys-BC-256. In this attack, we use a 7-round related-tweakey impossible boomerang distinguisher, prefix one round at the beginning and append 2 rounds at the end of the distinguisher to mount the attack, as shown in Figure 19. From the figure, we can get the parameters used for this attack: $r_B = 2c$, $c_B = 2c$, $r_F = 9c$, $c_F = 9c$, $|k_B \cup k_F| = 13c$, $c'_B = 2c$, $c'_F = 3c$, $|k'_B \cup k'_F| = 6c$. In the key-recovery extensions of this attack, it adopts deterministic extensions.

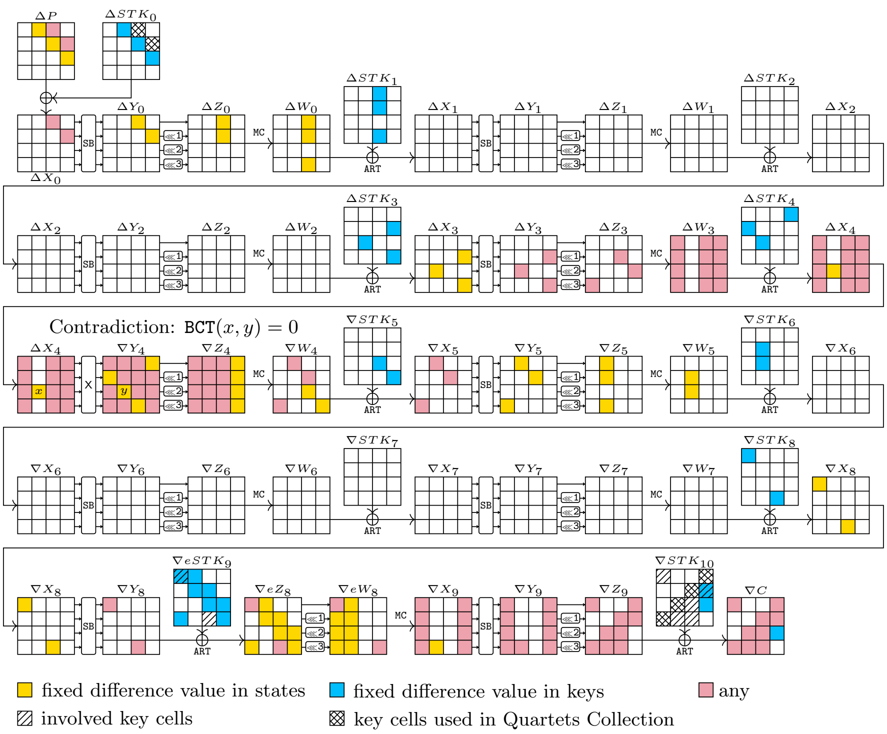


Fig. 19: The related-tweakey impossible boomerang attack against 10-round Deoxys-BC-256

Quartets Collection. In this phase, we guess 2^{6c} possible values of $STK_{10}[3,6,9,12]$ and $STK_0[8,13]$ to generate quartets. There will be $Q = 2^{2c_B^* + 2c_F^* + LG(g)} = 2^{12c+LG(g)}$ quartets need to be prepared.

Guess-and-Filter. For Q quartets under each pre-guessed subkey bits:

1. *Satisfying the cell conditions in* ∇X_9. Guess $STK_{10}[11]$. The condition with known $\nabla X_9[7]$ on both sides of the boomerang will lead to two c-bit filters. Time complexity of this step is $2^{7c} \cdot Q \cdot 4 \cdot \frac{1}{10 \cdot 16} = Q2^{7c-5.32}$.
2. *Satisfying the cell conditions in* ∇X_8. Guess $eSTK_9[11]$ and compute $\nabla X_8[11]$. The condition with known $\nabla X_8[11]$ will lead to two c-bit filters. Time complexity of this step is $2^{8c} \cdot Q \cdot 2^{-2c} \cdot 4 \cdot \frac{1}{10 \cdot 16} = Q2^{6c-5.32}$.
3. *Satisfying the cell conditions in* ∇X_8. Guess $STK_{10}[0, 7, 10, 13]$. The condition $\Delta eW_8[12-14] = 0$ will lead to six c-bit filter. Guess $eSTK_9[0]$. The condition with known $\nabla X_8[0]$ on both sides of the boomerang will lead to two c-bit filters. Time complexity of this step is $2^{12c} \cdot Q \cdot 2^{-4c} \cdot 4 \cdot \frac{4}{10 \cdot 16} = Q2^{8c-3.32}$.

Complexity. The data complexity is $D = 2^{16c+\frac{LG(g)}{2}+2}$. The time complexity is $2^{6c+16c+\frac{LG(g)}{2}+2} \cdot \frac{6}{10 \cdot 16} + 2^{6c+16c+\frac{LG(g)}{2}+1} \cdot \frac{6}{10 \cdot 16} + 2^{20c+LG(g)-3.32} + 2^{32c-g}$. We set $g = 80$, then $D = 2^{132.9}$, $T = 2^{177.42}$, $M = 2^{101.79}$.

J.3 Related-Tweakey Impossible Boomerang Attack on Deoxys-BC-384

In this section, we propose a new 14-round related-tweakey impossible boomerang attack against Deoxys-BC-384. In this attack, we use a 9-round related-tweakey impossible boomerang distinguisher, prefix 3 rounds at the beginning and append 2 rounds at the end of the distinguisher to mount the attack, as shown in Figure 20. From the figure, we can get the parameters used for this attack: $r_B = 16c$, $c_B = 16c$, $r_F = 6c$, $c_F = 6c$, $|k_B \cup k_F| = 35c$, $c'_B = 8c$, $c'_F = 6c$, $|k'_B \cup k'_F| = 26c$. In the key-recovery extensions of this attack, it adopts deterministic extensions.

Quartets Collection. In this phase, we guess 2^{26c} possible values of $STK_0[0-15]$, $STK_1[1-2]$, $STK_{14}[0, 1, 7, 10, 13, 14]$ and $eSTK_{13}[11, 15]$ to generate quartets. There will be $Q = 2^{2c_B^*+2c_F^*+LG(g)} = 2^{16c+LG(g)}$ quartets need to be prepared.

Guess-and-Filter. For Q quartets under each pre-guessed subkey bits:

1. *Satisfying the cell conditions in* ΔY_1. Guess $STK_1[7]$ and compute $\Delta Y_1[7]$. The condition with known $\Delta Y_1[7]$ will lead to two c-bit filters. Similarly, the conditions with known $\Delta Y_1[10, 11, 15]$ will lead to six c-bit filters by guessing $STK_1[10, 11, 15]$. Time complexity of this step would be approximated by $2^{27c} \cdot Q \cdot 4 \cdot \frac{1}{14 \cdot 16} = Q2^{27c-5.81}$.
2. *Satisfying the cell conditions in* ΔW_1. Guess $STK_1[3, 4, 9, 14]$. The condition with known $\Delta W_1[4]$ on both sides of the boomerang will lead to two c-bit filters. Time complexity of this step is $2^{34c} \cdot Q \cdot 2^{-8c} \cdot 4 \cdot \frac{4}{14 \cdot 16} = Q2^{26c-3.81}$.
3. *Satisfying the cell conditions in* ΔY_2. We can determine $STK_2[5, 6]$ from known $STK_0[3, 8]$, $STK_1[10, 15]$ and $STK_{14}[13, 14]$ by applying the tweakey schedule of Deoxys-BC-384. Guess $STK_2[7]$. The conditions $\Delta Y_2[5-7] = 0$ will lead to six c-bit filter. Time complexity of this step would be a negligible one compared to previous steps.

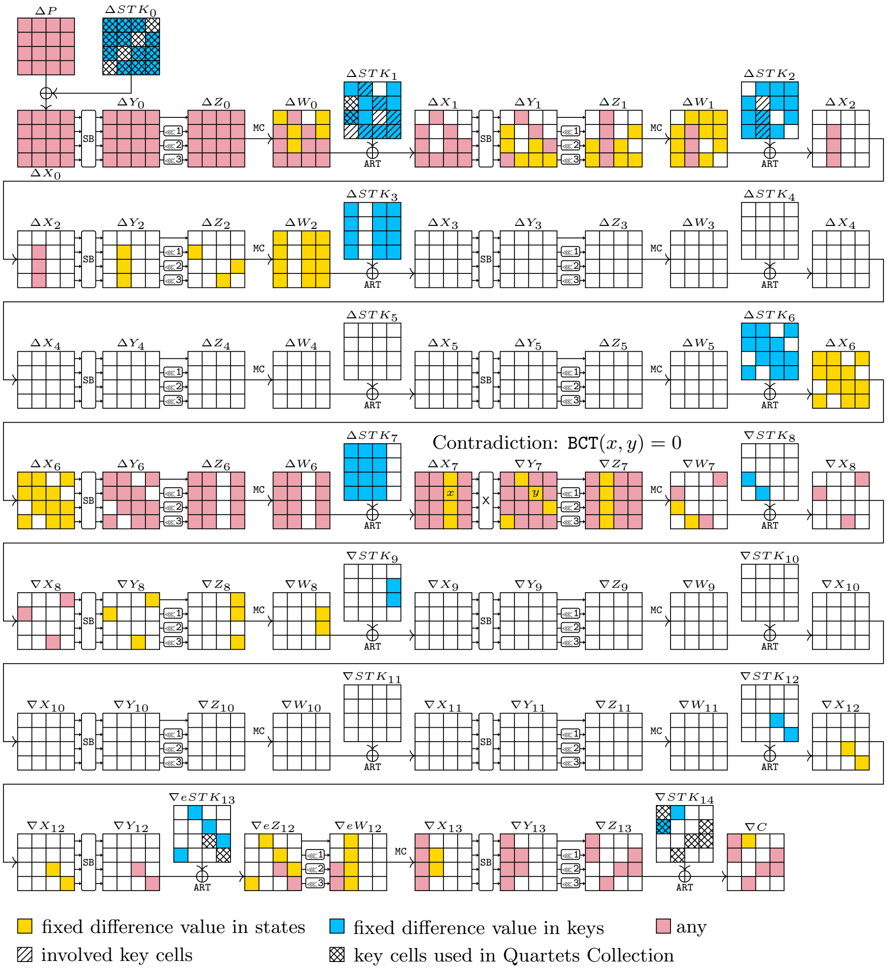


Fig. 20: The related-tweakey impossible boomerang attack against 14-round Deoxys-BC-384

Complexity. The data complexity is $D = 2^{16c+\frac{LG(g)}{2}+2}$. The time complexity of the whole attack is $2^{26c+16c+\frac{LG(g)}{2}+2} \cdot \frac{26}{14 \cdot 16} + 2^{26c+16c+LG(g)+1} \cdot \frac{26}{14 \cdot 16} + 2^{27c+16c+LG(g)-5.81} + 2^{48c-g}$. We set $g = 41$, then $D = 2^{132.41}$, $T = 2^{343.05}$, $M = 2^{132.83}$.