Finding the Impossible: Automated Search for Full Impossible-Differential, Zero-Correlation, and Integral Attacks

Hosein Hadipour[1](✉), Sadegh Sadeghi[2], and Maria Eichlseder[1]

[1] Graz University of Technology, Graz, Austria
hsn.hadipour@gmail.com, maria.eichlseder@iaik.tugraz.at
[2] Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran

Abstract. Impossible differential (ID), zero-correlation (ZC), and integral attacks are a family of important attacks on block ciphers. For example, the impossible differential attack was the first cryptanalytic attack on 7 rounds of AES. Evaluating the security of block ciphers against these attacks is very important but also challenging: Finding these attacks usually implies a combinatorial optimization problem involving many parameters and constraints that is very hard to solve using manual approaches. Automated solvers, such as Constraint Programming (CP) solvers, can help the cryptanalyst to find suitable attacks. However, previous CP-based methods focus on finding only the ID, ZC, and integral distinguishers, often only in a limited search space. Notably, none can be extended to a unified optimization problem for finding full attacks, including efficient key-recovery steps.
In this paper, we present a new CP-based method to search for ID, ZC, and integral distinguishers and extend it to a unified constraint optimization problem for finding full ID, ZC, and integral attacks. To show the effectiveness and usefulness of our method, we applied it to several block ciphers, including SKINNY, CRAFT, SKINNYe-v2, and SKINNYee. For the ISO standard block cipher SKINNY, we significantly improve all existing ID, ZC, and integral attacks. In particular, we improve the integral attacks on SKINNY-n-$3n$ and SKINNY-n-$2n$ by 3 and 2 rounds, respectively, obtaining the best cryptanalytic results on these variants in the single-key setting. We improve the ZC attack on SKINNY-n-n (SKINNY-n-$2n$) by 2 (resp. 1) rounds. We also improve the ID attacks on all variants of SKINNY. Particularly, we improve the time complexity of the best previous single-tweakey (related-tweakey) ID attack on SKINNY-128-256 (resp. SKINNY-128-384) by a factor of $2^{22.57}$ (resp. $2^{15.39}$). On CRAFT, we propose a 21-round (20-round) ID (resp. ZC) attack, which improves the best previous single-tweakey attack by 2 (resp. 1) rounds. Using our new model, we also provide several practical integral distinguishers for reduced-round SKINNY, CRAFT, and Deoxys-BC. Our method is generic and applicable to other strongly aligned block ciphers.

Keywords: Impossible differential attacks · Zero-correlation attacks · Integral attacks · SKINNY · SKINNYe · CRAFT · SKINNYee · Deoxys-BC

1 Introduction

The impossible differential (ID) attack, independently introduced by Biham et al. [5] and Knudsen [24], is one of the most important attacks on block ciphers. For example, the ID attack is the first attack breaking 7 rounds of AES-128 [28]. The ID attack exploits an impossible differential in a block cipher, which usually originates from slow diffusion, to retrieve the master key. The zero-correlation (ZC) attack, first introduced by Bogdanov and Rijmen [8], is the dual method of the ID attack in the context of linear analysis, which exploits an unbiased linear approximation to retrieve the master key.

The integral attack is another important attack on block ciphers which was first introduced as a theoretical generalization of differential analysis by Lai [25] and as a practical attack by Daemen et al. [13]. The core idea of integral attacks is finding a set of inputs such that the sum of the resulting outputs is key-independent in some positions. At ASIACRYPT 2012, Bogdanov et al. established a link between the (multidimensional) ZC approximation and integral distinguishers [7]. Sun et al. at CRYPTO 2015 [41] developed further the links among the ID, ZC, and integral attacks. Thanks to this link, we can use search techniques for ZC distinguishers to find integral distinguishers. Ankele et al. studied the influence of the tweakey schedule in ZC analysis of tweakable block ciphers at ToSC 2019 [1] and showed that taking the tweakey schedule into account can result in a longer ZC distinguisher.

The search for ID, ZC, and integral attacks on a block cipher contains two main phases: finding a distinguisher and mounting a key recovery based on the discovered distinguisher. One of the main techniques to find ID and ZC distinguishers is the miss-in-the-middle technique [5, 7]. The idea is to find two differences (linear masks) that propagate halfway through the cipher forward and backward with certainty but contradict each other in the middle. However, applying this technique requires tracing the propagation of differences (resp. linear masks) at the word- or bit-level of block ciphers, which is a time-consuming and potentially error-prone process using a manual approach. When it comes to the key recovery, we should extend the distinguisher at both sides and trace the propagation of more cryptographic properties taking many critical parameters into account. In general, finding an optimum complete ID, ZC, or integral attack usually implies a combinatorial optimization problem which is very hard to solve using a manual approach, especially when the block size is large and there are many possible solutions. Therefore, developing automatic tools is important to evaluate the security of block ciphers against these attacks, mainly, in designing and analyzing lightweight cryptographic primitives, where a higher precision in security analysis lets us minimize security margins.

One approach to solving the optimization problems stemming from cryptanalytic attacks is developing dedicated algorithms. For instance, in CRYPTO 2016, Derbez and Fouque proposed a dedicated algorithm [14] to find $\mathcal{DS}$-MITM and ID attacks. However, developing and implementing efficient algorithms is difficult and implies a hard programming task. In addition, other researchers may

want to adapt these algorithms to other problems with some common features and some differences. This may, again, be very difficult and time-consuming.

Another approach is converting the cryptanalytic problem into a constraint satisfaction problem (CSP) or a constraint optimization problem (COP) and then solving it with off-the-shelf constraint programming (CP) solvers. Recently, many CP-based approaches have been introduced to solve challenging symmetric cryptanalysis problems, which outperform the previous manual or dedicated methods in terms of accuracy and efficiency [20, 30, 37, 39, 46]. For example, at EUROCRYPT 2017, Sasaki and Todo proposed a new automatic tool based on mixed integer linear programming (MILP) solvers to find ID distinguishers [37]. Cui et al. proposed a similar approach to find ID and ZC distinguishers [12]. Sun et al. recently proposed a new CP-based method to search for ID and ZC distinguishers at ToSC 2020 [42].

Although the automatic methods to search for ID, ZC, and integral attacks had significant advances over the past years, they still have some basic limitations:

- The CP models for finding ID/ZC distinguishers proposed in [12, 37, 43] rely on the unsatisfiability of the models where the input/output difference/mask is fixed. This is also the case in all existing CP models to search for integral distinguishers based on division property [15, 44] or monomial prediction [19, 22]. However, finding an optimal key recovery attack is an optimization problem, which is based on satisfiability. Hence, the previous CP models for finding the ID, ZC, and integral distinguishers can not be extended to a unified optimization model for finding a complete attack. The previous CP models for finding ID, ZC, and integral distinguishers also require checking each input/output property individually. As a result, it is computationally hard to find all possible distinguishers when the block size is large enough.
- The CP model proposed in [42] employs the miss-in-the-middle technique to find ID/ZC distinguishers. This approach does not fix the input/output differences/masks. However, the compatibility between the two parts of the distinguisher is checked outside of the CP model by iterating over a loop where the activeness pattern of a state cell at the meeting point should be fixed in each iteration.
- All previous CP models regarding ID, ZC, and integral attacks only focus on finding the longest distinguishers. However, many other important factors affect the final complexity of these attacks, which we can not take into account by only modeling the distinguisher part. For example, the position and the number of active cells in the input/output of the distinguisher, the number of filters in verifying the desired properties at the input/output of distinguishers, and the number of involved key bits in the key recovery are only a few critical parameters that affect the final complexity of the attack but can be considered only by modeling the key recovery part. We show that the best attack does not necessarily require the longest distinguisher. Hence, it is important to unify the key recovery and distinguishing phases for finding better ID, ZC, and integral attacks.

– The tool introduced by Derbez and Fouque [14] is the only tool to find full ID attacks. However, this tool is based on a dedicated algorithm implemented in C/C++ and is not as generic as the CP-based methods. In addition, this tool can not take all critical parameters of ID attacks into account to minimize the final complexity. As other limitations, this tool can not find related-(twea)key ID attacks and is not applicable for ZC and integral attacks.
– None of the previous automatic tools takes the relationship between ZC and integral attacks into account to find ZC distinguishers suitable for integral key recovery. Particularly, there is no automatic tool to take the meet-in-the-middle technique into account for ZC-based integral attacks.

Our contributions. We propose a new generic, CP-based, and easy-to-use automatic method to find full ID, ZC, and integral attacks, addressing the above limitations. Unlike all previous CP models for these distinguishers, which are based on unsatisfiability, our CP model relies on satisfiability for finding distinguishers. This way, each solution of our CP models corresponds to an ID, ZC, or integral distinguisher. This key feature enables us to extend our distinguisher models to a unified model for finding an optimal key-recovery attack. Furthermore, our unified CP model takes advantage of key-bridging and meet-in-the-middle techniques. To show the usefulness of our method, we apply it to SKINNY [3], CRAFT [4], SKINNYe-v2 [31], and SKINNYee [32] and significantly improve the ZC, ID, and integral attacks on these ciphers. Table 1 summarizes our results.

– We improve the integral attacks on SKINNY-n-$2n$ and SKINNY-n-$3n$ by 2 and 3 rounds, respectively. To the best of our knowledge, our integral attacks are the best single-key attacks on these variants of SKINNY.
– We improve the ZC attacks on SKINNY-n-n (SKINNY-n-$2n$) by 2 (resp. 1) rounds. We also propose the first 21-round ZC attack on SKINNY-n-$3n$. Our ZC attacks are the best attacks on SKINNY in a known-plaintext setting.
– On CRAFT, we provide a 21-round (20-round) single-tweakey ID (resp. ZC) attack that is 2 (resp. 1) rounds longer than the best previous single-tweakey attack proposed on this cipher at ASIACRYPT 2022 [40].
– We improve all previous single-tweakey ID attacks on all variants of SKINNY. We reduce the time complexity of the ID attack on SKINNY-128-256 by a factor of $2^{22.57}$. Our ID attacks are the best single-tweakey attacks on SKINNY-128-128, and all variants of SKINNY-64. We also improved the related-tweakey ID attack on SKINNY-n-$3n$.
– We provide the first third-party analysis of SKINNYee by proposing 26-round integral and 27-round ID attacks.
– We propose several practical integral distinguishers for reduced round of Deoxys-BC, SKINNY, CRAFT, and SKINNYe-v2/ee (see Table 3).
– Our tool identified several flaws in previous cryptanalytic results on SKINNY (see Table 2). Our tool is efficient and can find all reported results in a few seconds when running on a regular laptop. Its source code is publicly available at the following link: https://github.com/hadipourh/zero

Table 1: Summary of our cryptanalytic results. ID/ZC/Int = impossible differential, zero-correlation, integral. STK/RTK = single/related-tweakey. SK = single-key with given keysize, CP/KP = chosen/known plaintext, CT = chosen tweak. †: attack has minor issues.

Cipher	#R	Time	Data	Mem.	Attack	Setting / Model	Ref.
SKINNY-64-192	**21**	$2^{185.83}$	$2^{62.63}$	2^{49}	ZC	STK / KP	G.3
	21	$2^{180.50}$	2^{62}	2^{170}	ID	STK / CP	[47]
	21	$\mathbf{2^{174.42}}$	$2^{62.43}$	2^{168}	ID	STK / CP	F.3
	23†	$2^{155.60}$	$2^{73.20}$	2^{138}	Int †	180,SK / CP,CT	[1]
	26	2^{172}	2^{61}	2^{172}	Int	180,SK / CP,CT	H.2
	27	2^{189}	$2^{63.53}$	2^{184}	ID	RTK / CP	[27]
	27	$\mathbf{2^{183.26}}$	$2^{63.64}$	2^{172}	ID	RTK / CP	F.4
SKINNY-128-384	**21**	$2^{372.82}$	$2^{122.81}$	2^{98}	ZC	STK / KP	G.3
	21	$2^{353.60}$	2^{123}	2^{341}	ID	STK / CP	[47]
	21	$\mathbf{2^{347.35}}$	$2^{122.89}$	2^{336}	ID	STK / CP	F.3
	26	2^{344}	2^{121}	2^{340}	Int	360,SK / CP,CT	H.2
	27	2^{378}	$2^{126.03}$	2^{368}	ID	RTK / CP	[27]
	27	$\mathbf{2^{362.61}}$	$2^{124.99}$	2^{344}	ID	RTK / CP	F.4
SKINNY-64-128	18	2^{126}	$2^{62.68}$	2^{64}	ZC	STK / KP	[36]
	19	$2^{119.12}$	$2^{62.89}$	2^{49}	ZC	STK / KP	G.2
	19	$2^{119.80}$	2^{62}	2^{110}	ID	STK / CP	[47]
	19	$\mathbf{2^{110.34}}$	$2^{60.86}$	2^{104}	ID	STK / CP	F.2
	20†	$2^{97.50}$	$2^{68.40}$	2^{82}	Int †	120,SK / CP,CT	[1]
	22	2^{110}	$2^{57.58}$	2^{108}	Int	120,SK / CP,CT	H.1
SKINNY-128-256	**19**	$2^{240.07}$	$2^{122.90}$	2^{98}	ZC	STK / KP	G.2
	19	$2^{241.80}$	2^{123}	2^{221}	ID	STK / CP	[47]
	19	$\mathbf{2^{219.23}}$	$2^{117.86}$	2^{208}	ID	STK / CP	F.2
	22	2^{216}	$2^{113.58}$	2^{216}	Int	240,SK / CP,CT	H.1
SKINNY-64-64	14	2^{62}	$2^{62.58}$	2^{64}	ZC	STK / KP	[36]
	16	$2^{62.71}$	$2^{61.35}$	$2^{37.80}$	ZC	STK / KP	G.1
	17	$2^{61.80}$	$2^{59.50}$	$2^{49.60}$	ID	STK / CP	[47]
	17	$\mathbf{2^{59}}$	$2^{58.79}$	2^{40}	ID	STK / CP	F.1
SKINNY-128-128	**16**	$2^{122.79}$	$2^{122.30}$	$2^{74.80}$	ZC	STK / KP	G.1
	17	$2^{120.80}$	$2^{118.50}$	$2^{97.50}$	ID	STK / CP	[47]
	17	$\mathbf{2^{116.51}}$	$2^{116.37}$	2^{80}	ID	STK / CP	F.1
CRAFT	**20**	$2^{120.43}$	$2^{62.89}$	2^{49}	ZC	STK / KP	K.2
	21	$2^{106.53}$	$2^{60.99}$	2^{100}	ID	STK / CP	K.3
SKINNYee	**26**	2^{113}	2^{66}	2^{108}	Int	SK / CP,CT	I.3
	27	$2^{123.04}$	$2^{62.79}$	2^{108}	ID	RTK / CP	I.2
SKINNYe-v2	**30**	2^{232}	2^{65}	2^{228}	Int	240,SK / CP,CT	H.3

Outline. We recall the background on ID and ZC attacks and review the link between ZC and integral attacks in Section 2. In Section 3, we show how to convert the problem of searching for ID and ZC distinguishers to a CSP problem. In Section 4, we show how to extend our distinguisher models to create a unified model for finding optimum ID attacks. We discuss the extension of our models for ZC and integral attacks in Section 5, and finally conclude in Section 6.

Table 2: Attacks with a serious flaw (invalid attacks).

Cipher	Attack	#R	Setting / Model	Ref.	Flaw
SKINNY-n-n	ID	18	STK / CP	[45]	Section 4.2
SKINNY-n-$2n$	ID	20	STK / CP	[45]	Section 4.2
	ZC/Int †	22	SK / CP, CT	[48]	Section 3
SKINNY-n-$3n$	ID	22	STK / CP	[45]	Section 4.2
	ZC/Int †	26	SK / CP, CT	[48]	Section 3

† [48] was published after publishing the first version of our paper.

2 Background

Here, we recall the basics of ID and ZC attacks and briefly review the link between the ZC and integral attacks. We also introduce the notations we use in the rest of this paper. We refer to the appendix for the specification of SKINNY and SKINNYe (Section C), CRAFT (Section K.1), and SKINNYee (Section I.1).

2.1 Impossible Differential Attack

The impossible differential attack was independently introduced by Biham et al. [5] and Knudsen [24]. The core idea of an impossible differential attack is exploiting an impossible differential in a cipher to retrieve the key by discarding all key candidates leading to such an impossible differential. The first requirement of the ID attack is an ID distinguisher, i.e., an input difference that can never propagate to a particular output difference. Then, we extend the ID distinguisher by some rounds backward and forward. A candidate for the key that partially encrypts/decrypts a given pair to the impossible differential is certainly not valid. The goal is to discard as many wrong keys as possible. Lastly, we uniquely retrieve the key by exhaustively searching the remaining candidates.

We recall the complexity analysis of the ID attack based on [10, 11]. Let E be a block cipher with n-bit block size and k-bit key. As illustrated in Figure 1, assume that there is an impossible differential $\Delta_{\mathrm{U}} \nrightarrow \Delta_{\mathrm{L}}$ for r_{D} rounds of E denoted by E_{D}. Suppose that Δ_{U} (Δ_{L}) propagates backward (resp. forward) with probability 1 through E_{B}^{-1} (resp. E_{F}) to Δ_{B} (Δ_{F}), and $|\Delta_{\mathrm{B}}|$ ($|\Delta_{\mathrm{F}}|$) denotes the dimension of vector space Δ_{B} (resp. Δ_{F}). Let c_{B} (c_{F}) be the number of

bit-conditions that should be satisfied for $\Delta_{\mathrm{B}} \to \Delta_{\mathrm{U}}$ (resp. $\Delta_{\mathrm{L}} \leftarrow \Delta_{\mathrm{F}}$), i.e., $\Pr(\Delta_{\mathrm{B}} \to \Delta_{\mathrm{U}}) = 2^{-c_{\mathrm{B}}}$ (resp. $\Pr(\Delta_{\mathrm{L}} \leftarrow \Delta_{\mathrm{F}}) = 2^{-c_{\mathrm{F}}}$). Moreover, assume that k_{B} (k_{F}) denotes the key information, typically subkey bits, involved in E_{B} (resp. E_{F}). With these assumptions we can divide the ID attacks into three steps:

- *Step 1: Pair Generation.* Given access to the encryption oracle (and possibly the decryption oracle), we generate N pairs $(x, y) \in \{0,1\}^{2n}$ such that $x \oplus y \in \Delta_{\mathrm{B}}$ and $E(x) \oplus E(y) \in \Delta_{\mathrm{F}}$ and store them. This is a limited birthday problem, and according to [11] the complexity of this step is:

$$T_0 = \max\left\{ \min_{\Delta \in \{\Delta_{\mathrm{B}}, \Delta_{\mathrm{F}}\}} \left\{ \sqrt{N 2^{n+1-|\Delta|}} \right\}, N 2^{n+1-|\Delta_{\mathrm{B}}|-|\Delta_{\mathrm{F}}|} \right\} \tag{1}$$

- *Step 2: Guess-and-Filter.* The goal of this step is to discard all subkeys in $k_{\mathrm{B}} \cup k_{\mathrm{F}}$ which are invalidated by at least one of the generated pairs. Rather than guessing all subkeys $k_{\mathrm{B}} \cup k_{\mathrm{F}}$ at once and testing them with all pairs, we can optimize this step by using the *early abort* technique [29]: We divide $k_{\mathrm{B}} \cup k_{\mathrm{F}}$ into smaller subsets, typically the round keys, and guess them step by step. At each step, we reduce the remaining pairs by checking if they satisfy the conditions of the truncated differential trail through E_{B} and E_{F}. The minimum number of partial encryptions/decryptions in this step is [10]:

$$T_1 + T_2 = N + 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|} \frac{N}{2^{c_{\mathrm{B}}+c_{\mathrm{F}}}} \tag{2}$$

- *Step 3: Exhaustive Search.* The probability that a wrong key survives through the guess-and-filter step is $P = \left(1 - 2^{-(c_{\mathrm{B}}+c_{\mathrm{F}})}\right)^N$. Therefore, the number of candidates after performing the guess-and-filter is $P \cdot 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|}$ on average. On the other hand, the guess-and-filter step does not involve $k - |k_{\mathrm{B}} \cup k_{\mathrm{F}}|$ bits of key information. As a result, to uniquely determine the key, we should exhaustively search a space of size $T_3 = 2^{k-|k_{\mathrm{B}} \cup k_{\mathrm{F}}|} \cdot P \cdot 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|} = 2^k \cdot P$.

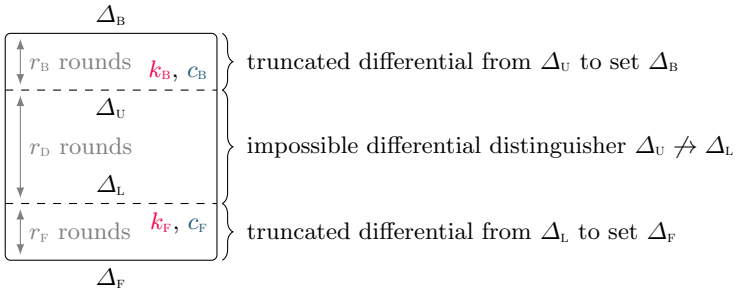

Fig. 1: Main parameters of the ID attack using an r_{D}-round impossible differential distinguisher $\Delta_{\mathrm{U}} \not\to \Delta_{\mathrm{L}}$. The distinguisher is extended with truncated differential propagation to sets $\Delta_{\mathrm{U}} \to \Delta_{\mathrm{B}}$ over r_{B} rounds backwards and $\Delta_{\mathrm{L}} \to \Delta_{\mathrm{F}}$ over r_{F} rounds forward. The inverse differentials $\Delta_{\mathrm{B}} \to \Delta_{\mathrm{U}}$ and $\Delta_{\mathrm{F}} \to \Delta_{\mathrm{L}}$ involve $k_{\mathrm{B}}, k_{\mathrm{F}}$ key bits and have weight $c_{\mathrm{B}}, c_{\mathrm{F}}$, respectively.

Then, the total time complexity of the ID attack is:

$$T_{tot} = (T_0 + (T_1 + T_2)\, C_{E'} + T_3)\, C_E, \tag{3}$$

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

To keep the data complexity less than the full codebook, we require $T_0 < 2^n$. In addition, to retrieve at least one bit of key information in the guess-and-filter step, $P < \frac{1}{2}$ should hold. Note that Equation 2 is the average time complexity of the guess-and-filter step; for each ID attack, we must evaluate its complexity accurately to ensure we meet this bound in practice. To see the complexity analysis of the ID attack in the related-(twea)key setting, refer to Appendix A.

2.2 Multidimensional Zero-Correlation Attack

Zero-correlation attacks, firstly introduced by Bogdanov and Rijmen [8], are the dual of the ID attack in the context of linear analysis and exploit a linear approximation with zero correlation. The major limitation of the basic ZC attack is its enormous data complexity, equal to the full codebook. To reduce the data complexity of the ZC attack, Bogdanov and Wang proposed the multiple ZC attack at FSE 2012 [9], which utilizes multiple ZC linear approximations. However, the multiple ZC attack relies on the assumption that all involved ZC approximations are independent, which limits its applications. To overcome this assumption, Bogdanov et al. introduced the multidimensional ZC attack at ASIACRYPT 2012 [7]. We briefly recall the basics of a multidimensional ZC attack.

Let E_{D} represents the reduced-round block cipher E with a block size of n bits. Assume that the correlation of m independent linear approximations $\langle u_i, x\rangle + \langle w_i, E_{\mathrm{D}}(x)\rangle$ and all their nonzero linear combinations are zero, where $u_i, w_i, x \in \mathbb{F}_2^n$, for $i = 0, \ldots, m-1$. We denote by $l = 2^m$ the number of ZC linear approximations. In addition, assume we are given N input/output pairs $(x, y = E_{\mathrm{D}}(x))$. Then, we can construct a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ which maps x to $z(x) = (z_0, \ldots, z_{m-1})$, where $z_i := \langle u_i, x\rangle + \langle w_i, E_{\mathrm{D}}(x)\rangle$ for all i. The idea of the multidimensional ZC distinguisher is that the output of this function follows the *multivariate hypergeometric distribution*, whereas the m-tuples of bits drawn at random from a uniform distribution on $\mathbb{F}_2^m$ follow a *multinomial distribution* [7]. For sufficiently large N, we distinguish E_{D} from a random permutation as follows.

We initialize 2^m counters $V[z]$ to zero, $z \in \mathbb{F}_2^m$. Then, for each of the N pairs (x, y), we compute $z_i = \langle u_i, x\rangle + \langle w_i, y\rangle$ for all $i = 0, \ldots, 2^m - 1$, and increment $V[z]$ where $z = (z_0, \ldots, z_{m-1})$. Finally, we compute the following statistic:

$$T = \frac{N \cdot 2^m}{1 - 2^{-m}} \sum_{z=0}^{2^m-1} \left(\frac{V[z]}{N} - \frac{1}{2^m} \right)^2. \tag{4}$$

For the pairs (x, y) derived from E_{D}, i.e., $y = E_{\mathrm{D}}(x)$, the statistic T follows a χ^2-distribution with mean $\mu_0 = (l-1)\frac{2^n - N}{2^n - 1}$ and variance $\sigma_0^2 = 2(l-1)(\frac{2^n - N}{2^n - 1})^2$. However, it follows a χ^2-distribution with mean $\mu_1 = (l-1)$ and variance $\sigma_1^2 =$

$2(l-1)$ for a random permutation [7]. By defining a decision threshold $\tau = \mu_0 + \sigma_0 Z_{1-\alpha} = \mu_1 - \sigma_1 Z_{1-\beta}$, the output of test is 'cipher', i.e., the pairs are derived from E_{D}, if $T \leq \tau$. Otherwise, the output of the test is 'random'.

This test may wrongfully classify E_{D} as a random permutation (type-I error) or may wrongfully accept a random permutation as E_{D} (type-II error). Let the probability of the type-I and type-II errors be α and β. Then, the number of required pairs N to successfully distinguish E_{D} from a random permutation is [7]:

$$N = \frac{2^n(Z_{1-\alpha} + Z_{1-\beta})}{\sqrt{l/2} - Z_{1-\beta}}, \tag{5}$$

where $Z_{1-\alpha}$, and $Z_{1-\beta}$ are respective quantiles of the standard normal distribution. Thus, the data complexity of the multidimensional ZC attack depends on the number of ZC approximations, $l = 2^m$, and the error probabilities α and β.

To mount a key recovery based on a multidimensional ZC distinguisher for E_{D}, we extend E_{D} by a few rounds at both ends, $E = E_{\mathrm{F}} \circ E_{\mathrm{D}} \circ E_{\mathrm{B}}$. Given N plaintext/ciphertext pairs $(p, c = E(p))$, we can recover the key in two steps:

- *Step 1: Guess-and-filter.* We guess the value of involved key bits in E_{B} (E_{F}) and partially encrypt (decrypt) the plaintexts (ciphertexts) to derive N pairs (x, y) for the input $x = E_{\mathrm{B}}(p)$ and output $y = E_{\mathrm{F}}^{-1}(c)$ of E_{D}. Assuming that wrong keys yield pairs (x, y) randomly chosen from $\mathbb{F}_2^{2n}$, we use the statistic T to discard all keys for which $T \leq \tau$.
- *Step 2: Exhaustive Search.* Finally, we exhaustively search the remaining key candidates to find the correct key.

The time complexity of the guess-and-filter step depends on the number of pairs N and the size of involved key bits in E_{B} and E_{F}. Given that typically a subset of internal variables is involved in the partial encryptions/decryptions, we can take advantage of the partial sum technique [16] to reduce the time complexity of the guess-and-filter step. Moreover, by adjusting the value of α and β, we can make a trade-off between the time and data complexities as α and β affect the data, and β influences the time complexity of the exhaustive search.

2.3 Relation Between the Zero-Correlation and Integral Attacks

Bogdanov et al. [7] showed that an integral distinguisher[3] always implies a ZC distinguisher, but its converse is true only if the input and output linear masks of the ZC distinguisher are independent. Later, Sun et al. [41] proposed the following theorem that the conditions for deriving an integral distinguisher from a ZC linear hull in [7] can be removed.

Theorem 1 (Sun et al. [41]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a vectorial Boolean function. Assume A is a subspace of $\mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n \setminus \{0\}$ such that (α, β) is a ZC approximation for any $\alpha \in A$. Then, for any $\lambda \in \mathbb{F}_2^n$, $\langle \beta, F(x+\lambda) \rangle$ is balanced over the set*

$$A^{\perp} = \{x \in \mathbb{F}_2^n \mid \forall\ \alpha \in A : \langle \alpha, x \rangle = 0\}.$$

[3] Under the definition that integral property is a balanced vectorial Boolean function

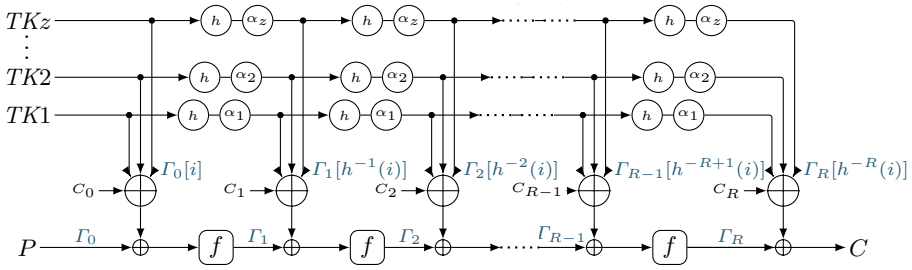

Fig. 2: The STK construction of the TWEAKEY framework.

According to Theorem 1, the data complexity of the resulting integral distinguisher is 2^{n-m}, where n is the block size and m is the dimension of the linear space spanned by the input linear masks in the corresponding ZC linear hull.

At ToSC 2019, Ankele et al. [1] considered the effect of the tweakey on ZC distinguishers of tweakable block ciphers (TBCs). They showed that taking the tweakey schedule into account can lead to a longer ZC distinguisher and thus a longer integral distinguisher. They proposed Theorem 2, which provides an algorithm to find ZC linear hulls for TBCs following the super-position tweakey (STK) construction of the TWEAKEY framework [23] (see Figure 2).

Theorem 2 (Ankele et al. [1]). *Let $E_K(T,P) : \mathbb{F}_2^{t\times n} \to \mathbb{F}_2^n$ be a TBC following the STK construction. Assume that the tweakey schedule of E_K has z parallel paths and applies a permutation h on the tweakey cells in each path. Let (Γ_0, Γ_r) be a pair of linear masks for r rounds of E_K, and $\Gamma_1, \ldots, \Gamma_{r-1}$ represents a possible sequence for the intermediate linear masks. If there is a cell position i such that any possible sequence $\Gamma_0[i], \Gamma_1[h^{-1}(i)], \Gamma_2[h^{-2}(i)], \ldots \Gamma_r[h^{-r}(i)]$ has at most z linearly active cells, then (Γ_0, Γ_r) yields a ZC linear hull for r rounds of E.*

Ankele et al. used Theorem 2 to manually find ZC linear hulls for several twekable block ciphers including SKINNY, QARMA [2], and MANTIS [3]. Later, Hadipour et al. [21] proposed a bitwise automatic method based on SAT to search for ZC linear hulls of tweakable block ciphers. This automatic method was then reused by Niu et al. [34] to revisit the ZC linear hulls of SKINNY-64-{128,192}.

2.4 Constraint Satisfaction and Constraint Optimization Problems

A constraint satisfaction problem (CSP) is a mathematical problem including a set of constraints over a set of variables that should be satisfied. More formally, a CSP is a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{X} = \{X_0, X_1, \ldots, X_{n-1}\}$ is a set of variables; $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{n-1}\}$ is the set of domains such that $X_i \in \mathcal{D}_i$, $0 \le i \le n-1$; and $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{n-1}\}$ is a set of constraints. Each constraint $\mathcal{C}_j \in \mathcal{C}$ is a tuple $(\mathcal{S}_j, \mathcal{R}_j)$, where $\mathcal{S}_j = \{X_{i_0}, \ldots, X_{i_{k-1}}\} \subseteq \mathcal{X}$ and $\mathcal{R}_j$ is a relation on the corresponding domains, i.e., $\mathcal{R}_j \subseteq \mathcal{D}_{i_0} \times \cdots \times \mathcal{D}_{i_{k-1}}$.

Any value assignment of the variables satisfying all constraints of a CSP problem is a feasible solution. The constraint optimization problem extends the

CSP problem by including an objective function to be minimized (or maximized). Searching for the solution of a CSP or COP problem is referred to as constraint programming (CP), and the solvers performing the search are called CP solvers.

In this paper, we use MiniZinc [33] to model and solve the CSP and COP problems over integer and real numbers. MiniZinc allows modeling the CSP and COP problems in a high-level and solver-independent way. It compiles the model into FlatZinc, a standard language supported by a wide range of CP solvers. For CSP/COP problems over integer numbers, we use Or-Tools [35], and for CSP/COP problems over real numbers, we employ Gurobi [17] as the solver.

2.5 Encoding Deterministic Truncated Trails

Here, we recall the method proposed in [42] to encode deterministic truncated differential trails. Thanks to the duality relation between differential and linear analysis, one can adjust this method for deterministic truncated linear trails; thus, we omit the details for the linear trails. We define two types of variables to encode the deterministic truncated differential trails. Assume that $\Delta X = (\Delta X[0], \ldots, \Delta X[m-1])$ represents the difference of the internal state X in an n-bit block cipher E, where $n = m \cdot c$, and $\Delta X[i] \in \mathbb{F}_2^c$ for all $i = 0, \ldots, m-1$. We use an integer variable $\texttt{AX}[i]$ to encode the activeness pattern of $\Delta X[i]$ and another integer variable $\texttt{DX}[i]$ to encode the actual c-bit difference value of $\Delta X[i]$:

$$\texttt{AX}[i] = \begin{cases} 0 & \Delta X[i] = 0 \\ 1 & \Delta X[i] \text{ is nonzero and fixed} \\ 2 & \Delta X[i] \text{ can be any nonzero value} \\ 3 & \Delta X[i] \text{ can take any value} \end{cases} \qquad \texttt{DX}[i] \in \begin{cases} \{0\} & \texttt{AX}[i] = 0 \\ \{1, \ldots, 2^c - 1\} & \texttt{AX}[i] = 1 \\ \{-1\} & \texttt{AX}[i] = 2 \\ \{-2\} & \texttt{AX}[i] = 3 \end{cases}$$

Then, we link $\texttt{AX}[i]$ and $\texttt{DX}[i]$ for all $i = 0, \ldots, m-1$ as follows:

$$\mathit{Link}(\texttt{AX}[i], \texttt{DX}[i]) := \begin{cases} \mathit{if}\ \texttt{AX}[i] = 0\ \mathit{then}\ \texttt{DX}[i] = 0 \\ \mathit{elseif}\ \texttt{AX}[i] = 1\ \mathit{then}\ \texttt{DX}[i] > 0 \\ \mathit{elseif}\ \texttt{AX}[i] = 2\ \mathit{then}\ \texttt{DX}[i] = -1 \\ \mathit{else}\ \texttt{DX}[i] = -2\ \mathit{endif} \end{cases}$$

MiniZinc supports conditional expression '*if-then-else-endif*', so we do not need to convert to integer inequalities. Next, we briefly explain the propagation rules of deterministic truncated differential trails.

Proposition 1 (Branching). *For $F : \mathbb{F}_2^c \to \mathbb{F}_2^{2c}$, $F(X) = (Y, Z)$ where $Z = Y = X$, the valid transitions for deterministic truncated differential trails satisfy*

$$\mathit{Branch}(\mathit{AX}, \mathit{DX}, \mathit{AY}, \mathit{DY}, \mathit{AZ}, \mathit{DZ}) := (\mathit{AZ} = \mathit{AX} \ \wedge\ \mathit{DZ} = \mathit{DX} \ \wedge\ \mathit{AY} = \mathit{AX} \ \wedge\ \mathit{DY} = \mathit{DX})$$

Proposition 2 (XOR). *For $F : \mathbb{F}_2^{2c} \to \mathbb{F}_2^c$, $F(X, Y) = Z$ where $Z = X \oplus Y$, the valid transitions for deterministic truncated differential trails satisfy*

$$\mathit{XOR}(\mathit{AX}, \mathit{DX}, \mathit{AY}, \mathit{DY}, \mathit{AZ}, \mathit{DZ}) := \begin{cases} \mathit{if}\ \mathit{AX} + \mathit{AY} > 2\ \mathit{then}\ \mathit{AZ} = 3 \wedge \mathit{DZ} = -2 \\ \mathit{elseif}\ \mathit{AX} + \mathit{AY} = 1\ \mathit{then}\ \mathit{AZ} = 1 \wedge \mathit{DZ} = \mathit{DX} + \mathit{DY} \\ \mathit{elseif}\ \mathit{AX} = \mathit{AY} = 0\ \mathit{then}\ \mathit{AZ} = 0 \wedge \mathit{DZ} = 0 \\ \mathit{elseif}\ \mathit{DX} + \mathit{DY} < 0\ \mathit{then}\ \mathit{AZ} = 2 \wedge \mathit{DZ} = -1 \\ \mathit{elseif}\ \mathit{DX} = \mathit{DY}\ \mathit{then}\ \mathit{AZ} = 0 \wedge \mathit{DZ} = 0 \\ \mathit{else}\ \mathit{AZ} = 1 \wedge \mathit{DZ} = \mathit{DX} \oplus \mathit{DY}\ \mathit{endif} \end{cases}$$

Proposition 3 (S-box). *Assume that $S : \mathbb{F}_2^c \to \mathbb{F}_2^c$ is a c-bit S-box and $Y = S(X)$. The valid transitions for deterministic truncated differential trails satisfy*

$$\mathit{S\text{-}box}(\mathit{AX}, \mathit{AY}) := (\mathit{AY} \neq 1\ \wedge\ \mathit{AX} + \mathit{AY} \in \{0, 3, 4, 6\}\ \wedge\ \mathit{AY} \geq \mathit{AX}\ \wedge\ \mathit{AY} - \mathit{AX} \leq 1)$$

For encoding the MDS matrices, see Appendix B. To encode non-MDS matrices, such as the matrix employed in SKINNY, as described in Appendix D, we can use the rules of XOR and branching to encode the propagation.

3 Modeling the Distinguishers

Although the key recovery of ZC and ID attacks are different, the construction of ZC and ID distinguishers relies on the same approach, which is the miss-in-the-middle technique [5,6]. The idea is to find two differences (linear masks) that propagate halfway through the cipher forward and backward with certainty but contradict each other in the middle. The incompatibility between these propagations results in an impossible differential (resp. unbiased linear hull).

Suppose we are looking for an ID or ZC distinguisher for E_{D}, which represents r_{D} rounds of a block cipher E. Moreover, we assume that the block size of E is n bits, where $n = m \cdot c$ with c being the cell size and m being the number of cells. We convert the miss-in-the-middle technique to a CSP problem to automatically find ID and ZC distinguishers. We first divide E_{D} into two parts, as illustrated in Figure 3: An upper part E_{U} covering r_{U} rounds and a lower part E_{L} of r_{L} rounds. Hereafter, we refer to the trails discovered for E_{U} (E_{L}) as the upper (lower) trail. We denote the internal state of E_{U} (E_{L}) after r rounds by XU_r (XL_r). The state $XU_{r_{\mathrm{U}}}$ (or XL_0) at the intersection of E_{U} and E_{L} is called the meeting point.

Let $\mathtt{AXU}_r$ and $\mathtt{AXL}_r$ denote the activeness pattern of the state variables XU_r and XL_r, as shown in Figure 3. Let $\mathtt{DXU}_r$ and $\mathtt{DXL}_r$ denote the actual difference values in round r of E_{U} and E_{L}. We encode the deterministic truncated differential trail propagation through E_{U} and E_{L} in opposite directions as two independent CSP problems using the rules described in Section 2.5. We exclude trivial solutions by adding the constraints $\sum_{i=0}^{m-1} \mathtt{AXU}_0[i] \neq 0$ and $\sum_{i=0}^{m-1} \mathtt{AXL}_{r_{\mathrm{L}}} \neq 0$. Let $\mathit{CSP}_{\mathrm{U}}(\mathtt{AXU}_0, \mathtt{DXU}_0, \ldots, \mathtt{AXU}_{r_{\mathrm{U}}}, \mathtt{DXU}_{r_{\mathrm{U}}})$ be the model for propagation of deterministic truncated trails over E_{U} and $\mathit{CSP}_{\mathrm{L}}(\mathtt{AXL}_0, \mathtt{DXL}_0, \ldots, \mathtt{AXL}_{r_{\mathrm{L}}}, \mathtt{DXL}_{r_{\mathrm{L}}})$ for E_{L}^{-1}.

The last internal state in E_{U} and the first internal state of E_{L} overlap at the meeting point as they correspond to the same internals state. We define some

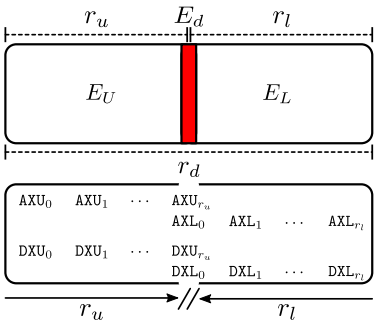

Fig. 3: Modeling the miss-in-the-middle technique as a CSP problem

additional constraints to ensure the incompatibility between the deterministic differential trails of E_{U} and E_{L} at the position of the meeting point:

$$\begin{aligned}&\mathit{CSP}_M(\mathtt{AXU}_{r_{\mathrm{L}}}, \mathtt{DXU}_{r_{\mathrm{L}}}, \mathtt{AXL}_0, \mathtt{DXL}_0) \coloneqq \\ &\bigvee_{i=0}^{m-1} \begin{pmatrix} (\mathtt{AXU}_{r_{\mathrm{U}}}[i] + \mathtt{AXL}_0[i] < 3) \wedge \\ \mathtt{AXU}_{r_{\mathrm{U}}}[i] \neq \mathtt{AXL}_0[i] \end{pmatrix} \vee \bigvee_{i=0}^{m-1} \begin{pmatrix} \mathtt{AXU}_{r_{\mathrm{U}}}[i] = 1 \ \wedge \mathtt{AXL}_0[i] \ = 1 \wedge \\ \mathtt{DXU}_{r_{\mathrm{U}}}[i] \neq \mathtt{DXL}_0[i] \end{pmatrix} = \mathit{True}\end{aligned} \tag{6}$$

The constraints included in CSP_M guarantee the incompatibility between the upper and lower deterministic trails in at least one cell at the meeting point. Lastly, we define $\mathit{CSP}_{\mathrm{D}} \coloneqq \mathit{CSP}_{\mathrm{U}} \wedge \mathit{CSP}_{\mathrm{L}} \wedge \mathit{CSP}_M$, which is the union of all three CSPs. As a result, any feasible solution of $\mathit{CSP}_{\mathrm{D}}$ corresponds to an impossible differential. We can follow the same approach to find ZC distinguishers.

Although we encode the deterministic truncated trails in the same way as [42], our method to search for distinguishers has some important differences. Sun et al. [42] solves $\mathit{CSP}_{\mathrm{U}}$ and $\mathit{CSP}_{\mathrm{L}}$ separately through a loop where the activeness pattern of a cell at the meeting point is fixed in each iteration. The main advantage of our model is that any solutions of $\mathit{CSP}_{\mathrm{D}}$ corresponds to an ID (or ZC) distinguisher. In addition, we do not constrain the value of our model at the input/output or at meeting point. These key feature enables us to extend our model for the key recovery and build a unified COP for finding the nearly optimum ID and ZC attacks in the next sections.

We showed how to encode and detect the contradiction in the meeting point. However, the contradiction may occur in other positions, such as in the tweakey schedule (see Theorem 2), leading to longer distinguishers. Next, we show how to generalize this approach to detect the contradiction in the tweakey schedule while searching for ZC-integral distinguishers according to Theorem 2.

Consider a block cipher E that follows the STK construction with z parallel independent paths in the tweakey schedule. Assume that E applies the permutation h to shuffle the position of cells in each path of tweakey schedule. Let $STK_r[i]$ be the ith cell of subtweakey after r rounds. For all $i = 0, \ldots, m-1$, we define the integer variable $\mathtt{ASTK}_r[i] \in \{0, 1, 2, 3\}$, to indicate the activeness pattern of $STK_r[i]$. Then we define the following constraints to ensure that there

is a contradiction in the tweakey schedule and the condition of Theorem 2 holds:

$$\begin{aligned}
&\mathit{CSP}_{TK}(\mathtt{ASTK}_0, \ldots, \mathtt{ASTK}_{r_\mathrm{D}-1}) \coloneqq \\
&\bigvee_{i=0}^{m-1} \left(\left(\begin{array}{l} \sum_{r=0}^{r_\mathrm{D}-1} \mathtt{bool2int}\left(\mathtt{ASTK}_r[h^{-r}(i)] \neq 0\right) \leq z \\ \quad \wedge \bigvee_{r=0}^{r_\mathrm{D}-1} \left(\mathtt{ASTK}_r[h^{-r}(i)] = 1\right) \end{array} \right) \vee \left(\bigwedge_{r=0}^{r_\mathrm{D}-1} \mathtt{ASTK}_r[h^{-r}(i)] = 0 \right) \right) \qquad (7)
\end{aligned}$$

Equation 7 guarantees that at least one path of the tweakey schedule has at most z active cells, or it is totally inactive. Finally, we create the CSP problem $\mathit{CSP}_\mathrm{D} \coloneqq \mathit{CSP}_\mathrm{U} \wedge \mathit{CSP}_\mathrm{L} \wedge \mathit{CSP}_{TK}$ to find ZC distinguishers of tweakable block ciphers taking the tweakey schedule into account. According to Equation 7, if the sequence of linear masks in the involved tweakey lane has z non-zero values, i.e., $\{1, 2\}$, then at least one of the taken non-zero values should be 1. We also practically verified on reduced-round examples that this condition is indeed necessary to obtain valid ZC-integral distinguishers. This essential condition is ignored in [48]; unfortunately, their claimed distinguishers (and hence their attacks) are invalid. We contacted the authors of [48], and they confirmed our claim.

In our model for distinguisher, we assume that the round keys are independent. Thus, our method regards even those differential or linear propagations over multiple rounds that cannot occur due to the global dependency between the round keys as possible propagations. We also consider the S-box as a black box and do not exploit its internal structure. As a result, regardless of the (twea)key schedule and the choice of S-box, the ID/ZC/Integral distinguishers discovered by our method are always valid.

Before extending our models for key recovery, we first show some of the interesting features of our new model for distinguishers. We can optimize the desired property by adding an objective function to our CSP models for finding distinguishers. According to Theorem 1, maximizing the number of active cells at the input of the ZC linear hull is equivalent to minimizing the data complexity of the corresponding integral distinguisher. Therefore, we maximize the integer addition of the activeness pattern at the input of the ZC-Integral distinguisher. Thanks to this feature, we discovered many practical integral distinguishers for reduced-round Deoxys-BC, SKINNY, SKINNYe-v2, SKINNYee, and CRAFT. Table 3 briefly describes the specification of our integral distinguishers for five ciphers. We note that finding integral distinguishers with minimum data complexity is a challenging task using division property [15, 44] or monomial prediction [19, 22], especially when the block cipher employs large S-boxes. However, our tool can find integral distinguishers with low data complexity by only one iteration that takes a few seconds on a regular laptop. For a more detailed comparison between our method and monomial prediction or division property, see Section M.

Table 3: Summary of integral distinguishers for some ciphers, cell size $c \in \{4, 8\}$.

Cipher	#Rounds	Data complexity	Ref.
SKINNY-n-n	10 / 11 / 12	$\mathbf{2^{5 \cdot c}}$ / $\mathbf{2^{8 \cdot c}}$ / $2^{13 \cdot c}$	J
SKINNY-n-$2n$	12 / 13 / 14	$\mathbf{2^{6 \cdot c}}$ / $\mathbf{2^{9 \cdot c}}$ / $2^{14 \cdot c}$	J
SKINNY-n-$3n$	14 / 15 / 16	$\mathbf{2^{7 \cdot c}}$ / $2^{10 \cdot c}$ / $2^{15 \cdot c}$	J
SKINNYe-v2 / SKINNYee	16 / 17 / 18	$\mathbf{2^{32}}$ / 2^{44} / 2^{64}	J
CRAFT	12 / 13 / 14 / 15	$\mathbf{2^{28}}$ / 2^{44} / 2^{56} / 2^{64}	K.4
Deoxys-BC-256	5 / 6	$\mathbf{2^{24}}$ / 2^{56}	L
Deoxys-BC-384	6 / 7	$\mathbf{2^{32}}$ / 2^{64}	L

4 Modeling the Key Recovery for Impossible Differentials

In this section, we present a generic framework which receives four integer numbers (r_{B}, r_{U}, r_{L}, r_{F}) specifying the lengths of each part in Figure 1, and outputs an optimized full ID attack for $r = r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}} + r_{\mathrm{F}}$ rounds of the targeted block cipher. To this end, we extend the CSP model for ID distinguishers in Section 3 to make a unified COP model for finding an optimized full ID attack taking all critical parameters affecting the final complexity into account.

Before discussing our framework, we first reformulate the complexity analysis of the ID attack to make it compatible with our COP model. Suppose that the block size is n bits and the key size is k bits. Let N be the number of pairs generated in the pair generation phase, and P represents the probability that a wrong key survives the guess-and-filter step. According to Section 2.1, $P = (1 - 2^{-(c_{\mathrm{B}}+c_{\mathrm{F}})})^N$. Let g be the number of key bits we can retrieve through the guess-and-filter step, i.e., $P = 2^{-g}$. Since $P < \frac{1}{2}$, we have $1 < g \leq |k_{\mathrm{B}} \cup k_{\mathrm{F}}|$. Assuming that $(1-2^{-(c_{\mathrm{B}}+c_{\mathrm{F}})})^N \approx e^{-N \cdot 2^{-(c_{\mathrm{B}}+c_{\mathrm{F}})}}$, we have $N = 2^{c_{\mathrm{B}}+c_{\mathrm{F}}+\log_2(g)-0.53}$. Moreover, suppose that $LG(g) = \log_2(g) - 0.53$. Therefore, we can reformulate the complexity analysis of the ID attack as follows:

$$
\begin{aligned}
&T_0 = \max \left\{ \begin{array}{l} \min\limits_{\Delta \in \{\Delta_{\mathrm{B}}, \Delta_{\mathrm{F}}\}} \{2^{\frac{c_{\mathrm{B}}+c_{\mathrm{F}}+n+1-|\Delta|+LG(g)}{2}}\}, \\ 2^{c_{\mathrm{B}}+c_{\mathrm{F}}+n+1-|\Delta_{\mathrm{B}}|-|\Delta_{\mathrm{F}}|+LG(g)} \end{array} \right\},\ T_0 < 2^n, \\
&T_1 = 2^{c_{\mathrm{B}}+c_{\mathrm{F}}+LG(g)},\ T_2 = 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|+LG(g)},\ T_3 = 2^{k-g}, \\
&T_{\mathrm{tot}} = (T_0 + (T_1 + T_2)\, C_{E'} + T_3)\, C_E,\ T_{\mathrm{tot}} < 2^k, \\
&M_{\mathrm{tot}} = \min\left\{2^{c_{\mathrm{B}}+c_{\mathrm{F}}+LG(g)}, 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|}\right\},\ M_{\mathrm{tot}} < 2^k.
\end{aligned}
\tag{8}
$$

When searching for an optimal full ID attack, we aim to minimize the total time complexity while keeping the memory and data complexities under the threshold values. As can be seen in Equation 8, $c_{\mathrm{B}}, c_{\mathrm{F}}, |\Delta_{\mathrm{B}}|, |\Delta_{\mathrm{F}}|$, and $|k_{\mathrm{B}} \cup k_{\mathrm{F}}|$, are the critical parameters which directly affect the final complexity of the ID attack. To determine $(c_{\mathrm{B}}, |\Delta_{\mathrm{B}}|)$, we need to model the propagation of truncated

differential trails through E_B, taking the probability of all differential cancellations into account. To determine k_B, we need to detect the state cells whose difference or data values are needed through the partial encryption over E_B. The same applies for partial decryption over E_F^{-1} to determine c_F, $|\Delta_F|$, k_F. Moreover, to determine the actual size of $k_B \cup k_F$, we should take the (twea)key schedule and key-bridging technique into account.

4.1 Overview of the COP Model

Our model includes several components:

- **Model the distinguisher** as in Section 3. Unlike the previous methods, our model imposes no constraints on the input/output of the distinguisher.
- **Model the difference propagation in outer parts** for truncated trails $\Delta_B \xleftarrow{E_B^{-1}} \Delta_U$ and $\Delta_L \xrightarrow{E_F} \Delta_F$ with probability one. Unlike our model for the distinguisher part, where we use integer variables with domain $\{0, \ldots, 3\}$, here, we only use binary variables to encode active/inactive cells. We also model the number of filters c_B and c_F using new binary variables and constraints to encode the probability of $\Delta_B \xrightarrow{E_B} \Delta_U$ and $\Delta_L \xleftarrow{E_F^{-1}} \Delta_F$.
- **Model the guess-and-determine in outer parts.** In this component, we model the determination relationships over E_B and E_F to detect the state cells whose difference or data values must be known for verifying the differences Δ_U, and Δ_L. Moreover, we model the relation between round (twea)keys and the internal state to detect the (twea)key cells whose values should be guessed during the determination of data values over E_B, and E_F.
- **Model the key bridging.** In this component, we model the (twea)key schedule to determine the number of involved sub-(twea)keys in the key recovery. For this, we can use the general CP-based model for key-bridging proposed by Hadipour and Eichlseder in [18], or cipher-dedicated models.
- **Model the complexity formulas.** In this component, we model the complexity formulas in Equation 8 with the following constraints:

$$
\begin{aligned}
&D[0] \coloneqq \mathit{min}_{\Delta \in \{\Delta_B, \Delta_F\}} \{\tfrac{1}{2}(c_B + c_F + n + 1 - |\Delta| + LG(g))\},\\
&D[1] \coloneqq c_B + c_F + n + 1 - |\Delta_B| - |\Delta_F| + LG(g),\\
&T[0] \coloneqq \mathit{max}\{D[0], D[1]\},\ T[0] < n,\\
&T[1] \coloneqq c_B + c_F + LG(g),\ T[2] \coloneqq |k_B \cup k_F| + LG(g),\ T[3] \coloneqq k - g,\\
&T \coloneqq \mathit{max}\{T[0], T[1], T[2], T[3]\},\ T < k.
\end{aligned}
\tag{9}
$$

  Lastly, we set the objective function to `Minimize T`.

All variables in our model are binary or integer variables with a limited domain except for D and $T[i]$ for $i \in \{0, 1, 2, 3\}$ in Equation 9, which are real numbers. MiniZinc and many MILP solvers such as Gurobi support max, and min operators. We also precompute the values of $LG(g)$ with 3 floating point precision for all $g \in \{2, \ldots, k\}$, and use the $\mathtt{table}$ feature of MiniZinc to model $LG(g)$. As a result, our COP model considers all the critical parameters of the

ID attacks. We recall that the only inputs of our tool are four integer numbers to specify the lengths of $E_{\mathrm{B}}, E_{\mathrm{U}}, E_{\mathrm{L}}$, and E_{F}. So, one can try different lengths for these four parts to find a nearly optimal attack. We can also modify the objective function of our model to minimize the data or memory complexities where time or any other parameter is constrained. One can extend this single-tweakey model for the related-tweakey setting, as we will show next.

4.2 Detailed model for SKINNY

Next, we show in more detail how to perform each step. To this end, we build the COP model for finding full related-tweakey ID attacks on SKINNY as an example. We choose the largest variant of SKINNY, i.e., SKINNY-n-$3n$ with cell size $c \in \{4, 8\}$ to explain our model (see Appendix C for the cipher specification). In what follows, given four integer numbers $r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{L}}, r_{\mathrm{F}}$, we model the full ID attack on $r = r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}} + r_{\mathrm{F}}$ rounds of SKINNY, where $r_{\mathrm{D}} = r_{\mathrm{U}} + r_{\mathrm{L}}$ is the length of the distinguisher and r_{B}, and r_{F} are the lengths of extended parts in backward and forward directions, respectively.

Model the distinguisher We first model the difference propagation through the tweakey schedule of SKINNY. For the tweakey schedule of SKINNY, we can either use the word-wise model proposed in [3] or a bit-wise model (see algorithm 1). Here, we explain the bit-wise model. The tweakey path of $TK1$ only shuffles the position of tweakey cells in each round. Thus, for tweakey path $TK1$, we only define the integer variable $\mathtt{DTK1}[i]$ to encode the c-bit difference in the ith cell of $TK1$. For tweakey path TKm, where $m \in \{2, 3\}$, we define the integer variables $\mathtt{DTK}m_r[i]$ to encode the c-bit difference value in the ith cell of TKm_r, where $0 \leq i \leq 15$. We also define the integer variables $\mathtt{ASTK}_r[i]$ and $\mathtt{DSTK}_r[i]$ to encode the activeness pattern as well as the c-bit difference value in the ith cell of STK_r. Our CSP model for the tweakey schedule of SKINNY is a bit-wise model. We use the *table* feature of MiniZinc to encode the LFSRs. To this end, we first precompute the LFSR as a lookup table and then constrain the variables at the input/output of LFSR to satisfy the precomputed lookup table. This approach is applicable for encoding any function that can be represented as an integer lookup table, such as DDT/LAT of S-boxes. We tested word-wise and bit-wise models and found the word-wise model more efficient.

In the data path of SKINNY, SubCells, AddRoundTweakey, and MixColumns can change the activeness pattern of the state while propagating the deterministic differences. Thus, for the internal state before and after these basic operations, we define two types of variables to encode the activeness pattern and difference value in each state cell. Next, as described in algorithm 2 and algorithm 6, we build $\mathit{CSP}_{\mathrm{U}}$ and $\mathit{CSP}_{\mathrm{L}}$. We also build the CSP_M according to Equation 6. The combined CSP model is $\mathit{CSP}_{\mathrm{D}} \coloneqq \mathit{CSP}_{\mathrm{U}} \wedge \mathit{CSP}_{\mathrm{L}} \wedge \mathit{CSP}_M \wedge \mathit{CSP}_{DTK}$. Hence, any feasible solution of $\mathit{CSP}_{\mathrm{D}}$ corresponds to a related-tweakey ID distinguisher for SKINNY-n-$3n$. By setting $\mathtt{DTK3}_0$ in algorithm 1 to zero, we can find related-tweakey ID distinguishers for SKINNY-n-$2n$. We can also set $\mathtt{DTK1}, \mathtt{DTK2}_0, \mathtt{DTK3}_0$ in algorithm 1 to zero to find single-tweakey ID distinguishers of SKINNY.

Algorithm 1: CSP model for the tweakey schedule of SKINNY

Input: Four integer numbers $(r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{L}}, r_{\mathrm{F}})$
Output: CSP_{DTK}

$R \leftarrow r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}} + r_{\mathrm{F}} - 1$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK1}[i] \in \{0, \ldots, 2^c - 1\} : \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK2}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK3}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{ASTK}_r[i] \in \{0, 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 7\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DSTK}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 7\}$;
for $r = 0, \ldots, R; \ i = 0, \ldots, 7$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{Link}(\mathtt{ASTK}_r[i], \mathtt{DSTK}_r[i])$;
for $r = 1, \ldots, R; \ i = 0, \ldots 15$ **do**
    if $i \leq 7$ **then**
        $\mathcal{M}.\mathtt{con} \leftarrow \mathit{table}([\mathtt{DTK2}_{r-1}[h(i)], \mathtt{DTK2}_r[i]], \mathit{lfsr2})$;
        $\mathcal{M}.\mathtt{con} \leftarrow \mathit{table}([\mathtt{DTK3}_{r-1}[h(i)], \mathtt{DTK3}_r[i]], \mathit{lfsr3})$;
    else
        $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{DTK2}_r[i] = \mathtt{DTK2}_{r-1}[h(i)]$;
        $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{DTK3}_r[i] = \mathtt{DTK3}_{r-1}[h(i)]$;
for $r = 0, \ldots, R; \ i = 0, \ldots 7$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{DSTK}_r[i] = \mathtt{DTK1}[h^r(i)] \oplus \mathtt{DTK2}_r[i] \oplus \mathtt{DTK3}_r[i]$;
return $\mathcal{M}$;

The first operation in the round function of SKINNY is SubCells. However, we can consider the first SubCells layer as a part of E_{B} and start the distinguisher after it. This way, our model takes advantage of the differential cancellation over the AddRoundTweakey and MixColumns layers to derive longer distinguishers. It happens if the input differences in the internal state (or tweakey paths) are fixed and can cancel out each other through AddRoundTweakey or MixColumns. In this case, we skip the constraints in line 14 of algorithm 2 for the first round, $r = 0$.

Model the difference propagation in outer parts To model the deterministic difference propagations $\Delta_{\mathrm{B}} \xleftarrow{E_{\mathrm{B}}^{-1}} \Delta_{\mathrm{U}}$, and $\Delta_{\mathrm{L}} \xrightarrow{E_{\mathrm{F}}} \Delta_{\mathrm{F}}$, we define a binary variable for each state cell to indicate whether its difference value is zero. Since the SubCells layer does not change the status of state cells in terms of having zero/nonzero differences, we ignore it in this model.

To model the probability of difference propagations $\Delta_{\mathrm{B}} \xrightarrow{E_{\mathrm{B}}} \Delta_{\mathrm{U}}$, and $\Delta_{\mathrm{L}} \xleftarrow{E_{\mathrm{F}}^{-1}} \Delta_{\mathrm{F}}$, note that there are two types of probabilistic transitions. The first type is differential cancellation through an XOR operation. The second type is any differential transition ($\mathtt{truncated} \xrightarrow{S} \mathtt{fixed}$) for S-boxes; this is only considered at the distinguisher's boundary, at the first S-box layer of E_{F} or the last of E_{B}.

Let $Z = X \oplus Y$, where $X, Y, Z \in \mathbb{F}_2^c$. Let $\mathtt{AX}, \mathtt{AY}, \mathtt{AZ} \in \{0, 1\}$ indicate whether the difference of X, Y, Z are zero. We define the new constraint XOR_1 to model

Algorithm 2: CSP_{U} for upper trail in distinguisher of SKINNY

Input: $CSP_{DTK}.\texttt{var}$ and the integer numbers $r_{\text{B}}, r_{\text{U}}$
Output: CSP_{U}

$r_{\text{off}} \leftarrow r_{\text{B}}$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\texttt{var} \leftarrow CSP_{DTK}.\texttt{var}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AXU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DXU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AYU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}-1, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DYU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\text{U}}-1, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AZU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}-1, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DZU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\text{U}}-1, \ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{con} \leftarrow \sum_{i=0}^{15} \texttt{AXU}_0[i] + \sum_{i=0}^{15} \texttt{DTK1}[i] + \sum_{i=0}^{15} \texttt{DTK2}_0 + \sum_{i=0}^{15} \texttt{DTK3}_0[i] \ge 1$;
for $r = 0,\dots,r_{\text{U}}-1, \ i = 0,\dots,15$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{Link}(\texttt{AXU}_r[i],\texttt{DXU}_r[i]) \wedge \mathit{Link}(\texttt{AYU}_r[i],\texttt{DYU}_r[i]) \wedge \mathit{Link}(\texttt{AZU}_r[i],\texttt{DZU}_r[i])$;
for $r = 0,\dots,r_{\text{U}}-1, \ i = 0,\dots,15$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{S\text{-}box}(\texttt{AXU}_r[i], \texttt{AYU}_r[i])$;
for $r = 0,\dots,r_{\text{U}}-1, \ i = 0,\dots,7$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}(\texttt{AYU}_r[i], \texttt{DYU}_r[i], \texttt{ASTK}_{r_{\text{off}}+r}[i], \texttt{DSTK}_{r_{\text{off}}+r}[i], \texttt{AZU}_r[i], \texttt{DZU}_r[i])$;
  $\mathcal{M}.\texttt{con} \leftarrow (\texttt{AZU}_r[i+8] = \texttt{AYU}_r[i+8]) \wedge (\texttt{DZU}_r[i+8] = \texttt{DYU}_r[i+8])$;
for $r = 0,\dots,r_{\text{U}}-1, \ i = 0,\dots,3$ **do**
  $\mathit{I}_1 \leftarrow [\texttt{AZU}_r[P[i]], \texttt{AZU}_r[P[i+4]], \texttt{AZU}_r[P[i+8]], \texttt{AZU}_r[P[i+12]]]$;
  $\mathit{I}_2 \leftarrow [\texttt{DZU}_r[P[i]], \texttt{DZU}_r[P[i+4]], \texttt{DZU}_r[P[i+8]], \texttt{DZU}_r[P[i+12]]]$;
  $\mathit{O}_1 \leftarrow [\texttt{AXU}_{r+1}[i], \texttt{AXU}_{r+1}[i+4], \texttt{AXU}_{r+1}[i+8], \texttt{AXU}_{r+1}[i+12]]$;
  $\mathit{O}_2 \leftarrow [\texttt{DXU}_{r+1}[i], \texttt{DXU}_{r+1}[i+4], \texttt{DXU}_{r+1}[i+8], \texttt{DXU}_{r+1}[i+12]]$;
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{Mdiff}(\mathit{I}_1, \mathit{I}_2, \mathit{O}_1, \mathit{O}_2)$;
return $\mathcal{M}$;

the difference propagation with probability one through XOR:

$$\mathit{XOR}_1(\texttt{AX},\texttt{AY},\texttt{AZ}) \coloneqq (\texttt{AZ} \ge \texttt{AX}) \wedge (\texttt{AZ} \ge \texttt{AY}) \wedge (\texttt{AZ} \le \texttt{AX} + \texttt{AY}) \tag{10}$$

We define a binary variable $\texttt{CB}_r[i]$ ($\texttt{CF}_r[i]$) for each XOR operation in the rth round of E_{B} (resp. E_{F}) to indicate whether there is a difference cancellation over the corresponding XOR, where $0 \le i \le 19$. We also define the following constraint to encode the differential cancellation for each XOR operation:

$$\mathit{XOR}_p(\texttt{AX},\texttt{AY},\texttt{AZ},\texttt{CB}) \coloneqq \mathit{if}\ (\texttt{AX}+\texttt{AY}=2 \wedge \texttt{AZ}=0)\ \mathit{then}\ \texttt{CB}=1\ \mathit{else}\ \texttt{CB}=0 \tag{11}$$

Algorithm 3 and algorithm 7 describe our model for difference propagation over E_{B} and E_{F}. We combine CSP_{B}^{dp} and CSP_{F}^{dp} into $CSP_{\text{DP}} \coloneqq CSP_{\text{B}}^{dp} \wedge CSP_{\text{F}}^{dp}$ to model the difference propagation through the outer parts.

Model the guess-and-determine in outer parts We now detect the state cells whose difference or value is needed for the filters in $\Delta_{\text{B}} \to \Delta_{\text{U}}$ and $\Delta_{\text{L}} \leftarrow \Delta_{\text{F}}$.

Algorithm 3: CSP_{B}^{dp} difference propagation through E_{B} for SKINNY

Input: $CSP_{DTK}.\texttt{var}$, $CSP_{\text{U}}.\texttt{var}$ and the integer number r_{B}
Output: CSP_{B}^{dp}

1 Declare an empty CSP model $\mathcal{M}$;
2 $\mathcal{M}.\texttt{var} \leftarrow CSP_{DTK}.\texttt{var}$;
3 $\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AXB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\text{B}},\ 0 \le i \le 15\}$;
4 $\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AZB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\text{B}}-1,\ 0 \le i \le 15\}$;
5 $\mathcal{M}.\texttt{var} \leftarrow \{\texttt{CB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\text{B}}-1,\ 0 \le i \le 19\}$;
6 **for** $i = 0, \ldots, 15$ **do**
7 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{if}\ \texttt{AXU}_0[i] \ge 1\ \mathit{then}\ \texttt{AXB}_{r_{\text{B}}}[i] = 1\ \mathit{else}\ \texttt{AXB}_{r_{\text{B}}}[i] = 0$;
8 **for** $r = 0, \ldots, r_{\text{B}}-1,\ i = 0, \ldots, 3$ **do**
9 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{Minvdiff}_1 \left(\begin{pmatrix} \texttt{AXB}_{r+1}[i] \\ \texttt{AXB}_{r+1}[i+4] \\ \texttt{AXB}_{r+1}[i+8] \\ \texttt{AXB}_{r+1}[i+12] \end{pmatrix}, \begin{pmatrix} \texttt{AZB}_r[P[i]] \\ \texttt{AZB}_r[P[i+4]] \\ \texttt{AZB}_r[P[i+8]] \\ \texttt{AZB}_r[P[i+12]] \end{pmatrix} \right)$;
10 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}_p(\texttt{AZB}_r[P[i+4]], \texttt{AZB}_r[P[i+8]], \texttt{AXB}_{r+1}[i+8], \texttt{CB}_r[i])$;
11 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}_p(\texttt{AZB}_r[P[i]], \texttt{AZB}_r[P[i+8]], \texttt{AXB}_{r+1}[i+12], \texttt{CB}_r[i+4])$;
12 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}_p(\texttt{AXB}_{r+1}[i+12], \texttt{AZB}_r[P[i+12]], \texttt{AXB}_{r+1}[i], \texttt{CB}_r[i+8])$;
13 **for** $r = 0, \ldots, r_{\text{B}}-1,\ i = 0, \ldots, 7$ **do**
14 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}_1(\texttt{AZB}_r[i], \texttt{ASTK}_r[i], \texttt{AXB}_r[i])$;
15 $\quad \mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}_p(\texttt{AXB}_r[i], \texttt{ASTK}_r[i], \texttt{AZB}_r[i], \texttt{CB}_r[i+12])$;
16 $\quad \mathcal{M}.\texttt{con} \leftarrow (\texttt{AXB}_r[i+8] = \texttt{AZB}_r[i+8])$;
17 **return** $\mathcal{M}$;

We first discuss detecting the state cells whose difference values are needed. The difference value in a state cell is needed if the corresponding state cell contributes to a filter, i.e., a differential cancellation. We know that AddRoundTweakey and MixColumns are the only places where a differential cancellation may occur. We thus define the binary variables $\texttt{KDXB}_r[i]$ and $\texttt{KDZB}_r[i]$ to indicate whether the difference value of $X_r[i]$ and $Z_r[i]$ over E_b should be known. We recall that the difference cancellation through each XOR over E_b is already encoded by $\texttt{CB}_r[i]$. If $\texttt{CB}_r[i] = 1$, then the difference value in the state cells contributing to this differential cancellation is needed. For instance, if $\texttt{CB}_r[i] = 1$, then $\texttt{KDZB}_r[P[i+4]] = 1$ and $\texttt{KDZB}_r[P[i+4]] = 1$, where $0 \le i \le 3$ and $0 \le r \le r_u - 1$. Besides detecting the new state cells whose difference values are needed in each round, we encode the propagation of this property from the previous rounds, as in lines 14–17 of algorithm 4. We also define new constraint (line 11) to link the beginning of E_{U} to the end of E_{B}. For E_{F}, we also define new binary variables $\texttt{KDXF}_r[i]$ and $\texttt{KDZF}_r[i]$ to indicate whether the difference values of $X_r[i]$ and $Z_r[i]$ are needed. Then, we follow a similar approach to model the determination of difference values.

When modeling the determination of data values, SubCells comes into effect. We explain modeling the determination of data values over S-boxes in E_{B}; a similar model can be used for E_{F}. Suppose that $Y_r[i] = S(X_r[i])$, and the value

Algorithm 4: $\mathit{CSP}_{\mathrm{B}}^{gd}$ guess-and-determine through E_{B} for SKINNY

Input: $\mathit{CSP}_{\mathrm{U}}.\mathtt{var}$, $\mathit{CSP}_{\mathrm{B}}^{dp}$ and the integer number r_{B}
Output: $\mathit{CSP}_{\mathrm{B}}^{gd}$

Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{\mathrm{U}}.\mathtt{var} \cup \mathit{CSP}_{\mathrm{B}}^{dp}.\mathtt{var}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KDXB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{con} \leftarrow \{\mathtt{KDXB}_r[i] \le \mathtt{AXB}_r[i] :\ 0 \le r \le r_{\mathrm{B}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KDZB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{con} \leftarrow \{\mathtt{KDZB}_r[i] \le \mathtt{AZB}_r[i] :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KXB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KYB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{IKB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 15\}$;
for $i = 0, \ldots, 15$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AXU}_0[i] = 1\ \mathit{then}\ \mathtt{KDXB}_{r_{\mathrm{B}}}[i] = 1\ \mathit{else}\ \mathtt{KDXB}_{r_{\mathrm{B}}}[i] = 0$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AYU}_0[i] = 1\ \mathit{then}\ \mathtt{KXB}_{r_{\mathrm{B}}}[i] = 1\ \mathit{else}\ \mathtt{KXB}_{r_{\mathrm{B}}}[i] = 0$;
for $r = 0, \ldots, r_{\mathrm{B}} - 1,\ i = 0, \ldots, 3$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDXB}_{r+1}[i] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDZB}_r[P[i]] = \mathtt{AZB}_r[P[i]] \wedge \\ \mathtt{KDZB}_r[P[i+8]] = \mathtt{AZB}_r[P[i+8]] \wedge \\ \mathtt{KDZB}_r[P[i+12]] = \mathtt{AZB}_r[P[i+12]] \end{pmatrix}$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDXB}_{r+1}[i+4] = 1\ \mathit{then}\ \mathtt{KDZB}_r[P[i]] = \mathtt{AZB}_r[P[i]]$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDXB}_{r+1}[i+8] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDZB}_r[P[i+4]] = \mathtt{AZB}_r[P[i+4]] \wedge \\ \mathtt{KDZB}_r[P[i+8]] = \mathtt{AZB}_r[P[i+8]] \end{pmatrix}$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDXB}_{r+1}[i+12] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDZB}_r[P[i]] = \mathtt{AZB}_r[P[i]] \wedge \\ \mathtt{KDZB}_r[P[i+8]] = \mathtt{AZB}_r[P[i+8]] \end{pmatrix}$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CB}_r[i] = 1\ \mathit{then}\ (\mathtt{KDZB}_r[P[i+4]] = 1 \wedge \mathtt{KDZB}_r[P[i+8]] = 1)$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CB}_r[i+4] = 1\ \mathit{then} (\mathtt{KDZB}_r[P[i]] = 1 \wedge \mathtt{KDZB}_r[P[i+8]] = 1)$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CB}_r[i+8] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDZB}_r[P[i]] = \mathtt{AZB}_r[P[i]] \wedge \\ \mathtt{KDZB}_r[P[i+8]] = \mathtt{AZB}_r[P[i+8]] \wedge \\ \mathtt{KDZB}_r[P[i+12]] = 1 \end{pmatrix}$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{Minvdata} \left(\begin{pmatrix} \mathtt{KXB}_{r+1}[i] \\ \mathtt{KXB}_{r+1}[i+4] \\ \mathtt{KXB}_{r+1}[i+8] \\ \mathtt{KXB}_{r+1}[i+12] \end{pmatrix}, \begin{pmatrix} \mathtt{KYB}_r[P[i]] \\ \mathtt{KYB}_r[P[i+4]] \\ \mathtt{KYB}_r[P[i+8]] \\ \mathtt{KYB}_r[P[i+12]] \end{pmatrix} \right)$;
for $r = 0, \ldots, r_{\mathrm{B}} - 1,\ i = 0, \ldots, 7$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{KDXB}_r[i] \ge \mathtt{KDZB}_r[i]$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{KDXB}_r[i+8] = \mathtt{KDZB}_r[i+8]$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CB}_r[i+12] = 1\ \mathit{then}\ \mathtt{KDXB}_r[i] = 1$;
    $\mathcal{M}.\mathtt{con} \leftarrow (\mathtt{IKB}_r[i] = \mathtt{KYB}_r[i] \wedge \mathtt{IKB}_r[i+8] = 0)$;
for $r = 0, \ldots, r_{\mathrm{B}} - 1,\ i = 0, \ldots, 15$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{S\text{-}box}_{gd}(\mathtt{KYB}_r[i], \mathtt{KXB}_r[i], \mathtt{KDXB}_r[i])$;
return $\mathcal{M}$;

of ΔX_r is known. If we want to determine the value of $\Delta Y_r[i]$, e.g., to check a filter, we need to know the value of $X_r[i]$. Accordingly, we need the value of $X_r[i]$ if either we want to determine $Y_r[i]$, or we want to determine $\Delta Y_r[i]$. On the other hand, if neither data nor difference values after the S-box is needed, we do not need to know the data value before the S-box. Therefore, we define binary variables $\texttt{KXB}_r[i]$ and $\texttt{KYB}_r[i]$ to indicate whether the values of $X_r[i]$ and $Y_r[i]$ are needed. Then, we model the determination flow over the S-boxes as follows:

$$\mathit{S\text{-}box}_{gd}(\texttt{KXB}_r[i], \texttt{KYB}_r[i], \texttt{KDXB}_r[i]) \coloneqq \begin{cases} (\texttt{KYB}_r[i] \geq \texttt{KXB}_r[i]) \wedge (\texttt{KYB}_r[i] \geq \texttt{KDXB}_r[i]) \wedge \\ (\texttt{KYB}_r[i] \leq \texttt{KXB}_r[i] + \texttt{KDXB}_r[i]) \end{cases}$$

We also model MixColumns according to Equation 16 when encoding the determination of data values over E_{B} and E_{F}.

We now explain how to detect the subtweakey cells that are involved in the determination of data values. Let $\texttt{IKB}_r[i]$ be a binary variable that indicates whether the ith cell of subtweakey in the rth round of E_{B} is involved, where $0 \leq r \leq r_{\mathrm{B}} - 1$ and $0 \leq i \leq 15$. One can see that $\texttt{IKB}_r[i] = 1$ if and only if $i \leq 7$ and $\texttt{KYB}_r[i] = 1$. Otherwise $\texttt{IKB}_r[i] = 0$. We define binary variables $\texttt{IKF}_r[i]$ to encode the involved subtweakey in E_{F} similarly. Algorithm 4 and algorithm 8 describe our CSP models for the guess-and-determine through E_{B} and E_{F}. We refer to $\mathit{CSP}_{\mathrm{GD}} \coloneqq \mathit{CSP}_{\mathrm{B}}^{gd} \wedge \mathit{CSP}_{\mathrm{F}}^{gd}$ as our CSP model for the guess-and-determine through the outer parts.

Model the key bridging Although the subtweakeys involved in E_{B} and E_{F} are separated by r_{D} rounds, they may have some relations due to the tweakey schedule. Guessing the values of some involved key cells may determine the value of others. Key-bridging uses the relations between subwteakeys to reduce the number of actual guessed key variables. We can integrate the generic CSP model for key-bridging over arbitrary tweakey schedules introduced in [18] into our model. However, the tweakey schedule of SKINNY is linear, and we provide a more straightforward method to model the key-bridging of SKINNY. We explain our model for SKINNY-n-$3n$; it can easily be adapted for the smaller variants.

For the ith cell of subtweakey after r rounds, we have $STK_r[i] = TK1[h^r(i)] \oplus LFSR_2^r(TK1[h^r(i)]) \oplus LFSR_3^r(TK3[h^r(i)])$. Accordingly, knowing $STK_r[h^{-r}(i)]$ in 3 rounds yields 3 independent equations in variables $TK1[i]$, $TK2[i]$, $TK3[i]$, which uniquely determine the master tweakey cells $TK1[i]$, $TK2[i]$, and $TK3[i]$. Hence, we do not need to guess $STK_r[h^{-r}(i)]$ for more than 3 different rs. To take this fact into account, we first define new integer variables $\texttt{IK} \in \{0, \ldots, r_{\mathrm{B}} + r_{\mathrm{F}} - 1\}$, $\texttt{KE} \in \{0, 1, 2, 3\}$, and $\texttt{KS} \in \{0, \ldots, 48\}$. Then, assuming that $r_{\mathrm{off}} = r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}}$ and $z = 3$, we use the following constraints to model the key-bridging:

$$\mathit{CSP}_{\mathrm{KB}} \coloneqq \begin{cases} \texttt{IK}[i] = \sum_{r=0}^{r_{\mathrm{B}}-1} \texttt{IKB}_r[h^{-r}(i)] + \sum_{r=0}^{r_{\mathrm{F}}-1} \texttt{IKF}_r[h^{-(r_{\mathrm{off}}+r)}(i)] \text{ for } 0 \leq i \leq 15, \\ \mathit{if}\ \texttt{IK}[i] \geq z\ \mathit{then}\ \texttt{KE}[i] = z\ \mathit{else}\ \texttt{KE}[i] = \texttt{IK}[i] \text{ for } 0 \leq i \leq 15, \\ \texttt{KS} = \sum_{i=0}^{15} \texttt{KE}[i] \end{cases} \tag{12}$$

Model the complexity formulas We now show how to combine all CSP models and model the complexity formulas. The variable KS in Equation 12 determines the number of involved key cells, corresponding to $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = \mathtt{c} \cdot \mathtt{KS}$ involved key bits for cell size c. We can model the other critical parameters of the ID attack as shown in algorithm 5. We combine all CSP problems into a unified model and define an objective function to minimize the time complexity of the ID attack.

Results We applied our method to find full ID attacks on all variants of SKINNY in both single and related-tweakey settings. Our model includes integer and real variables, so we used Gurobi to solve the resulting COP problems. Table 1 shows our results. Our ID attacks' time, date, and memory complexity are much smaller than the best previous ID attacks. Notably, the time complexity of our 19-round single-tweakey ID attack on SKINNY-128-256 (Figure 8, details in Section F.2) is

Algorithm 5: COP model for the full ID attack on SKINNY

Input: Four integer numbers $r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{L}}, r_{\mathrm{F}}$
Output: *COP*

Declare an empty COP model $\mathcal{M}$;
$\mathcal{M} \leftarrow \mathit{CSP}_{\mathrm{D}} \wedge \mathit{CSP}_{\mathrm{DP}} \wedge \mathit{CSP}_{\mathrm{GD}} \wedge \mathit{CSP}_{\mathrm{KB}}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{g} \in \{1, \ldots, \mathtt{z} \cdot 16 \cdot \mathtt{c}\}$; /* Corresponding to parameter g */
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{C_B} \in \{0, \ldots, 20 \cdot r_{\mathrm{B}} + 16\}$; /* Corresponding to c_{B} */
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{C_F} \in \{0, \ldots, 20 \cdot r_{\mathrm{F}} + 16\}$; /* Corresponding to c_{F} */
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{W_B} \in \{0, \ldots, 16\}$; /* Corresponding to $|\Delta_{\mathrm{B}}|$ */
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{W_F} \in \{0, \ldots, 16\}$; /* Corresponding to $|\Delta_{\mathrm{F}}|$ */
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{D}[i] \in [0, \mathtt{z} \cdot 16 \cdot \mathtt{c}] : i \in \{0,1,2,3\}\}$; /* For data complexity */
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{T}[i] \in [0, \mathtt{z} \cdot 16 \cdot \mathtt{c}] : i \in \{0,1,2,3\}\}$; /* For time complexity */
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{T}_{max} \in [0, \mathtt{z} \cdot 16 \cdot \mathtt{c}]$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{C_B} = \sum_{r=1}^{r_{\mathrm{B}}-1} \sum_{i=0}^{19} \mathtt{CB}_r[i] + \sum_{i=0}^{15} \mathtt{KXB}_{r_{\mathrm{B}}}[i]$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{C_F} = \sum_{r=0}^{r_{\mathrm{F}}-2} \sum_{i=0}^{19} \mathtt{CF}_r[i] + \sum_{i=0}^{7} \mathtt{CF}_{r_{\mathrm{F}}-1}[i] + \sum_{i=0}^{15} \mathtt{KXF}_0[i]$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{W_B} = \sum_{i=0}^{15} \mathtt{AXB}_1[i]$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathtt{W_F} = \sum_{i=0}^{15} \mathtt{AXF}_{r_{\mathrm{F}}-1}[i]$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{D}[0] = 0.5 \cdot (\mathtt{c}(\mathtt{C_B} + \mathtt{C_F}) + \mathtt{n} - \mathtt{c} \cdot \mathtt{W_B} + \mathit{LG}(\mathtt{g}) + 2)$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{D}[1] = 0.5 \cdot (\mathtt{c}(\mathtt{C_B} + \mathtt{C_F}) + \mathtt{n} - \mathtt{c} \cdot \mathtt{W_F} + \mathit{LG}(\mathtt{g}) + 2)$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{D}[2] = \mathit{min}(\mathtt{D}[0], \mathtt{D}[1])$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{D}[3] = \mathtt{c} \cdot (\mathtt{C_B} + \mathtt{C_F}) + \mathtt{n} + 1 - \mathtt{c} \cdot (\mathtt{W_B} + \mathtt{W_F}) + \mathit{LG}(\mathtt{g})$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{T}[0] = \mathit{max}(\mathtt{D}[2], \mathtt{D}[3])$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{T}[1] = \mathtt{c} \cdot (\mathtt{C_B} + \mathtt{C_F}) + \mathit{LG}(\mathtt{g})$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{T}[2] = \mathtt{c} \cdot \mathtt{KS} + \mathit{LG}(\mathtt{g})$; /* Corresponding to $|k_{\mathrm{B}} \cup k_{\mathrm{F}}|$ */
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{T}[3] = \mathtt{k} - \mathtt{g}$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{g} \leq \mathtt{c} \cdot \mathtt{KS} \wedge g > 1$;
$\mathcal{M}.\mathtt{con} \leftarrow \mathtt{T}_{max} = \mathit{max}(T[0], T[1], T[2], T[3])$;
$\mathcal{M}.\mathtt{con} \leftarrow (\mathtt{T}[0] < \mathtt{n} + 1 \wedge \mathtt{T}_{max} < \mathtt{k})$;
$\mathcal{M}.\mathtt{obj} \leftarrow \mathtt{Minimize}\ \mathtt{T}_{max}$;
return $\mathcal{M}$;

smaller by a factor of $2^{22.57}$ compared to the best previous one [47]. As another example, we improved the time complexity of the related-tweakey ID attack on SKINNY-128-384 by a factor of $2^{15.39}$ (Figure 10), with smaller data and memory complexity than the best previous one [27]. Our tool can discover the longest ID distinguishers for SKINNY so far in both single and related-tweakey settings. However, we noticed that the best ID attacks do not necessarily rely on the longest distinguishers. For instance, our single-tweakey ID attacks on SKINNY use 11-round distinguishers, whereas our tool also finds 12-round distinguishers.

We also applied our tool to CRAFT and SKINNYee. On CRAFT, we found a 21-round ID attack which is 2 rounds longer than the best previous single-tweakey attack presented at ASIACRYPT 2022 [40]. For SKINNYee, we found a 27-round related-tweakey ID attack. Our tool can produce all the reported results on a laptop in a few seconds. Besides improving the security evaluation against ID attacks, our tool can significantly reduce human effort and error.

We also used our tool to check the validity of the previous results. To do so, we fix the activeness pattern in our model to that at the input/output of the claimed distinguisher. Moreover, we constrain the time, memory, and data complexities to the claimed bounds. An infeasible model indicates potential issues with the claimed attack. We manually check the attack to find the possible issue in this case. If the model is feasible, we match the claimed critical parameters with the output of our tool. In case of any mismatch, we manually check the corresponding parameter in the claimed attack to ensure it is calculated correctly.

We followed this approach to check the validity of the ID attacks on SKINNY proposed in [45]. For example, our tool returns 'unsatisfiable' when we limit it to find a 22-round ID attack on SKINNY-n-$3n$ with the claimed parameters in [45]. To figure out the issue, we relax the time/memory/data complexity bounds and only fix the activeness pattern according to the claimed distinguisher. This way, our tool returns different attack parameters compared to the claimed ones. According to [45, Sec. 6], $c_{\mathrm{B}} + c_{\mathrm{F}}$ is supposed to be $18c$ for 22-round ID attack on SKINNY-n-$3n$ with cell size c. However, our tool returns $c_{\mathrm{B}} = 6c$ and $c_{\mathrm{F}} = 15c$, and thus $c_{\mathrm{B}} + c_{\mathrm{F}} = 21c$. Accordingly, the actual probability that a wrong tweakey is discarded with one pair is about 2^{-21c}. So, the 22-round ID attack on SKINNY-n-$3n$ in [45] requires more data and thus time by a factor of 2^{3c}. The time complexity of the 22-round ID attack on SKINNY-64-192 (SKINNY-128-384) in [45] is $2^{183.97}$ (resp. $2^{373.48}$). As a result, the corrected attack requires more time than the exhaustive search. We also checked the 20-round ID attacks on SKINNY-n-$2n$ in [45]. We noticed that a similar issue makes the corrected attack require more data than the full codebook or more time than the exhaustive search. We contacted the authors of [45], and they confirmed our claim.

5 Modeling the Key Recovery of ZC and Integral Attacks

Similar to our approach for ID attacks, we can extend our models for the ZC and integral distinguishers to make a unified model for finding full ZC and ZC-based integral attacks. One of the critical parameters in the key recovery of

Fig. 4: ID attack on 19 rounds of SKINNY-n-$2n$, $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 26 \cdot c$, $c_{\mathrm{B}} = 6 \cdot c$, $c_{\mathrm{F}} = 15 \cdot c$, $\Delta_{\mathrm{B}} = 7 \cdot c$, $\Delta_{\mathrm{F}} = 16 \cdot c$

ZC and integral attacks is the number of involved key cells in the outer parts. Another effective parameter is the number of involved state cells through the outer parts. Thus, we should consider these parameters when modeling the key recovery of the ZC and integral attacks. Moreover, the meet-in-the-middle and partial-sum techniques are essential to reduce the time complexity of integral attacks. Therefore, taking these techniques into account, we provide a generic CP model for key recovery of ZC and ZC-based integral attacks as follows:

- **Model the distinguisher** as described in Section 3.
- **Model the guess-and-determine part** by modeling the value paths in the outer part and detecting the state/key cells whose values are needed in key recovery.
- **Model the key bridging** for the key recovery.
- **Model the meet-in-the-middle technique** for the key recovery of integral attacks.
- **Set the objective function** to minimize the final time complexity, keeping the data and memory complexities under the thresholds.

We only describe modeling the meet-in-the-middle technique. Other modules can be constructed similarly to our models for ID attacks. Given that there is no restriction for the output of ZC-integral distinguishers in our model, some ZC-integral distinguishers might have more than one balanced output cell. With more than one balanced cell, we might be able to use the meet-in-the-middle (MitM) technique [38] to reduce the time complexity. For example, we can use MitM if the ZC-integral distinguisher of SKINNY has two active output cells in one column, indicating that the sum of these cells is balanced. Then, we can recover the integral sums of these two cells for any keyguess separately and merge compatible key guesses that yield the same sum, i.e., that sum to zero overall.

To consider the MitM technique, we model the path values for each output cell of the distinguisher separately in an independent CP submodel. We also define a new integer variable to capture the number of involved key cells in each path. For example, our CP model for integral attacks on SKINNY splits into 16 submodels for the appended rounds after the distinguisher. Each submodel aims at encoding the involved cells in retrieving a certain output cell of the distinguisher. We note that these submodels, together with our CP model for distinguisher, are all combined into one large unified CP model. This way, we can encode and then minimize the complexity of the most critical path, which requires the maximum number of guessed keys in the guess-and-filter step. Similarly to our CP model for ID attacks, our model for ZC and integral attack receives only four integer numbers as input and returns the full ZC or ZC-based integral attack.

We solve our CP models for integral attacks in two steps with two different objective functions:

- We first solve a CP model to minimize the number of involved key cells.
- Next, we limit the number of involved key cells to the output of the previous step and solve the CP model with the objective of maximizing the number of active cells at the input of ZC-integral distinguisher.

As a result, besides reducing the time complexity, we can reduce the data complexity of the resulting integral attacks. To compute the exact final complexity, we introduce an additional helper tool, `AutoPSy`, which automates the partial-sum technique [16], and apply it as a post-processing step to the CP output. `AutoPSy` optimizes the column order in each round of partial-sum key recovery.

We applied our unified framework for finding full ZC and integral attacks to CRAFT, SKINNYe-v2, SKINNYee, and all variants of SKINNY and obtained a series of substantially improved results. Table 1 briefly describes our results. More details on our ZC and integral attacks can be found in Appendices G, H, and I.3. As can be seen in Figure 14, Figure 15, Figure 19, the inputs of the corresponding ZC distinguishers have 4 active cells, and the outputs have 2 active cells. The previous tools which fix the input/output linear masks to vectors with at most one active cell can not find such a distinguisher.

Our CP models for ZC and integral attacks include only integer variables. Thus, we can take advantage of all integer programming (IP) solvers. We used Or-Tools in this application, and running on a regular laptop, our tool can find all the reported results in a few seconds.

When reproducing the best previous results on SKINNY with our automatic tool, we again noticed some issues in previous works. The previous ZC-integral attacks on SKINNY proposed by Ankele et al. at ToSC 2019 [1] have some minor issues where the propagation in the key recovery part is incorrect. For example, in the 20-round TK2 attack in [1, Figure 20] between X_{18}, Y_{18}, the last row is not shifted; in the 23-round TK3 attack in [1, Figure 22], the mixing between Y_{20}, Z_{20} is not correct. In both cases, this impacts the correctness of all following rounds. However, the attacks can be fixed to obtain similar complexities as claimed.

The comparison with those attacks highlights three advantages of our automated approach: (1) Our approach is much less prone to such small hard-to-spot errors; (2) Our approach can find distinguishers with many active input cells (rather than just one as classical approaches), which is particularly helpful in ZC-integral attacks where a higher input weight implies a lower data complexity; (3) Our approach optimizes the key recovery together with the distinguisher, which together with (2) allows us to attach more key-recovery rounds (7 vs. 5 for TK2 in [1], 9 vs. 7 for TK3 in [1]).

6 Conclusion and Future Works

In this paper, we presented a unified CP model to find full ID, ZC, and ZC-based integral attacks for the first time. Our frameworks are generic and can be applied to word-oriented block ciphers. To show the effectiveness and usefulness of our approach, we applied it to CRAFT, SKINNYe-v2, SKINNYee, and all members of the SKINNY family of block ciphers. In all cases, we obtained a series of substantially improved results compared to the best previous ID, ZC, and integral attacks on these ciphers. Our tool can help the cryptanalysts and the designers of block ciphers to evaluate the security of block ciphers against three important attacks, i.e., ID, ZC, and ZC-based integral attacks, more accurately

and efficiently. While we focused on the application to SPN block ciphers, it is also applicable to Feistel ciphers. Applying our approach to other block ciphers such as AES or Feistel ciphers is an interesting direction for future work.

Our improved results show the advantage of our method. However, it also has some limitations. Our CP model for the distinguisher part detects the contradictions in the level of words and does not exploit the internal structure of S-boxes (i.e., DDT/LAT) to consider bit-level contradictions. Thus, one interesting future work is to provide a unified model considering bit-level contradictions. We note that our CP framework for ID, ZC, and integral attacks is modular. The key-recovery part of our CP model can be combined with other CP-based methods for finding distinguishers. For example, regardless of the distinguisher part, one can feed our CP model for the key-recovery part by a set of input/output activeness patterns for the distinguisher part to find the activeness pattern yielding the best key-recovery attack. Next, one can use a more fine-grained CP model that detects bit-level contradictions to check if the selected activeness pattern yields an ID or ZC distinguisher. We recall that in CP models, we can specify a set of input/output activeness patterns by a set of constraints, and we do not have to enumerate all possible input/output activeness patterns. Currently, our tool automatically applies the partial-sum technique as a post-processing step in integral attacks for a refined complexity analysis. Thus, another interesting future work is integrating the partial-sum technique into our CP model for integral attacks. This way, one may be able to improve the integral attacks further.

Acknowledgments. This work has been supported in part by the Austrian Science Fund (FWF SFB project SPyCoDe). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y.: Zero-correlation attacks on tweakable block ciphers with linear tweakey expansion. IACR Transactions on Symmetric Cryptology **2019**(1), 192–235 (Mar 2019). https://doi.org/10.13154/tosc.v2019.i1.192-235
2. Avanzi, R.: The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/tosc.v2017.i1.4-44
3. 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: CRYPTO 2016. pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
4. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. **2019**(1), 5–45 (2019). https://doi.org/10.13154/tosc.v2019.i1.5-45

5. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer (1999). https://doi.org/10.1007/3-540-48910-X_2
6. Biham, E., Biryukov, A., Shamir, A.: Miss in the middle attacks on IDEA and khufu. In: FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer (1999)
7. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_16
8. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Des. Codes Cryptogr. **70**(3), 369–383 (2014). https://doi.org/10.1007/s10623-012-9697-z
9. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: FSE 2012. LNCS, vol. 7549, pp. 29–48. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_3
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. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to clefia, camellia, lblock and simon. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 179–199. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_10
12. Cui, T., Chen, S., Jia, K., Fu, K., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. IACR Cryptology ePrint Archive, Report 2016/689 (2016), https://eprint.iacr.org/2016/689
13. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer (1997). https://doi.org/10.1007/BFb0052343
14. Derbez, P., Fouque, P.A.: Automatic search of meet-in-the-middle and impossible differential attacks. In: CRYPTO 2016. LNCS, vol. 9815, pp. 157–184. Springer (2016)
15. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In: SAC. LNCS, vol. 11349, pp. 115–138. Springer (2018). https://doi.org/10.1007/978-3-030-10970-7_6
16. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of Rijndael. In: FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer (2000). https://doi.org/10.1007/3-540-44706-7_15
17. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022), https://www.gurobi.com
18. Hadipour, H., Eichlseder, M.: Autoguess: A tool for finding guess-and-determine attacks and key bridges. In: ACNS 2022. LNCS, vol. 13269, pp. 230–250. Springer (2022). https://doi.org/10.1007/978-3-031-09234-3_12
19. Hadipour, H., Eichlseder, M.: Integral cryptanalysis of WARP based on monomial prediction. IACR Trans. Symmetric Cryptol. **2022**(2), 92–112 (2022). https://doi.org/10.46586/tosc.v2022.i2.92-112
20. Hadipour, H., Nageler, M., Eichlseder, M.: Throwing boomerangs into feistel structures: Application to CLEFIA, WARP, LBlock, LBlock-s and TWINE. IACR Trans. Symmetric Cryptol. **2022**(3), 271–302 (2022). https://doi.org/10.46586/tosc.v2022.i3.271-302

21. Hadipour, H., Sadeghi, S., Niknam, M.M., Song, L., Bagheri, N.: Comprehensive security analysis of CRAFT. IACR Trans. Symmetric Cryptol. **2019**(4), 290–317 (2019). https://doi.org/10.13154/tosc.v2019.i4.290-317
22. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_15
23. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer (2014). https://doi.org/10.1007/978-3-662-45608-8_15
24. Knudsen, L.: Deal-a 128-bit block cipher. complexity **258**(2), 216 (1998)
25. Lai, X.: Higher order derivatives and differential cryptanalysis pp. 227–233 (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
26. Lambin, B., Derbez, P., Fouque, P.: Linearly equivalent s-boxes and the division property. Des. Codes Cryptogr. **88**(10), 2207–2231 (2020). https://doi.org/10.1007/s10623-020-00773-4
27. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings. IACR Trans. Symmetric Cryptol. **2017**(3), 37–72 (2017). https://doi.org/10.13154/tosc.v2017.i3.37-72
28. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In: INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer (2008)
29. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In: CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer (2008)
30. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt. LNCS, vol. 7537, pp. 57–76. Springer (2011). https://doi.org/10.1007/978-3-642-34704-7_5
31. Naito, Y., Sasaki, Y., Sugawara, T.: Lightweight authenticated encryption mode suitable for threshold implementation. In: EUROCRYPT 2020. LNCS, vol. 12106, pp. 705–735. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_24
32. Naito, Y., Sasaki, Y., Sugawara, T.: Secret can be public: Low-memory AEAD mode for high-order masking. In: CRYPTO 2022. LNCS, vol. 13509, pp. 315–345. Springer (2022). https://doi.org/10.1007/978-3-031-15982-4_11
33. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard CP modelling language. In: CP 2007. LNCS, vol. 4741, pp. 529–543. Springer (2007)
34. Niu, C., Li, M., Sun, S., Wang, M.: Zero-correlation linear cryptanalysis with equal treatment for plaintexts and tweakeys. In: CT-RSA 2021. LNCS, vol. 12704, pp. 126–147. Springer (2021). https://doi.org/10.1007/978-3-030-75539-3_6
35. Perron, L., Furnon, V.: OR-Tools, https://developers.google.com/optimization/
36. Sadeghi, S., Mohammadi, T., Bagheri, N.: Cryptanalysis of reduced round SKINNY block cipher. IACR Trans. Symmetric Cryptol. **2018**(3), 124–162 (2018). https://doi.org/10.13154/tosc.v2018.i3.124-162
37. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: EUROCRYPT 2017. pp. 185–215. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7
38. Sasaki, Y., Wang, L.: Meet-in-the-middle technique for integral attacks against Feistel ciphers. In: SAC 2012. LNCS, vol. 7707, pp. 234–251. Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_16

39. Shi, D., Sun, S., Derbez, P., Todo, Y., Sun, B., Hu, L.: Programming the demirci-selçuk meet-in-the-middle attack with constraints. In: ASIACRYPT 2018. LNCS, vol. 11273, pp. 3–34. Springer (2018). https://doi.org/10.1007/978-3-030-03329-3_1
40. 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: ASIACRYPT 2022. LNCS, vol. 13791, pp. 410–440. Springer (2022). https://doi.org/10.1007/978-3-031-22963-3_14
41. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., AlKhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: CRYPTO 2015. LNCS, vol. 9215, pp. 95–115. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_5
42. Sun, L., Gerault, D., Wang, W., Wang, M.: On the usage of deterministic (related-key) truncated differentials and multidimensional linear approximations for spn ciphers. IACR Transactions on Symmetric Cryptology **2020**(3), 262–287 (Sep 2020). https://doi.org/10.13154/tosc.v2020.i3.262-287
43. Sun, S., Gerault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., Hu, L.: Analysis of AES, SKINNY, and others with constraint programming. IACR Transactions on Symmetric Cryptology **2017**(1), 281–306 (Mar 2017). https://doi.org/10.13154/tosc.v2017.i1.281-306
44. Todo, Y.: Structural evaluation by generalized integral property. In: EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_12
45. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: AFRICACRYPT 2017. LNCS, vol. 10239, pp. 117–134 (2017). https://doi.org/10.1007/978-3-319-57339-7_7
46. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24
47. Yang, D., Qi, W., Chen, H.: Impossible differential attacks on the SKINNY family of block ciphers. IET Inf. Secur. **11**(6), 377–385 (2017). https://doi.org/10.1049/iet-ifs.2016.0488
48. Zhang, Y., Cui, T., Wang, C.: Zero-correlation linear attack on reduced-round SKINNY. Frontiers of Computer Science **17**(174808 (2023)), 377–385 (2022). https://doi.org/10.1007/s11704-022-2206-2

— Supplementary Material —

A Complexity Analysis of the ID Attack in the Related (Twea)key Setting

In the related (twea)key ID attack, we have access to two encryption (or decryption) oracles employing the keys K and $K \oplus \Delta K$, with a known difference ΔK. The goal is to retrieve the secret key K. Assume that the differential transition $(\Delta K, \Delta_{\mathrm{U}}) \rightarrow \Delta_{\mathrm{L}}$ is impossible. We can mount a key recovery attack similar to the single (twea)key setting. However, in the related (twea)key setting, any structure is encrypted with two different (twea)keys. On the other hand, any plaintext P in each structure yields two different pairs $((K,P),(K \oplus \Delta K, P \oplus \Delta P))$, and $((K \oplus \Delta K, P),(K, P \oplus \Delta P))$. Hence, all formulas in Equation 1 remains unchanged except for T_0 which should be modifed as follows:

$$T_0 = \max\left\{ \min_{\Delta \in \{\Delta_{\mathrm{B}}, \Delta_{\mathrm{F}}\}} \left\{ 2\sqrt{N 2^{n-|\Delta|}} \right\}, N 2^{n+1-|\Delta_{\mathrm{B}}|-|\Delta_{\mathrm{F}}|} \right\}. \tag{13}$$

By the way, the data complexity is equal to T_0. We can reformulate the complexity analysis to a CSP-friendly form as follows:

$$\begin{aligned}
&T_0 = \max\left\{ \begin{matrix} \min\limits_{\Delta \in \{\Delta_{\mathrm{B}}, \Delta_{\mathrm{F}}\}} \{2^{\frac{c_{\mathrm{B}}+c_{\mathrm{F}}+n-|\Delta|+LG(g)}{2}+1}\}, \\ 2^{c_{\mathrm{B}}+c_{\mathrm{F}}+n+1-|\Delta_{\mathrm{B}}|-|\Delta_{\mathrm{F}}|+LG(g)} \end{matrix} \right\},\ T_0 < \min\left\{2^k, 2^{n+1}\right\}, \\
&T_1 = 2^{c_{\mathrm{B}}+c_{\mathrm{F}}+LG(g)},\ T_2 = 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|+LG(g)},\ T_3 = 2^{k-g}, \\
&T_{tot} = (T_0 + (T_1 + T_2)\, C_{E'} + T_3)\, C_E,\ T_{tot} < 2^k, \\
&M_{tot} = \min\left\{2^{c_{\mathrm{B}}+c_{\mathrm{F}}+LG(g)}, 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|}\right\},\ M_{tot} < 2^k,
\end{aligned} \tag{14}$$

where $LG(g) = \log_2(g) - 0.53$.

B Encoding the MDS Matrices

Proposition 4. *For $M : \mathbb{F}_2^{q \cdot c} \rightarrow \mathbb{F}_2^{q \cdot c}$, where M is an MDS matrix and $Y = M(X)$, the following constraints encode all valid transitions for deterministic truncated differential trails through M:*

$$\begin{aligned}
&MDS(AX, DX, AY, DY) := \\
&\textit{if} \sum_{i=0}^{q-1} AX[i] = 0 \ \textit{then}\ AY[0] = AY[1] = \cdots = AY[q-1] = 0 \\
&\textit{elseif} \sum_{i=0}^{q-1} AX[i] = 1 \ \textit{then}\ AY[0] = AY[1] = \cdots = AY[q-1] = 1 \\
&\textit{elseif} \sum_{i=0}^{q-1} AX[i] = 2 \ \wedge \sum_{i=0}^{q-1} DX[i] < 0 \ \textit{then}\ AY[0] = AY[1] = \cdots = AY[q-1] = 2 \\
&\textit{else}\ AY[0] = AY[1] = \cdots = AY[q-1] = 3 \ \textit{endif}
\end{aligned}$$

Application to SKINNY

C Specification of the SKINNY family of tweakable block ciphers

The SKINNY family of tweakable block ciphers was introduced by Beierle et al. in CRYPTO 2016 [3]. Let n and t denote the block and tweakey sizes, respectively. The SKINNY family has six main members. We use SKINNY-n-t to represent a member of SKINNY family block ciphers with n-bit block size and t-bit tweakey size. There are two proposed block sizes, $n \in \{64, 128\}$, and for each block size there are three tweakey sizes available, $t \in \{n, 2n, 3n\}$. The internal state of SKINNY can be viewed as a 4×4 array of cells. Depending on the tweakey size, the tweakey state can be viewed as a z 4×4 array of cells, where $z = \frac{t}{n} \in \{1, 2, 3\}$. We use $TK1, TK2$, and $TK3$ to denote the tweakey arrays. The cell size is 4 (or 8) bits when $n = 64$ (resp. $n = 128$).

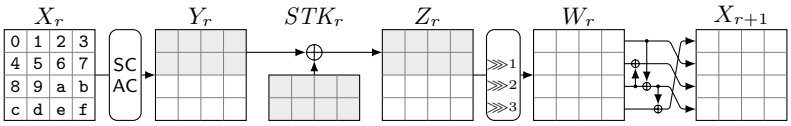

Fig. 5: Round function of SKINNY

As illustrated in Figure 5, each round of SKINNY applies five basic operations to the internal state: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR), and MixColumns (MC). The SC operation applies a 4-bit (or an 8-bit) S-box on each cell. AC combines the round constant with the internal state using the bitwise exclusive-or (XOR). In ART layer, the cells in the first and the second rows of subtweakey are XORed to the corresponding cells in the internals state. SR applies a permutation P on the position of the state cells, where $P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12]$. MC multiplies each column of the internal state by a non-MDS matrix M. M and its inverse are as follows:

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

Figure 5 represents the variables we use to denote the internal states of SKINNY after r rounds. We also use ΔX_r (ΓX_r) to represent the difference (resp. linear mask) of state X_r. To denote the ith cell of state X_r we use $X_r[i]$, where $0 \leq i \leq 15$. STK_r indicates the subtweakey after r rounds, and $ETK_r =$

$\texttt{MC} \circ \texttt{SR}(STK_r)$ which is called the equivalent subtweakey in round r. Figure 6 shows the relation between STK_r, ETK_r.

The tweakey schedule of SKINNY divides the master tweakey into z tweakey arrays $(TK1, \ldots, TKz)$ of lengths n bits each, where $z \in \{1, 2, 3\}$. Then, each tweakey array follows an independent schedule. The subtweakey of the ith round is generated as follows:

$$\begin{cases} STK_r = TK1_r & \text{if } t = n \\ STK_r = TK1_r \oplus TK2_r & \text{if } t = 2n \\ STK_r = TK1_r \oplus TK2_r \oplus TK3_r & \text{if } t = 3n, \end{cases} \tag{15}$$

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, such that $TKm_r[n] \leftarrow TKm_{r-1}[h(n)]$ for all $0 \leq n \leq 15$, and $m \in \{1, 2, 3\}$. Next, an LFSR is applied to each cell of the first and the second rows of $TK2_r$ and $TK3_r$. For more details on the specification of SKINNY we refer the reader to [3].

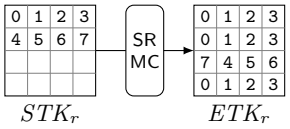

Fig. 6: Relation between the subtweakey and the equivalent subtweakey

D Encoding the Matrix of SKINNY

Suppose that $Y = M(X)$, where $X, Y \in \mathbb{F}_2^{4s}$, and M is the matrix of SKINNY. Morever, we compactly represent the constraint encoding the XOR operation $Y[k] = X[i] \oplus X[j]$, by $\mathit{XOR}(\texttt{AX}[i], \texttt{DX}[i], \texttt{AX}[j], \texttt{DX}[j], \texttt{AY}[k], \texttt{DY}[k])$. The following CP constraints encode the valid transitions for deterministic truncated differential trails through M:

$$\mathit{Mdiff}(\texttt{AX}, \texttt{DX}, \texttt{AY}, \texttt{DY}) := \begin{cases} \texttt{AY}[1] = \texttt{AX}[0] \ \wedge \ \texttt{DY}[1] = \texttt{DX}[0] \ \wedge \\ \mathit{XOR}(\texttt{AX}[1], \texttt{DX}[1], \texttt{AX}[2], \texttt{DX}[2], \texttt{AY}[2], \texttt{DY}[2]) \ \wedge \\ \mathit{XOR}(\texttt{AX}[0], \texttt{DX}[0], \texttt{AX}[2], \texttt{DX}[2], \texttt{AY}[3], \texttt{DY}[3]) \ \wedge \\ \mathit{XOR}(\texttt{AY}[3], \texttt{DY}[3], \texttt{AX}[3], \texttt{DX}[3], \texttt{AY}[0], \texttt{DY}[0]) \end{cases}$$

Assuming that $Y = M^{-1}(X)$, we use the following constraints to encode the propagation of deterministic truncated differential trails through M^{-1}:

$$\mathit{Minvdiff}(\texttt{AX}, \texttt{DX}, \texttt{AY}, \texttt{DY}) := \begin{cases} \texttt{AY}[0] = \texttt{AX}[1] \ \wedge \ \texttt{DY}[0] = \texttt{DX}[1] \ \wedge \\ \mathit{XOR}(\texttt{AX}[1], \texttt{DX}[1], \texttt{AX}[3], \texttt{DX}[3], \texttt{AY}[2], \texttt{DY}[2]) \ \wedge \\ \mathit{XOR}(\texttt{AX}[0], \texttt{DX}[0], \texttt{AX}[3], \texttt{DX}[3], \texttt{AY}[3], \texttt{DY}[3]) \ \wedge \\ \mathit{XOR}(\texttt{AY}[2], \texttt{DY}[2], \texttt{AX}[2], \texttt{DX}[2], \texttt{AY}[1], \texttt{DY}[1]) \end{cases}$$

We also use the following constraints to model the propagation of 0-1 differences with probability one through M, and M^{-1}:

$$\mathit{Mdiff}_1(\mathtt{AX},\mathtt{AY}) \coloneqq \begin{cases} \mathtt{AY}[1] = \mathtt{AX}[0] \ \wedge \ \mathit{XOR}_1(\mathtt{AX}[1],\mathtt{AX}[2],\mathtt{AY}[2]) \ \wedge \\ \mathit{XOR}_1(\mathtt{AX}[0],\mathtt{AX}[2],\mathtt{AY}[3]) \ \wedge \ \mathit{XOR}_1(\mathtt{AY}[3],\mathtt{AX}[3],\mathtt{AY}[0]) \end{cases}$$

$$\mathit{Minvdiff}_1(\mathtt{AX},\mathtt{AY}) \coloneqq \begin{cases} \mathtt{AY}[0] = \mathtt{AX}[1] \ \wedge \ \mathit{XOR}_1(\mathtt{AX}[1],\mathtt{AX}[3],\mathtt{AY}[2]) \ \wedge \\ \mathit{XOR}_1(\mathtt{AX}[0],\mathtt{AX}[3],\mathtt{AY}[3]) \ \wedge \ \mathit{XOR}_1(\mathtt{AY}[2],\mathtt{AX}[2],\mathtt{AY}[1]) \end{cases}$$

Let $\mathtt{LX}[i]$ be the actual c-bit value for the linear mask of state variable $X[i] \in \mathbb{F}_2^c$ and, as before, $\mathtt{AX}[i]$ denotes the activeness pattern of $X[i]$. We also define dummy variables $\mathtt{D}$, and $\mathtt{LD}$, such that $\mathtt{D} \in \{0,1,2,3\}$, and $\mathtt{LD} \in \{-2,-1,0,\ldots,2^c-1\}$. Then, we use the following constraints to encode the propagation of deterministic truncated linear trails through M, and M^{-1}:

$$\mathit{Mlin}(\mathtt{AX},\mathtt{LX},\mathtt{AY},\mathtt{LY}) \coloneqq \begin{cases} \mathtt{AY}[0] = \mathtt{AX}[3] \ \wedge \ \mathtt{LY}[0] = \mathtt{LX}[3] \ \wedge \\ \mathtt{AY}[2] = \mathtt{AX}[1] \ \wedge \ \mathtt{LY}[2] = \mathtt{LX}[1] \ \wedge \\ \mathit{XOR}(\mathtt{AX}[1],\mathtt{LX}[1],\mathtt{AX}[2],\mathtt{LX}[2],\mathtt{D},\mathtt{LD}) \ \wedge \\ \mathit{XOR}(\mathtt{D},\mathtt{LD},\mathtt{AX}[0],\mathtt{LX}[0],\mathtt{AY}[1],\mathtt{LY}[1]) \ \wedge \\ \mathit{XOR}(\mathtt{D},\mathtt{LD},\mathtt{AX}[3],\mathtt{LX}[3],\mathtt{AY}[3],\mathtt{LY}[3]) \end{cases}$$

$$\mathit{Minvlin}(\mathtt{AX},\mathtt{LX},\mathtt{AY},\mathtt{LY}) \coloneqq \begin{cases} \mathtt{AY}[1] = \mathtt{AX}[2] \ \wedge \ \mathtt{LY}[1] = \mathtt{LX}[2] \ \wedge \\ \mathtt{AY}[3] = \mathtt{AX}[0] \ \wedge \ \mathtt{LY}[3] = \mathtt{LX}[0] \ \wedge \\ \mathit{XOR}(\mathtt{AX}[0],\mathtt{LX}[0],\mathtt{AX}[3],\mathtt{LX}[3],\mathtt{D},\mathtt{LD}) \ \wedge \\ \mathit{XOR}(\mathtt{D},\mathtt{LD},\mathtt{AX}[1],\mathtt{LX}[1],\mathtt{AY}[0],\mathtt{LY}[0]) \ \wedge \\ \mathit{XOR}(\mathtt{D},\mathtt{LD},\mathtt{AX}[2],\mathtt{LX}[2],\mathtt{AY}[2],\mathtt{LY}[2]) \end{cases}$$

Let $\mathtt{KX}[i]$, and $\mathtt{KY}[i]$ be binary variables to indicate whether the value of $X[i]$, and $Y[i]$ are needed, respectively. Furthemore, assume that $\mathtt{D}$ is a binary variable. We use the following constraints to model the matrix of SKINNY in the guess-and-determine module:

$$\begin{aligned} \mathit{Mdata}(\mathtt{KX},\mathtt{KY}) &\coloneqq \begin{cases} \mathtt{KY}[0] = \mathtt{KX}[3] \wedge \mathtt{KY}[2] = \mathtt{KX}[1] \wedge \mathit{XOR}_1(\mathtt{KX}[1],\mathtt{KX}[2],\mathtt{D}) \wedge \\ \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[0],\mathtt{KY}[1]) \wedge \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[3],\mathtt{KY}[3]) \end{cases} \\ \mathit{Minvdata}(\mathtt{KX},\mathtt{KY}) &\coloneqq \begin{cases} \mathtt{KY}[1] = \mathtt{KX}[2] \wedge \mathtt{KY}[3] = \mathtt{KX}[0] \wedge \mathit{XOR}_1(\mathtt{KX}[0],\mathtt{KX}[3],\mathtt{D}) \wedge \\ \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[1],\mathtt{KY}[0]) \wedge \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[2],\mathtt{KY}[2]) \end{cases} \end{aligned} \tag{16}$$

E The COP Sub-Modules for the ID Attack on SKINNY

Algorithm 6: CSP_{L} for lower trail in distinguisher of SKINNY

Input: $CSP_{DTK}.\texttt{var}$ and the integer numbers $r_{\text{B}}, r_{\text{U}}, r_{\text{L}}$
Output: CSP_{U}

$r_{\text{off}} \leftarrow r_{\text{B}} + r_{\text{U}}$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\texttt{var} \leftarrow CSP_{DTK}.\texttt{var}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AXL}_r[i] \in \{0,1,2,3\} :\ 0 \le r \le r_{\text{L}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DXL}_r[i] \in \{-2,\dots,2^c-1\} :\ 0 \le r \le r_{\text{L}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AYL}_r[i] \in \{0,1,2,3\} :\ 0 \le r \le r_{\text{L}}-1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DYL}_r[i] \in \{-2,\dots,2^c-1\} :\ 0 \le r \le r_{\text{L}}-1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{AZL}_r[i] \in \{0,1,2,3\} :\ 0 \le r \le r_{\text{L}}-1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DZL}_r[i] \in \{-2,\dots,2^c-1\} :\ 0 \le r \le r_{\text{L}}-1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\texttt{con} \leftarrow \sum_{i}^{15} \texttt{AXL}_{r_{\text{L}}}[i] + \sum_{i=0}^{15} \texttt{DTK1}[i] + \sum_{i=0}^{15} \texttt{DTK2}_0 + \sum_{i=0}^{15} \texttt{DTK3}_0[i] \ge 1$;
for $r = 0,\dots,r_{\text{L}}-1,\ i = 0,\dots,15$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{Link}(\texttt{AXL}_r[i],\texttt{DXL}_r[i]) \wedge \mathit{Link}(\texttt{AYL}_r[i],\texttt{DYL}_r[i]) \wedge \mathit{Link}(\texttt{AZL}_r[i],\texttt{DZL}_r[i])$;
for $r = 0,\dots,r_{\text{L}}-1,\ i = 0,\dots,3$ **do**
  $I_1 \leftarrow [\texttt{AXL}_{r+1}[i], \texttt{AXL}_{r+1}[i+4], \texttt{AXL}_{r+1}[i+8], \texttt{AXL}_{r+1}[i+12]]$;
  $I_2 \leftarrow [\texttt{DXL}_{r+1}[i], \texttt{DXL}_{r+1}[i+4], \texttt{DXL}_{r+1}[i+8], \texttt{DXL}_{r+1}[i+12]]$;
  $O_1 \leftarrow [\texttt{AZL}_r[P[i]], \texttt{AZL}_r[P[i+4]], \texttt{AZL}_r[P[i+8]], \texttt{AZL}_r[P[i+12]]]$;
  $O_2 \leftarrow [\texttt{DZL}_r[P[i]], \texttt{DZL}_r[P[i+4]], \texttt{DZL}_r[P[i+8]], \texttt{DZL}_r[P[i+12]]]$;
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{Minvdiff}(I_1, I_2, O_1, O_2)$;
for $r = 0,\dots,r_{\text{L}}-1,\ i = 0,\dots,7$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{XOR}(\texttt{AZL}_r[i], \texttt{DZL}_r[i], \texttt{ASTK}_{r_{\text{off}}+r}[i], \texttt{DSTK}_{r_{\text{off}}+r}[i], \texttt{AYL}_r[i], \texttt{DYL}_r[i])$;
  $\mathcal{M}.\texttt{con} \leftarrow (\texttt{AYL}_r[i+8] = \texttt{AZL}_r[i+8]) \wedge (\texttt{DYL}_r[i+8] = \texttt{DZL}_r[i+8])$;
for $r = 0,\dots,r_{\text{L}}-1,\ i = 0,\dots,15$ **do**
  $\mathcal{M}.\texttt{con} \leftarrow \mathit{S\text{-}box}(\texttt{AYL}_r[i], \texttt{AXL}_r[i])$;
return $\mathcal{M}$;

Algorithm 7: $\mathit{CSP}_{\mathrm{F}}^{dp}$ difference propagation through E_{F} for SKINNY

Input: $\mathit{CSP}_{DTK}.\mathtt{var}$, $\mathit{CSP}_{\mathrm{L}}.\mathtt{var}$ and the integer numbers $r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{B}}, r_{\mathrm{F}}$
Output: $\mathit{CSP}_{\mathrm{F}}^{dp}$

$r_{\mathrm{off}} \leftarrow r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}}$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{DTK}.\mathtt{var}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AXF}_r[i] \in \{0,1\} : \ 0 \le r \le r_{\mathrm{F}}, \ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AZF}_r[i] \in \{0,1\} : \ 0 \le r \le r_{\mathrm{F}} - 1, \ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{CF}_r[i] \in \{0,1\} : \ 0 \le r \le r_{\mathrm{F}} - 1, \ 0 \le i \le 19\}$;
for $i = 0, \ldots, 15$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AXL}_{r_{\mathrm{L}}}[i] \ge 1\ \mathit{then}\ \mathtt{AXF}_0[i] = 1\ \mathit{else}\ \mathtt{AXF}_0[i] = 0$;
for $r = 0, \ldots, r_{\mathrm{F}} - 1, \ i = 0, \ldots, 7$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_1(\mathtt{AXF}_r[i], \mathtt{ASTK}_{r_{\mathrm{off}}+r}[i], \mathtt{AZF}_r[i])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AZF}_r[i], \mathtt{ASTK}_{r_{\mathrm{off}}+r}[i], \mathtt{AXF}_r[i], \mathtt{CF}_r[i])$;
    $\mathcal{M}.\mathtt{con} \leftarrow (\mathtt{AZF}_r[i+8] = \mathtt{AXF}_r[i+8])$;
for $r = 0, \ldots, r_{\mathrm{F}} - 1, \ i = 0, \ldots, 3$ **do**
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{Mdiff}_1\left(\begin{pmatrix} \mathtt{AZF}_r[P[i]] \\ \mathtt{AZF}_r[P[i+4]] \\ \mathtt{AZF}_r[P[i+8]] \\ \mathtt{AZF}_r[P[i+12]] \end{pmatrix}, \begin{pmatrix} \mathtt{AXF}_{r+1}[i] \\ \mathtt{AXF}_{r+1}[i+4] \\ \mathtt{AXF}_{r+1}[i+8] \\ \mathtt{AXF}_{r+1}[i+12] \end{pmatrix}\right)$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AZF}_r[P[i+8]], \mathtt{AXF}_{r+1}[i+8], \mathtt{AZF}_r[P[i+4]], \mathtt{CF}_r[i+8])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AXF}_{r+1}[i+4], \mathtt{AXF}_{r+1}[i+12], \mathtt{AZF}_r[P[i+8]], \mathtt{CF}_r[i+12])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AXF}_{r+1}[i], \mathtt{AXF}_{r+1}[i+12], \mathtt{AZF}_r[P[i+12]], \mathtt{CF}_r[i+16])$;
return $\mathcal{M}$;

Algorithm 8: $\mathit{CSP}_{\mathrm{F}}^{gd}$ guess-and-determine through E_{F} for SKINNY

Input: $\mathit{CSP}_{\mathrm{L}}.\mathtt{var}$, $\mathit{CSP}_{\mathrm{F}}^{dp}$ and the integer numbers $r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{L}}, r_{\mathrm{F}}$
Output: $\mathit{CSP}_{\mathrm{F}}^{gd}$

$r_{\mathrm{off}} \leftarrow r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}}$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{\mathrm{L}}.\mathtt{var} \cup \mathit{CSP}_{\mathrm{F}}^{dp}.\mathtt{var}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KDXF}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{F}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{con} \leftarrow \{\mathtt{KDXF}_r[i] \le \mathtt{AXF}_r[i] :\ 0 \le r \le r_{\mathrm{F}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KDZF}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{F}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{con} \leftarrow \{\mathtt{KDZF}_r[i] \le \mathtt{AZF}_r[i] :\ 0 \le r \le r_{\mathrm{F}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KXF}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{F}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{KYF}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{F}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{IKF}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{F}} - 1,\ 0 \le i \le 15\}$;
for $i = 0, \ldots, 15$ **do**
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AXL}_{r_{\mathrm{L}}}[i] = 1\ \mathit{then}\ \mathtt{KDXF}_0[i] = 1\ \mathit{else}\ \mathtt{KDXF}_0[i] = 0$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AXL}_{r_{\mathrm{L}}}[i] = 1\ \mathit{then}\ \mathtt{KXF}_0[i] = 1\ \mathit{else}\ \mathtt{KXF}_0[i] = 0$;
for $r = 0, \ldots, r_{\mathrm{F}} - 1,\ i = 0, \ldots, 15$ **do**
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{S\text{-}box}_{gd}(\mathtt{KXF}_r[i], \mathtt{KYF}_r[i], \mathtt{KDXF}_r[i])$;
for $r = 0, \ldots, r_{\mathrm{F}} - 1,\ i = 0, \ldots, 7$ **do**
  $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{KDZF}_r[i] \ge \mathtt{KDXF}_r[i]$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathtt{KDZF}_r[i+8] = \mathtt{KDXF}_r[i+8]$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CF}_r[i] = 1\ \mathit{then}\ \mathtt{KDZF}_r[i] = 1$;
  $\mathcal{M}.\mathtt{con} \leftarrow (\mathtt{IKF}_r[i] = \mathtt{KYF}_r[i] \wedge \mathtt{IKF}_r[i+8] = 0)$;
for $r = 0, \ldots, r_{\mathrm{F}} - 1,\ i = 0, \ldots, 3$ **do**
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDZF}_r[P[i]] = 1\ \mathit{then}\ \mathtt{KDXF}_{r+1}[i+4] = \mathtt{AXF}_{r+1}[i+4]$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDZF}_r[P[i+4]] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDXF}_{r+1}[i+4] = \mathtt{AXF}_{r+1}[i+4] \wedge \\ \mathtt{KDXF}_{r+1}[i+8] = \mathtt{AXF}_{r+1}[i+8] \wedge \\ \mathtt{KDXF}_{r+1}[i+12] = \mathtt{AXF}_{r+1}[i+12] \end{pmatrix}$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDZF}_r[P[i+8]] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDXF}_{r+1}[i+4] = \mathtt{AXF}_{r+1}[i+4] \wedge \\ \mathtt{KDXF}_{r+1}[i+12] = \mathtt{AXF}_{r+1}[i+12] \end{pmatrix}$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{KDZF}_r[P[i+12]] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDXF}_{r+1}[i] = \mathtt{AXF}_{r+1}[i] \wedge \\ \mathtt{KDXF}_{r+1}[i+12] = \mathtt{AXF}_{r+1}[i+12] \end{pmatrix}$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CF}_r[i+8] = 1\ \mathit{then} \begin{pmatrix} \mathtt{KDXF}_{r+1}[i+4] = \mathtt{AXF}_{r+1}[i+4] \wedge \\ \mathtt{KDXF}_{r+1}[i+12] = \mathtt{AXF}_{r+1}[i+12] \wedge \\ \mathtt{KDXF}_{r+1}[i+8] = 1 \end{pmatrix}$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CF}_r[i+12] = 1\ \mathit{then}\ (\mathtt{KDXF}_{r+1}[i+4] = 1 \wedge \mathtt{KDXF}_{r+1}[i+12] = 1)$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{CF}_r[i+16] = 1\ \mathit{then} (\mathtt{KDXF}_{r+1}[i] = 1 \wedge \mathtt{KDXF}_{r+1}[i+12] = 1)$;
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{Mdata} \left(\begin{pmatrix} \mathtt{KYF}_r[P[i]] \\ \mathtt{KYF}_r[P[i+4]] \\ \mathtt{KYF}_r[P[i+8]] \\ \mathtt{KYF}_r[P[i+12]] \end{pmatrix}, \begin{pmatrix} \mathtt{KXF}_{r+1}[i] \\ \mathtt{KXF}_{r+1}[i+4] \\ \mathtt{KXF}_{r+1}[i+8] \\ \mathtt{KXF}_{r+1}[i+12] \end{pmatrix} \right)$;
return $\mathcal{M}$;

F Impossible Differential Attack on SKINNY

As stated in Section 2.1, Equation 2 only gives an approximation for the time complexity of *Guess-and-Filter* step in the ID attack. Thus, after finding the nearly optimum attack using our automated tool, we provide a more detailed complexity analysis by computing the complexity of each step in *Guess-and-Filter* phase more precisely. We also use Lemma 1 in the complexity analysis of our ID attacks.

Lemma 1. *For a given S-box S, and any input/output difference $\delta_i, \delta_o \neq 0$, the equation $S(x \oplus \delta_i) \oplus S(x) = \delta_o$ has one solution on average that can be derived efficiently. In addition, the inverse of S has a similar property.*

F.1 17-round Impossible Differential Attack on SKINNY-n-n

This section details a 17-round attack on SKINNY-n-n that takes the 11-round impossible differential trail shown in Figure 7 and extends it by 3 rounds in both directions. Since there is no tweakey used before W_0', the plaintext P can be recovered by applying $\mathsf{MC}^{-1}, \mathsf{SR}^{-1}, \mathsf{AC}^{-1}$, and SC^{-1} layers on W_0'.

Pair Generation. The attacker should build 2^x structures at W_0' and evaluate all possible values in seven cells $W_0'[2, 4, 5, 7, 8, 10, 13]$ for each structure, while the other cells assume a fixed value. By using $2^{x+|\Delta_{\text{B}}|}$ plaintexts, we can have $2^{x+2|\Delta_{\text{B}}|-1}$ pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_{\text{B}}|-1-(n-|\Delta_{\text{F}}|)}$ pairs. Thus, $N = 2^{x+5c-1}$. This step needs a total of $2^{x+|\Delta_{\text{B}}|} = 2^{x+7c}$ encryption calls.

Guess-and-Filter. For each of the N pairs we perform the following steps:

a) *Satisfying round 17.* Calculate $\Delta X_{16}[11, 15]$ using the values of the ciphertext pairs. There is no requirement for any tweaked information to compute these cells here. We get $\Delta X_{16}[3] = \Delta X_{16}[11] = \Delta X_{16}[15]$ because of the MC operation on the active cells in the fourth column of W_{15}. Checking if $\Delta X_{17}[11] = \Delta X_{17}[15]$ will lead to a c-bit filter. From the knowledge of $\Delta Y_{16}[3]$ and $\Delta X_{16}[3]$, we can determine $Y_{16}[3]$ by applying Lemma 1. Thus, we can determine $STK_{16}[3]$ (due to $STK_{16}[3] = Z_{16}[3] \oplus Y_{16}[3]$). Now, we can calculate $\Delta X_{16}[13]$ from ciphertext values. Based on the MC operation on the active cells in the second column of W_{15}, we have $\Delta X_{16}[13] = \Delta X_{16}[5] = \Delta X_{16}[1]$. Similarly, the knowledge of this information and Lemma 1 helps us to derive the tweakey cells $STK_{16}[1, 5]$. The time complexity of this step is N, and the number of tests left for the next step is $N2^{-c}$.

b) *Satisfying round 1.* From the knowledge of $STK_{16}[1, 3, 5]$ (from the previous step), we can uniquely determine $ETK[7, 10, 13]$. Therefore, we determine $\Delta Y_1[7, 10, 13]$. Due to the MC^{-1} operation on the active cells in the first column of X_2, we have $\Delta Y_1[7] = \Delta Y_1[10] = \Delta Y_1[13]$ that will lead to two $c-$bit filters. The time complexity of this step is $N2^{-c}$, and the number of tests left for the next step is $N2^{-3c}$.

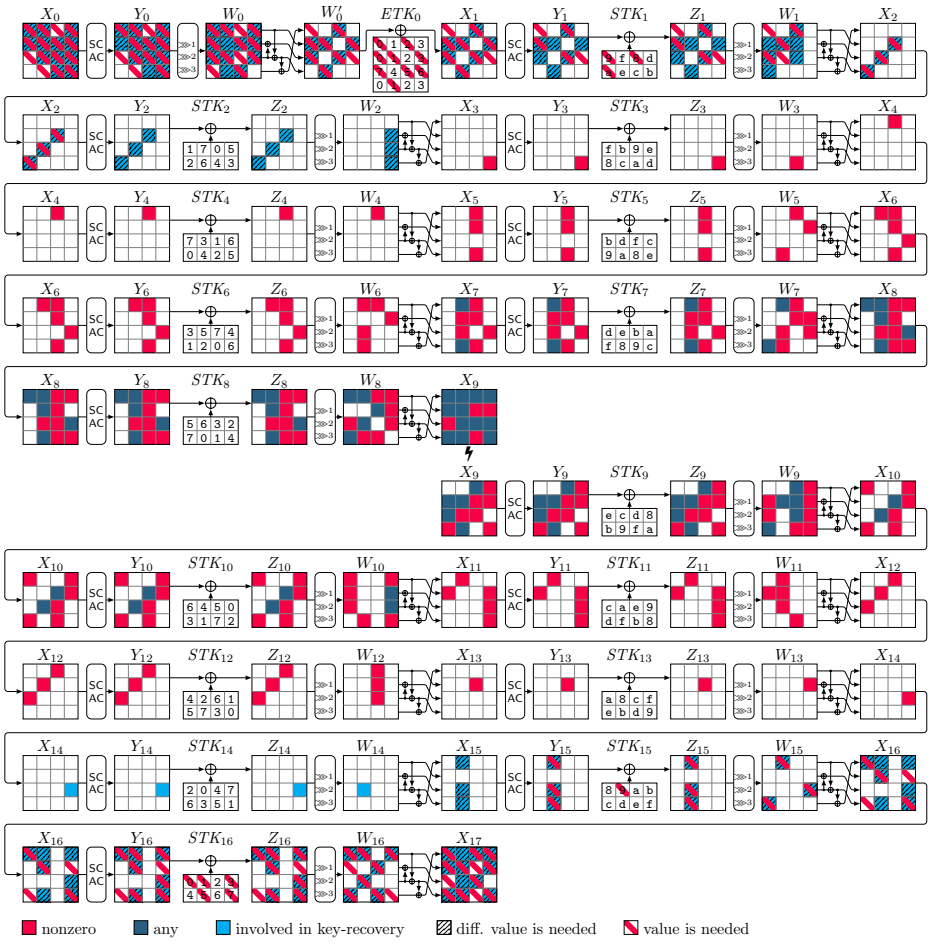

Fig. 7: ID attack on 17 rounds of SKINNY-n-n. $|k_B \cup k_F| = 10 \cdot c$, $c_B = 6 \cdot c$, $c_F = 6 \cdot c$, $\Delta_B = 7 \cdot c$, $\Delta_F = 7 \cdot c$.

c) *Satisfying round 17.* We guess $STK_{16}[0,7]$. Hence, we can compute Z_{15}, and ΔZ_{15} as shown in Figure 7. The time complexity of this step is $N2^{-c}$, and the number of tests left for the next step is $N2^{-c}$.

d) *Satisfying rounds 16 and 15.* We can calculate $\Delta X_{15}[9,13]$ using the value of $Z_{15}[9,13]$, and $\Delta Z_{15}[9,13]$. We have $\Delta X_{15}[1] = \Delta X_{15}[9] = \Delta X_{15}[13]$ because of the MC operation on the active cells in the second column of W_{14}. Checking if $\Delta X_{15}[9] = \Delta X_{15}[13]$ will lead to a c-bit filter. Given $\Delta X_{15}[1]$, we can determine $Y_{15}[1]$ and so $STK_{15}[1]$ by applying Lemma 1. Hence, we can compute ΔX_{14} as shown in Figure 7. The time complexity of this step is $N2^{-c}$, and the number of tests left for the next step is $N2^{-2c}$.

e) *Satisfying round 1.* From the knowledge of $STK_{16}[1,7]$ (from the previous steps), we can uniquely determine $ETK[5,8]$. Therefore, we determine $\Delta Y_1[5,8]$. Due to the MC^{-1} operation on the active cells in the third column of X_2, we have $\Delta Y_1[2] = \Delta Y_1[5] = \Delta Y_1[8]$. The equality $\Delta Y_1[5] = \Delta Y_1[8]$ will lead to a c-bit filter. From the knowledge of $\Delta Y_1[2]$ and $\Delta X_1[2]$, we can determine $X_1[2]$ by applying Lemma 1. Thus, we can derive $ETK[2]$ (duo to $ETK[2] = W'_0[2] \oplus X_1[2]$). We guess $ETK[11]$, and compute Y_1 and ΔY_1 as shown in Figure 7. The time complexity of this step is $N2^{-c}$, and the number of tests left for the next step is $N2^{-2c}$.

f) *Satisfying round 2.* Since we know the value of $STK_{15}[1]$, we can uniquely determine $STK_1[0]$. We guess $STK_1[4]$. Therefore, we can determine $\Delta Y_2[9,12]$. Due to the MC^{-1} operation on the active cells in the fourth column of X_3, we have $\Delta Y_2[6] = \Delta Y_2[9] = \Delta Y_2[12]$. The equality $\Delta Y_2[9] = \Delta Y_2[12]$ will lead to a c-bit filter. From the knowledge of $\Delta Y_2[6]$ and $\Delta X_2[6]$, we can determine $X_2[6]$ by applying Lemma 1. Therefore, we can determine the value of $STK_1[2]$ as $X_2[6] \oplus Y_1[6]$. Hence, we can compute ΔY_2 and thus, ΔX_3 as shown in Figure 7. The time complexity of this step is $N2^{-c}$, and the number of tests to verify the impossible distinguisher is $N2^{-2c}$.

Complexity. Analyzing N pairs has a time complexity of about $N\frac{1}{17}$ 17-round encryptions (it is dominated by step 1). The attack needs a data complexity of $D = N2^{n+1-|\Delta_{\text{B}}|-|\Delta_{\text{F}}|} = 2^{14c+1}g\ln 2$ ($N = 2^{c_{\text{B}}+c_{\text{F}}}g\ln 2$). The total time complexity is $T = D + N\frac{1}{17} + 2^{16c-g}$ (2^{16c-g} is related to *exhaustive search step*). Hence, to optimize the time complexity of the attack, we select $g = 5$ for $c = 4$, and $g = 15$ for $c = 8$. Thus, the data, time, and memory complexities of the attack on SKINNY-64-64 are $2^{58.79}$, $2^{59.90}$, and 2^{40}, respectively. The data, time, and memory complexities of the attack on SKINNY-128-128 are $2^{116.37}$, $2^{116.51}$, and 2^{80}, respectively.

F.2 19-round Impossible Differential Attack on SKINNY-n-$2n$

This part presents the details of our 19-round attack on SKINNY-n-$2n$. We extend the 11-round impossible differential trail illustrated in Figure 8 by 3 and 5 rounds in backward and forward directions, respectively.

Pair Generation. We should build 2^x structures at W'_0 and evaluate all possible values in seven cells $W'_0[2,4,5,7,8,10,13]$ for each structure, while the other cells assume a fixed value. By using 2^{x+7c} plaintexts, we can have $2^{x+14c-1}$ pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_{\text{B}}|-1-(n-|\Delta_{\text{F}}|)}$ pairs. In our 19-round attack $n = |\Delta_{\text{F}}|$, and so $N = 2^{x+2|\Delta_{\text{B}}|-1} = 2^{x+14c-1}$. This step needs 2^{x+7c} encryption calls.

Guess-and-Filter. For each of the N pairs we do the following steps:

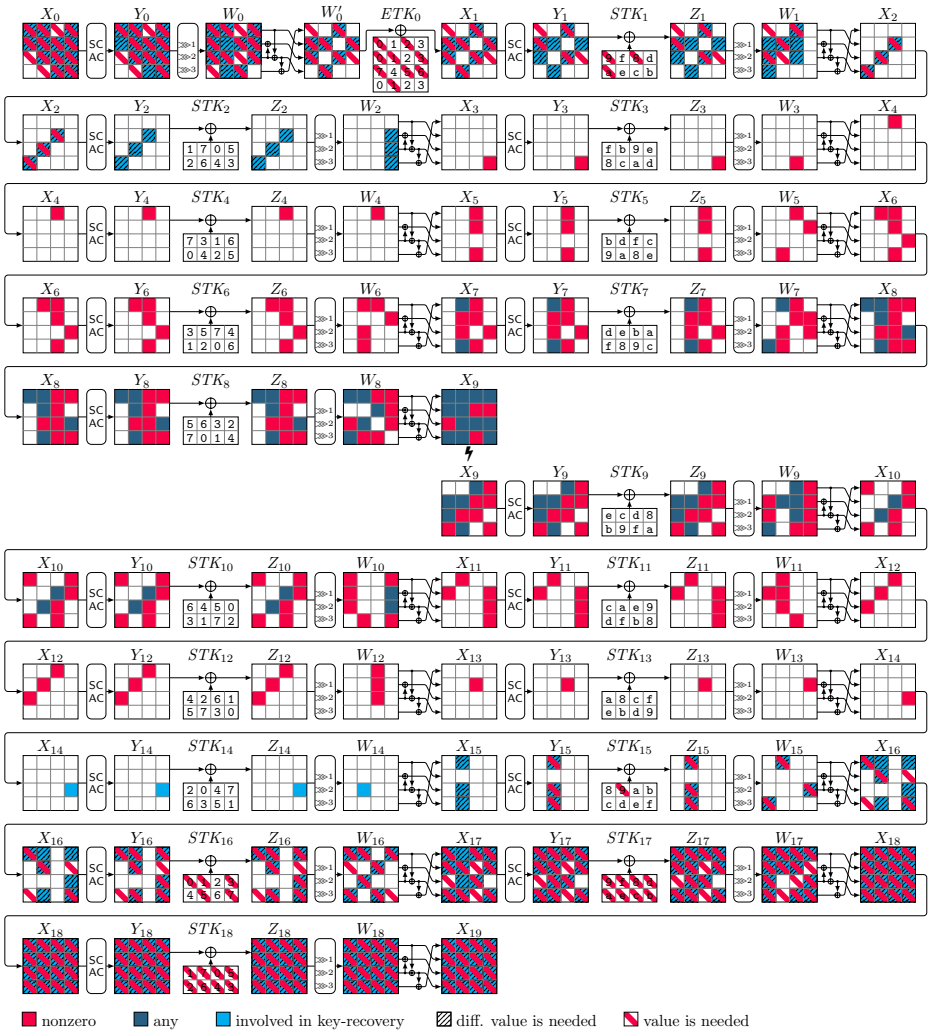

Fig. 8: ID attack on 19 rounds of SKINNY-n-$2n$, $|k_B \cup k_F| = 26 \cdot c$, $c_B = 6 \cdot c$, $c_F = 15 \cdot c$, $\Delta_B = 7 \cdot c$, $\Delta_F = 16 \cdot c$

a) *Satisfying round 19.* We can calculate $\Delta X_{18}[11, 15]$ using the values of the ciphertext pairs. We get $\Delta X_{18}[7] = \Delta X_{18}[11] \oplus \Delta X_{18}[15]$ because of the MC operation on the active cells in the fourth column of W_{17}. Given $\Delta X_{18}[7]$, we can determine $Y_{18}[7]$ by applying Lemma 1 and the knowledge of $\Delta Y_{18}[7]$. Now, we can determine $STK_{18}[7]$ (due to $STK_{18}[7] = Z_{18}[7] \oplus Y_{18}[7]$)). Similarly, due to the MC operation on the active cells in the third and the second column of W_{17}, we can also derive tweakey cells $STK_{18}[6]$ and $STK_{18}[1, 5]$, respectively. We guess $STK_{18}[0, 2, 3, 4]$. Hence, we can calculate Z_{17} and

ΔZ_{17} as shown in Figure 8. The time complexity of this step is $N2^{4c}$, and the number of tests left for the next step is $N2^{4c}$.

b) *Satisfying round 18.* We can calculate $\Delta X_{17}[15]$ using the value of $Z_{17}[15]$, and $\Delta Z_{17}[15]$. We have $\Delta X_{17}[15] = \Delta X_{17}[3] = \Delta X_{17}[7]$ because of the MC operation on the active cells in the fourth column of W_{16}. Given $\Delta X_{17}[3,7]$, we can determine $Y_{17}[3,7]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{17}[3,7]$. Now, we can determine $STK_{17}[3,7]$ (due to $Y_{17}[3] = Z_{17}[3] \oplus STK_{17}[3]$, and $Y_{17}[7] = Z_{17}[7] \oplus STK_{17}[7]$). Similarly, due to the MC operation on the active cells in the second and the first column of W_{16}, we can also derive tweakey cells $STK_{17}[1,5]$, and $STK_{17}[4]$. We guess $STK_{17}[2,6]$. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{6c}$.

c) *Satisfying round 17.* Based on the previous steps, we can compute the cells $Z_{16}[11,15]$, and $\Delta Z_{16}[11,15]$, and thus, $\Delta X_{16}[11,15]$ (there is no requirement for any tweaked information of STK_{16} to compute these cells here). On the other hand, We have $\Delta X_{16}[11] = \Delta X_{16}[15]$ due to the MC operation on the active cells in the fourth column of W_{15}. Checking if $\Delta X_{16}[11] = \Delta X_{16}[15]$ will lead to a c-bit filter. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{5c}$.

d) *Satisfying round 18.* We guess $STK_{17}[0]$. Hence, we can calculate Z_{16} and ΔZ_{16} as shown in Figure 8. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{6c}$.

e) *Satisfying round 17.* We calculate $\Delta X_{16}[15]$ using the values of $Z_{16}[15]$, and $\Delta Z_{16}[15]$. We have $\Delta X_{16}[15] = \Delta X_{16}[3]$ because of the MC operation on the active cells in the fourth column of W_{15}. Given $\Delta X_{16}[3]$, we can determine $Y_{16}[3]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{16}[3]$. Now, we can determine $STK_{16}[3]$ (due to $Y_{16}[3] = Z_{16}[3] \oplus STK_{16}[3]$). Similarly, due to the MC operation on the active cells in the second column of W_{15}, we can also derive tweakey cells $STK_{16}[1,5]$. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{6c}$.

f) *Satisfying round 1.* Since the values of $STK_{18}[0,3,7]$, and $STK_{16}[1,3,5]$ are known from the previous steps, we can uniquely determine $ETK[7,10,13]$. Therefore, we can determine $\Delta Y_1[7,10,13]$. Due to the MC^{-1} operation on the active cells in the first column of X_2, we have $\Delta Y_1[7] = \Delta Y_1[10] = \Delta Y_1[13]$ and this will lead to two $c-$bit filters. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{4c}$.

g) *Satisfying round 17.* We guess $STK_{16}[0,7]$. Hence, we can compute Z_{15}, and ΔZ_{15} as shown in Figure 8. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{6c}$.

h) *Satisfying rounds 16 and 15.* We calculate $\Delta X_{15}[9,13]$ using the values of $Z_{15}[9,13]$, and $\Delta Z_{15}[9,13]$. We have $\Delta X_{15}[1] = \Delta X_{15}[9] = \Delta X_{15}[13]$ because of the MC operation on the active cells in the second column of W_{14}. Checking if $\Delta X_{15}[9] = \Delta X_{15}[13]$ will lead to a c-bit filter. Given $\Delta X_{15}[1]$, we can determine $Y_{15}[1]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{15}[1]$. Now, we can determine $STK_{15}[1]$ (due to $Y_{15}[1] = Z_{15}[1] \oplus STK_{15}[1]$). Hence,

we can compute ΔX_{14} as shown in Figure 8. The time complexity of this step is $N2^{6c}$, and the number of tests left for the next step is $N2^{5c}$.

i) *Satisfying round 1.* From the knowledge of $STK_{18}[0,1]$, and $STK_{16}[1,7]$, we can uniquely determine $ETK[5,8]$. Therefore, we determine $\Delta Y_1[5,8]$. Due to the MC^{-1} operation on the active cells in the third column of X_2, we have $\Delta Y_1[2] = \Delta Y_1[5] = \Delta Y_1[8]$. The equality $\Delta Y_1[5] = \Delta Y_1[8]$ will lead to a c-bit filter. Since we know the values of $\Delta Y_1[2]$ and $\Delta X_1[2]$, we can determine $X_1[2]$ by applying Lemma 1. Hence, we derive the value of $ETK[2]$. The time complexity of this step is $N2^{5c}$, and the number of tests left for the next step is $N2^{4c}$.

j) *Satisfying round 1.* We know the values of $ETK[0,2,4,5,7,8,10,13]$ from the previous steps. We guess $ETK[11]$, and computes Y_1 and ΔY_1 as shown in Figure 8. The time complexity of this step is $N2^{5c}$, and the number of tests left for the next step is $N2^{5c}$.

k) *Satisfying rounds 2 and 3.* We know the values of $STK_{17}[0]$, and $STK_{15}[1]$ from the previous steps, so we can uniquely determine $STK_1[0]$. We guess $STK_1[4]$. Therefore, we can determine $\Delta Y_2[9,12]$. Due to the MC^{-1} operation on the active cells in the fourth column of X_3, we have $\Delta Y_2[6] = \Delta Y_2[9] = \Delta Y_2[12]$. The equality $\Delta Y_2[9] = \Delta Y_2[12]$ will lead to a c-bit filter. Since we know the values of $\Delta Y_2[6]$ and $\Delta X_2[6]$, we can determine $X_2[6]$ and so $Z_1[2]$ by applying Lemma 1. Therefore, we can determine the value of $STK_1[2]$ as $Y_1[2] \oplus Z_1[2]$. Hence, we can compute ΔX_3 as shown in Figure 8. The time complexity of this step is $N2^{6c}$, and the number of tests to verify the impossible distinguisher is $N2^{5c}$.

Complexity. Analyzing N pairs has a time complexity of about $N2^{6c}\frac{6}{19}$ 19-round encryptions. The attack needs a data complexity of $D = N2^{n+1-|\Delta_{\mathrm{B}}|-|\Delta_{\mathrm{F}}|} = 2^{15c+1}g\ln 2$ ($N = 2^{c_{\mathrm{B}}+c_{\mathrm{F}}}g\ln 2$). The total time complexity is $T = D + N2^{6c}\frac{6}{19} + 2^{32c-g}$. Hence, to optimize the time complexity of the attack, we select $g = 21$ for $c = 4$ and $g = 42$ for $c = 8$. Thus, the data, time, and memory complexities of the attack on SKINNY-64-128 are $2^{60.86}$, $2^{110.34}$, and 2^{104}, respectively. The data, time, and memory complexities of the attack on SKINNY-128-256 are $2^{117.86}$, $2^{219.23}$, and 2^{208}, respectively.

F.3 21-round Impossible Differential Attack on SKINNY-n-3n

An 11-round distinguisher is placed between Round 6 to Round 16 to attack 21 rounds of SKINNY-n-3n (see Figure 9). In this attack $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 42c$, $|\Delta_{\mathrm{B}}| = |\Delta_{\mathrm{F}}| = 16c$, $c_{\mathrm{B}} = c_{\mathrm{F}} = 15c$.

Pair Generation. We define a structure as the set of inputs that can take values in W_0'. By using 2^m plaintexts, we can have 2^{2m-1} pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{2m-1}$. This step needs 2^m encryption calls.

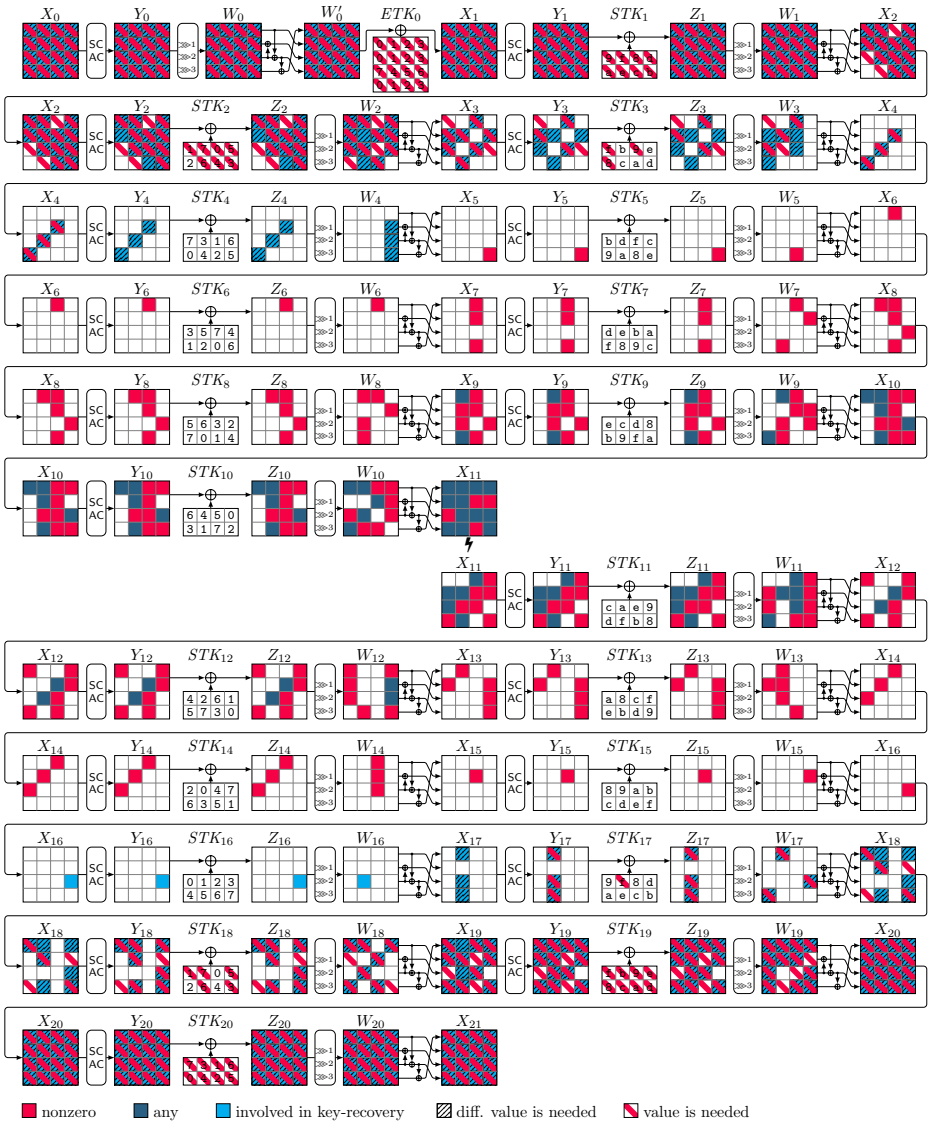

Fig. 9: ID attack on 21 rounds of SKINNY-n-3n. $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 42 \cdot c$, $c_{\mathrm{B}} = 15 \cdot c$, $c_{\mathrm{F}} = 15 \cdot c$, $\Delta_{\mathrm{B}} = 16 \cdot c$, $\Delta_{\mathrm{F}} = 16 \cdot c$.

Guess-and-Filter. For each of the N pairs we do the following steps:

a) *Satisfying round 21.* We guess $STK_{20}[0-7]$ and compute W_{19} as shown in Figure 9. Here, we have four c-bit filters based on the W_{19} state. The time complexity of this step is $N2^{8c}$, and the number of tests left for the next step is $N2^{4c}$.

b) *Satisfying round 1.* We guess $ETK[0-3, 8-11]$ and compute Y_1, and ΔY_1 as shown in Figure 9. Due to MC^{-1} operation on the active cells in the first, the second, and the third columns of X_2, we should have $\Delta Y_1[0] = \Delta Y_1[7] = \Delta Y_1[10]$, $\Delta Y_1[1] = \Delta Y_1[11]$, and $\Delta Y_1[2] \oplus \Delta Y_1[8] = \Delta Y_1[15]$, respectively, that will lead to four c-bit filters. The time complexity of this step is $N2^{12c}$, and the number of tests left for the next step is $N2^{8c}$.

c) *Satisfying round 2.* We guess $STK_1[0, 1, 2, 6]$ and compute $Y_2[1, 4, 11, 14]$, and so $\Delta Y_2[1, 4, 11, 14]$. Due to MC^{-1} operation on the active cells in the second column of X_3, we have $\Delta Y_2[4] = \Delta Y_2[11]$, and $\Delta Y_2[1] = \Delta Y_2[11] \oplus \Delta Y_2[14]$ that will lead to two c-bit filters. The time complexity of this step is $N2^{12c}$, and the number of tests left for the next step is $N2^{10c}$.

d) *Satisfying round 2.* We guess $STK_1[3, 4, 5]$ and compute $Y_2[0, 3, 6, 9, 10]$, and so $\Delta Y_2[0, 3, 6, 9, 10]$. Due to MC^{-1} operation on the active cells in the first and the fourth columns of X_3, we have $\Delta Y_2[0] = \Delta Y_2[10]$, and $\Delta Y_2[3] = \Delta Y_2[6] = \Delta Y_2[9]$ that will lead to three c-bit filters. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{10c}$.

e) *Satisfying round 2.* We guess $STK_1[7]$. We can compute Y_2, and ΔY_2 as shown in Figure 9. The time complexity of this step is $N2^{11c}$, and the number of tests left for the next step is $N2^{11c}$.

f) *Satisfying round 20.* We calculate $\Delta X_{19}[15]$ using the value of $Z_{19}[15]$, and $\Delta Z_{19}[15]$. We have $\Delta X_{19}[3] = \Delta X_{19}[7] = \Delta X_{19}[15]$ because of the MC operation on the active cells in the last column of W_{18}. Given $\Delta X_{19}[3, 7]$, we can determine $Y_{19}[3, 7]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{19}[3, 7]$. Now, we can determine $STK_{19}[3, 7]$ (due to $Y_{19}[3] = Z_{19}[3] \oplus STK_{19}[3]$, and $Y_{19}[7] = Z_{19}[7] \oplus STK_{19}[7]$). Similarly, due to the MC operation on the active cells in the second and the first columns of W_{18}, we can also derive tweakey cells $STK_{19}[1, 5]$ and $STK_{19}[4]$, respectively. We guess $STK_{19}[2]$. Now, we can compute $\Delta X_{18}[11, 15]$ that will lead to a c-bit filter (due to MC operation of the fourth column of W_{17}). The time complexity of this step is $N2^{12c}$, and the number of tests left for the next step is $N2^{11c}$.

g) *Satisfying round 20.* We guess $STK_{19}[0, 6]$. Thus, we can compute Z_{18} and ΔZ_{18} as shown in Figure 9. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{13c}$.

h) *Satisfying round 20.* We calculate $\Delta X_{18}[15]$ using the value of $Z_{18}[15]$. We have $\Delta X_{18}[3] = \Delta X_{18}[15]$ because of the MC operation on the active cells in the fourth column of W_{17}. Given $\Delta X_{18}[3]$, we can determine $Y_{18}[3]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{18}[3]$. Now, we can determine $STK_{18}[3]$ (due to $Y_{18}[3] = Z_{18}[3] \oplus STK_{18}[3]$). Similarly, due to the MC operation on the active cells in the second column of W_{17}, we can also derive tweakey cells $STK_{18}[1, 5]$. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{13c}$.

i) *Satisfying round 3.* We know the values of $STK_{20}[0, 3, 7]$, $STK_{18}[1, 3, 5]$, and $STK_0[5, 6, 7]$ from the previous steps. Therefore, we will have $STK_2[1, 3, 5]$. These values allow us to compute $W_2[1, 3, 6, 9, 10]$ and thus, $Y_3[7, 10, 13]$, and $\Delta Y_3[7, 10, 13]$. Due to MC^{-1} operation on the active cells in the first column of X_4, we have $\Delta Y_3[7] = \Delta Y_3[10] = \Delta Y_3[13]$ that will lead to two c-bit filters.

The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{11c}$.

j) *Satisfying round 19.* We guess $STK_{18}[0,7]$. We compute Z_{17} and ΔZ_{17} as shown in Figure 9. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{13c}$.

k) *Satisfying rounds 18 and 17.* We calculate $\Delta X_{17}[9,13]$ using the values of $Z_{17}[9,13]$. We have $\Delta X_{17}[9] = \Delta X_{17}[13] = \Delta X_{17}[1]$ because of the MC operation on the active cells in the second column of W_{16}. The equality $\Delta X_{17}[9] = \Delta X_{17}[13]$ will lead to a c-bit filter. Also, since we know $\Delta X_{17}[1]$, thus, we can determine $Y_{17}[1]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{17}[1]$. Thus, we can determine $STK_{17}[1]$ (due to $Y_{17}[1] = Z_{17}[1] \oplus STK_{17}[1]$). Compute ΔX_{16} as shown in Figure 9. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{12c}$.

k) *Satisfying round 3.* We know the values of $STK_{20}[0,1]$, $STK_{18}[1,7]$, and $STK_0[3,7]$ from the previous steps. Therefore, we will have $STK_2[1,7]$. We guess $STK_2[2]$. These values enables us to compute $W_2[1,2,4,8,10,14]$ and thus, $Y_3[7,10,13]$, and $\Delta Y_3[2,5,8]$. Due to MC^{-1} operation on the active cells in the third column of X_4, we have $\Delta Y_3[2] = \Delta Y_3[5] = \Delta Y_3[8]$ that will lead to two c-bit filters. The time complexity of this step is $N2^{13c}$, and the number of tests left for the next step is $N2^{11c}$.

l) *Satisfying round 3.* We can determine the values of $STK_2[0,1,3,5,7]$, from the knowledge of $STK_{20}[0-3,6,7]$, $STK_{18}[0,1,3,5,7]$, and $STK_0[1-3,5-7]$. We also know the value of $STK_2[2]$ from the previous step. We just guess $STK_2[6]$ and compute Y_3, and ΔY_3 as shown in Figure 9. The time complexity of this step is $N2^{12c}$, and the number of tests left for the next step is $N2^{12c}$.

m) *Satisfying round 4.* We guess $STK_3[4]$ to determine $X_4[9]$, $\Delta X_4[9]$ and thus, $\Delta Y_4[9]$. Due to MC^{-1} operation on the active cells in the fourth column of X_5, we have $\Delta Y_4[6] = \Delta Y_4[9]$. From the knowledge of $\Delta Y_4[6]$, we can determine $X_4[6]$ by applying Lemma 1 and the knowledge of the $\Delta X_4[6]$. Thus, we can determine the value of $STK_3[2]$ due to $X_4[6] = Y_3[2] \oplus STK_3[2]$. We also can determine the value of $STK_3[0]$, from the knowledge of $STK_{19}[0]$, $STK_{17}[1]$, and $STK_1[1]$. We compute Y_4, and ΔY_4 as shown in Figure 9. Due to MC^{-1} operation on the active cells in the fourth column of X_5, we have $\Delta X_4[9] = \Delta X_4[15]$ that will lead to a c-bit filter. We can compute X_5, and ΔX_5 as shown in Figure 9. The time complexity of this step is $N2^{13c}$, and the number of tests to verify the impossible distinguisher is $N2^{12c}$.

Complexity. Analyzing N pairs has a time complexity of about $N2^{13c}.\frac{7}{21}$ 21-round encryptions. The attack needs a data complexity of $D = \sqrt{2^{c_B+c_F+1}.g \ln 2}$. The total time complexity is $T = D + M.2^{13c}.\frac{7}{21} + 2^{48c-g}$. Hence, to optimize the time complexity of the attack, we select $g = 21$ for $c = 4$ and $g = 40$ for $c = 8$. Thus, the data, time, and memory complexities of the attack on SKINNY-64-192 are $2^{62.43}$, $2^{174.42}$, and 2^{168}, respectively. The data, time, and memory complexities of the attack on SKINNY-128-384 are $2^{122.89}$, $2^{347.35}$, and 2^{336}, respectively.

F.4 27-round Related-Tweakey ID Attack on SKINNY-n-3n

This section provides a 27-round ID attack on SKINNY-n-3n in the related-tweakey setting. Figure 10 illustrates the attack discovered by our tool. In our 27-round attack, we use a related-tweakey impossible differential as:

$$(\Delta Y_3, \Delta STK_3) = (000a0\cdots0, 000a0\cdots0) \rightarrow (\Delta X_{20}) = (0000000b000b0000),$$

where a, and b are fixed non-zero differences. There are 2^c-1 non-zero different values for a; we denote them with $a_i, (i = 1, \cdots, 2^c-1)$. For each a_i, we have only a fixed non-zero difference b_i for b, a fixed non-zero difference u_i for $\Delta STK_1[5]$, and also a fixed non-zero difference v_i for $\Delta STK_{21}[7]$.[4] Therefore, there are 2^c-1 independent distinguishers with fixed non-zero input/output differences for our 27-round attack. If we consider only one of the ID distinguishers in the attack, then we will need more data to attack. Hence, we can consider all 2^c-1 ID distinguishers according to all possible fixed non-zero values of a in our attack to improve data complexity. Inspired by [36], we use 2^c-1 lists $L_i, (i = 1, \cdots, 2^c-1)$ for storing pairs. In fact, during the attack procedure, the data related to fixed non-zero differences (u_i, a_i, b_i, v_i) is saved in list $L_i, (i = 1, \cdots, 2^c-1)$ and we continue the attack based on each list. Using this strategy, we can discard more incorrect keys than using just one distinguisher with the same initial data.

Pair Generation We should build 2^x structures at W_0' and evaluate all possible values in four cells $W_0'[5, 7, 10, 13]$ for each structure, while the other cells assume a fixed value. By using 2^{x+4c} plaintexts, we can have 2^{x+8c-1} pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_B|-(n-|\Delta_F|)}$ pairs. In our 27-round attack $n = |\Delta_F|$, and thus, $N = 2^{x+2|\Delta_B|} = 2^{x+8c}$. This step needs a total of 2^{x+4c+1} encryption calls.

Guess-and-Filter For each of the N pairs we do the following steps:

a) *Satisfying round 1.* We guess $ETK_0[5]$ and compute $Y_1[5]$. Then, we determine i index such $Y_1[5] = u_i$ and store the pair in the list L_i and repeat this for the other pairs. Clearly, each pair will be saved in a list, so no filtering is required in this step. The time complexity of this step is $N2^c$, and the number of tests left for the next step is $N2^c$.

b) *Satisfying rounds 27, 26, and 25.* We guess $STK_{26}[0-7], STK_{25}[0-7]$, and $STK_{24}[0-7]$ and compute Z_{23}, and ΔZ_{23} as shown in Figure 10. Since $\Delta Z_{23}[11] = 0$, we have a c-bit filter on all lists. The time complexity of this step is $N2^{25c}$, and the number of tests left for the next step is $N2^{24c}$.

c) *Satisfying round 24.* We calculate $\Delta X_{23}[15]$ using the value of $Z_{23}[15]$, and $\Delta Z_{23}[15]$. We have $\Delta X_{23}[3] = \Delta X_{23}[7] = \Delta X_{23}[15]$ because of the MC operation on the active cells in the last column of W_{22}. Given $\Delta X_{23}[3, 7]$, we can

[4] We consider the values of $\Delta STK_1[5]$, and $\Delta STK_{21}[7]$, since only these differences affect our 27-round attack (see Figure 10).

determine $Y_{23}[3,7]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{23}[3,7]$. Now, we can determine $STK_{23}[3,7]$ (due to $Y_{23}[3] = Z_{23}[3] \oplus STK_{23}[3]$, and $Y_{23}[7] = Z_{23}[7] \oplus STK_{23}[7]$). Similarly, due to the MC operation on the active cells in the second and the first columns of W_{22}, we can also derive tweakey cells $STK_{23}[5]$ and $STK_{23}[4]$, respectively. We guess $STK_{23}[1,2]$. Now, we can compute $\Delta X_{22}[10,11,14,15]$ that will lead to two c-bit filters (due to MC operation of the third and the fourth column of W_{21}). The time complexity of this step is $N2^{26c}$, and the number of tests left for the next step is $N2^{24c}$.

d) *Satisfying round 24.* We guess $STK_{23}[0,6]$. Thus, we can compute Z_{22} and ΔZ_{22} as shown in Figure 10. From the Knowledge of $Z_{22}[8]$, and $\Delta Z_{22}[8]$, we compute $\Delta X_{22}[8]$ and so $\Delta Z_{21}[7]$ for each pair on list L_i. On the other hand, we have $\Delta STK_{21}[7] = \Delta Z_{21}[7]$. Therefore, for each pair on list L_i, checking if $\Delta STK_{21}[7] = v_i$, will lead to a c-bit filter. The time complexity of this step is $N2^{26c}$, and the number of tests left for the next step is $N2^{25c}$.

e) *Satisfying round 23.* We calculate $\Delta X_{22}[13]$ using $Z_{22}[13]$. We have $\Delta X_{22}[1] = \Delta X_{22}[5] = \Delta X_{22}[13]$ because of the MC operation on the active cells in the second column of W_{21}. Given $\Delta X_{22}[1,5]$, we can determine $Y_{22}[1,5]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{22}[1,5]$. Now, we can determine $STK_{22}[1,5]$ (due to $Y_{22}[1] = Z_{22}[1] \oplus STK_{22}[1]$, and $Y_{22}[5] = Z_{22}[5] \oplus STK_{22}[5]$). Similarly, due to the MC operation on the active cells in the third and the fourth column of W_{21}, we can also derive tweakey cells $STK_{22}[2,3]$. The time complexity of this step is $N2^{25c}$, and the number of tests left for the next step is $N2^{25c}$.

f) *Satisfying round 1.* We know the values of $STK_{26}[2,5]$, $STK_{24}[0,6]$, and $STK_{22}[1], ETK_0[5]$ from the previous steps. Therefore, we do not need to guess $ETK_0[10,13]$. The values $ETK_0[10,13]$ allows us to compute $\Delta Y_1[10,13]$. Due to MC^{-1} operation on the active cells in the first column of X_2, we have $\Delta Y_1[7] = \Delta Y_1[10] = \Delta Y_1[13]$. The equality $\Delta Y_1[10] = \Delta Y_1[13]$ will lead to a c-bit filter. From the knowledge of $\Delta Y_1[13]$, Lemma 1 helps us to derive $ETK_0[7]$. The time complexity of this step is $N2^{24c}$, and the number of tests left for the next step is $N2^{24c}$.

g) *Satisfying rounds 23 and 22.* We guess $STK_{22}[6,7]$. We compute Z_{21} and ΔZ_{21} as shown in Figure 10. Due to MC^{-1} operation on the active cells in the second column of W_{20}, we have $\Delta X_{21}[1] = \Delta X_{21}[9] = \Delta X_{21}[13]$. The equality $\Delta X_{21}[9] = \Delta X_{21}[13]$ will lead to a c-bit filter (we can compute these cells from knowledge of $Z_{21}[9,13]$ and $\Delta Z_{21}[9,13]$). From the knowledge of $\Delta X_{21}[1]$, Lemma 1 helps us to derive $STK_{21}[1]$. The time complexity of this step is $N2^{26c}$, and the number of tests left for the next step is $N2^{25c}$.

h) *Satisfying rounds 22 and 21.* We guess $STK_{21}[5]$ to compute $\Delta X_{20}[11]$. For each pair on list L_i, checking if $\Delta X_{20}[11] = b_i$, will lead to a c-bit filter. The time complexity of this step is $N2^{26c}$, and the number of tests left for the next step is $N2^{25c}$.

i) *Satisfying rounds 22 and 21.* We guess $STK_{21}[4]$ to compute $\Delta X_{20}[7]$. For each pair on list L_i, checking if $\Delta X_{20}[7] = b_i$, will lead to a c-bit filter. The

time complexity of this step is $N2^{26c}$, and the number of tests left for the next step is $N2^{25c}$.

j) *Satisfying rounds 1, 2 and 3.* We know the values of $ETK_0[0,3,4,5,7,9 \sim 13]$, $STK_1[0,3,4]$, and $STK_2[3]$ from the previous steps. Therefore, we can compute Y_3 and ΔY_3 as shown in Figure 10. For each pair on list L_i, checking if $\Delta Y_3[3] = a_i$, will lead to a c-bit filter. The time complexity of this step is $N2^{25c}$, and the number of tests left to verify the impossible distinguisher is $N2^{24c}$.

Complexity. Analyzing N pairs has a time complexity of about $N2^{26c}\frac{7}{27}$ 27-round encryptions. The attack needs a data complexity of $D = 2^{15c}g\ln 2$. The total time complexity is $T = D + N2^{26c}\frac{7}{27} + 2^{48c-g}$. Hence, to optimize the time complexity of the attack, we select $g = 9$ for $c = 4$ and $g = 23$ for $c = 8$. Thus, the data, time, and memory complexities of the attack on SKINNY-64-192 are $2^{62.64}$, $2^{183.26}$, and 2^{172}, respectively. The data, time, and memory complexities of the attack on SKINNY-128-384 are $2^{123.99}$, $2^{362.61}$, and 2^{344}, respectively.

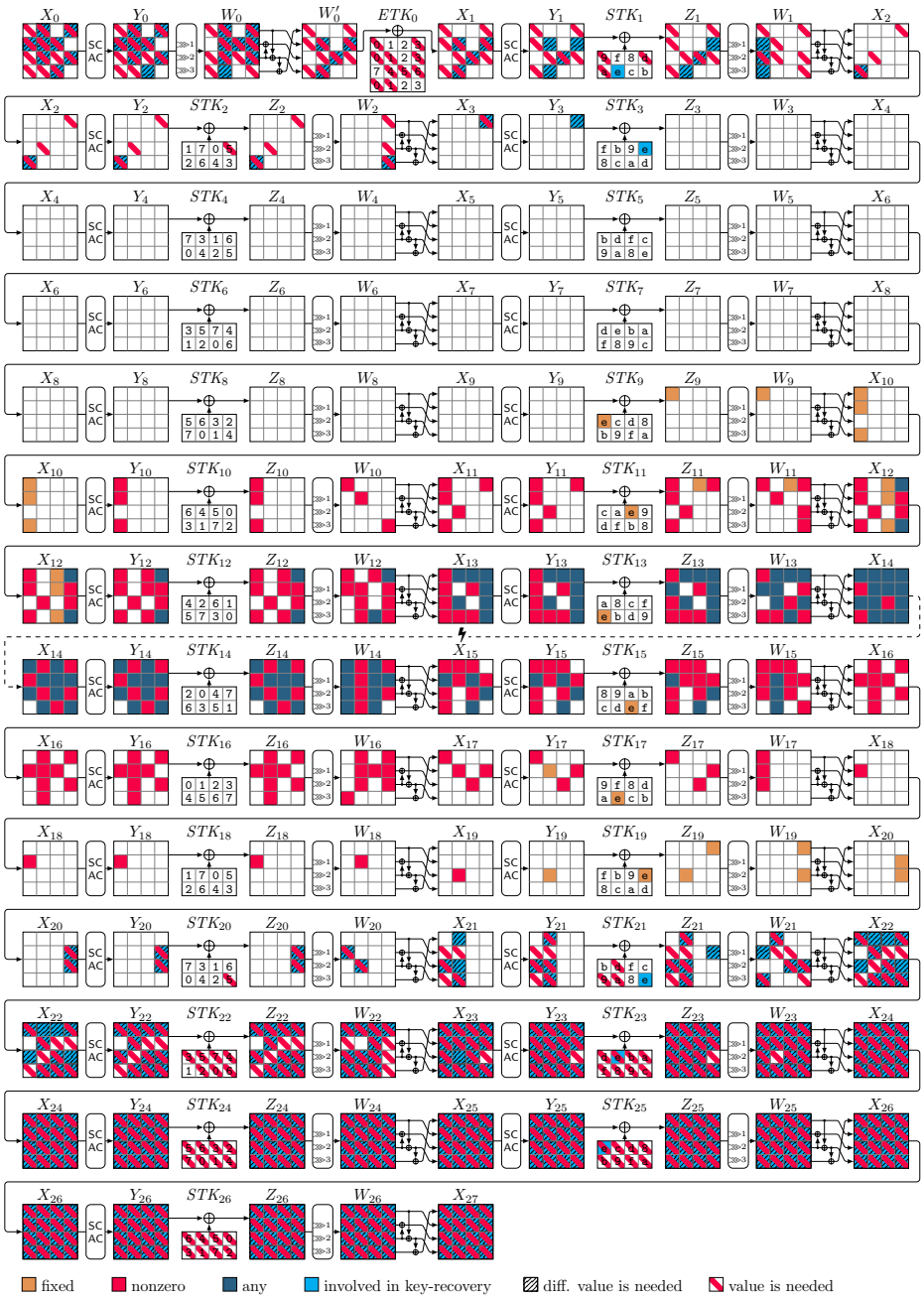

Fig. 10: ID attack on 27 rounds of SKINNY-n-$3n$ in the related-tweakey setting. $|k_B \cup k_F| = 43 \cdot c$, $c_B = 4 \cdot c$, $c_F = 16 \cdot c$, $\Delta_B = 4 \cdot c$, $\Delta_F = 16 \cdot c$.

G Multidimensional ZC Attacks on SKINNY

This section explains our multidimensional zero-correlation linear attacks on reduced-round of SKINNY.

G.1 ZC Attack on SKINNY-n-n

Fig. 11: ZC attack on 16 rounds of SKINNY-n-n. #Involved key cells: 8

If the ZC linear approximation over 9-round SKINNY in Figure 11 cover rounds 4 to 12, we can attack 16-round SKINNY-n-n by adding 3 rounds before and four rounds after the distinguisher, as shown in Figure 11. We can divide the attack procedure into the following steps:

1. Collect N pairs of plaintexts and the corresponding ciphertexts.

2. Allocate a $7c$-bit counter $N_0[W_0', Z_{15}]$ for all 2^{9c} possible value of $[W_0', Z_{15}]$ and initialize it to zero. Then, calculate the number of pairs of plaintext-ciphertext with given values W_0' and Z_{15} and increment the corresponding counter $N_0[W_0', Z_{15}]$. In this step, about 2^{16c} pairs divide into 2^{9c} distinct values of $[W_0', Z_{15}]$, so the $7c$-bit counter is sufficient.
3. Guess 3 cells $ETK[3, 5, 8]$. Next, allocate a counter $N_1[X_1, Z_{15}]$ for all 2^{9c} possible values of $[X_1, Z_{15}]$ and initialize it to zero. For all 2^{3c} possible values of W_0', encrypt W_0' one round to obtain X_1 and update the value $N_1[X_1, Z_{15}] = N_1[X_1, Z_{15}] + N_0[W_0', Z_{15}]$ for all 2^{6c} values of Z_{15}. The time complexity of this step is equal to $2^{3c} \times 2^{3c} \times 2^{6c} = 2^{12c}$ memory access, because we should guess 3 cells for ETK_1, and for 2^{3c} values encrypt Y_1 one round and update N_1 for 2^{6c} times.
4. Guess 2 cells $STK_1[3, 5]$. Next, allocate a counter $N_2[X_2, Z_{15}]$ for all 2^{8c} possible values of $[X_2, Z_{15}]$ and initialize it to zero. For all 2^{3c} possible values of X_1, encrypt X_1 one round to obtain X_2 and update the value $N_2[X_2, Z_{15}] = N_2[X_2, Z_{15}] + N_1[X_1, Z_{15}]$ for all 2^{6c} values of Z_{15}. The time complexity of this step is equal to $2^{3c+2c} \times 2^{3c} \times 2^{6c} = 2^{14c}$ memory access.
5. From the knowledge of $ETK[3]$, we can determine the value of $STK_2[7]$. Next, allocate a counter $N_3[X_3, Z_{15}]$ for all 2^{7c} possible value of $[X_3, Z_{15}]$ and initialize it to zero. For all 2^{2c} possible values of X_2, encrypt X_2 one round to obtain X_3 and update the value $N_3[X_3, Z_{15}] = N_3[X_3, Z_{15}] + N_2[X_2, Z_{15}]$ for all 2^{6c} values of Z_{15}. The time complexity of this step is equal to $2^{5c} \times 2^{2c} \times 2^{6c} = 2^{13c}$ memory access.
6. From the knowledge of $STK_1[3]$, we can determine the value of $STK_{15}[5]$. Guess 2 cells $STK_{15}[2, 7]$. Next, allocate a counter $N_4[X_3, Z_{14}]$ for all 2^{4c} possible value of $[X_3, Z_{14}]$ and initialize it to zero. For all 2^{6c} possible values of Z_{15}, decrypt Z_{15} to obtain Z_{14} and update the value $N_4[X_3, Z_{14}] = N_4[X_3, Z_{14}] + N_3[X_3, Z_{15}]$ for all 2^c values of X_3. The time complexity of this step is equal to $2^{5c+2c} \times 2^{6c} \times 2^c = 2^{14c}$ memory access.
7. From the knowledge of $ETK[7]$, we can determine the value of $STK_{14}[3]$. Guess 1 cell $STK_{14}[4]$. Next, allocate a counter $N_5[X_3, Z_{13}]$ for all 2^{3c} possible value of $[X_3, Z_{13}]$ and initialize it to zero. For all 2^{3c} possible values of Z_{14}, decrypt Z_{14} to obtain Z_{13} and update the value $N_5[X_3, Z_{13}] = N_5[X_3, Z_{13}] + N_4[X_3, Z_{15}]$ for all 2^c values of X_3. The time complexity of this step is equal to $2^{7c+c} \times 2^{3c} \times 2^c = 2^{12c}$ memory access.
8. From the knowledge of $STK_{15}[2]$, we can determine the value of $STK_{13}[0]$. Next, allocate a counter $N_6[X_3, X_{12}]$ for all 2^{2c} possible value of $[X_3, X_{12}]$ and initialize it to zero. For all 2^{2c} possible values of Z_{13}, decrypt Z_{13} to obtain X_{12} and update the value $N_6[X_3, X_{12}] = N_6[X_3, X_{12}] + N_5[X_3, Z_{13}]$ for all 2^c values of X_3. The time complexity of this step is equal to $2^{8c} \times 2^{2c} \times 2^c = 2^{11c}$ memory access.
9. To recover the secret key, we allocate a counter $V[z]$ for $2c$-bit z. For 2^{2c} values of $[X_3, X_{12}]$, evaluate all $2c$ basis ZC masks on $[X_3, X_{12}]$ and get z. Update the counter $V[z]$ by $V[z] = V[z] + N_6[X_3, X_{12}]$. Calculate the statistical value T (Equation 4), if $T < \tau$, the guessed key values are possible right key candidates.

The time complexity of this step is equal to $2^{8c} \times 7c \times 2^{2c}$ times of reading the $7c$-bit memory, because for all of guessed 2^{8c} keys in previous steps, we should read 2^{2c} values of $N_6[X_3, X_{12}]$.

10. Do an exhaustive search for all the right candidates. Due to β (the probability of accepting a wrong key) and the total number of recovered bits being $8c$, the number of remaining key candidates is $\beta \times 2^{8c}$. Then we exhaustively search other $16c-8c = 8c$ key bits, the time complexity will be $\beta \times 2^{8c} \times 2^{8c} = \beta \times 2^{16c}$ times of 16-round encryptions.

Complexity. In this attack, for $c = 4$, we set the type-I error probability $\alpha = 2^{-2.7}$ and the type-II error probability $\beta = 2^{-2}$, then $Z_{1-\alpha} = 1.01$, and $Z_{1-\beta} = 0.67$. Thus, based on the Equation 5; $N = 2^{61.35}$. The decision threshold is $\tau = \mu_0 + \sigma_0 Z_{1-\alpha}$. If we consider one memory access as one round encryption call, then the time complexity of our attack on 16-round SKINNY-64-64 is about $2^{61.35} + (2^{48} + 2^{56} + \cdots + 2^{44.80}) \times \frac{1}{16} + 2^{62} = 2^{62.71}$ 16-round encryptions. The required memory complexity is dominated by step 2, which needs about $2^{37.8}$ bytes.

For $c = 8$, our key recovery attack on 16-round SKINNY-128-128 needs $2^{122.3}$ known plaintexts, $2^{122.79}$ encryptions, and $2^{74.8}$ bytes memory, if we set $\alpha = 2^{-2.7}$ and $\beta = 2^{-7}$. Furthermore, the success probability of the attack is $1 - \alpha = 0.84$.

G.2 ZC Attack on SKINNY-n-$2n$

If the ZC linear approximations over 9-round SKINNY in Figure 12 cover rounds 5 to 13, we can attack 19-round SKINNY-n-$2n$ by adding four rounds before and six rounds after the linear approximations, as shown in Figure 12. We divide the attack procedure into the following steps:

1. Collect N pairs of plaintexts and the corresponding ciphertexts. Guess 13 cells $STK_{18}[0-7]$, and $STK_{17}[0,1,4,6,7]$, do the partial decryption and calculate Z_{16} for each pair. Allocate a $4c$-bit counter $N_0[W_0', Z_{16}]$ for all 2^{12c} possible value of $[W_0', Z_{16}]$ and initialize it to zero. Next, compute the number of pairs of plaintext-ciphertext with given values W_0', and Z_{16} and store it in $N_0[W_0', Z_{16}]$. In this step, around 2^{16c} pairs are divided into 2^{12c} distinct values of $[W_0', Z_{16}]$, so $4c$-bit counter is sufficient. The time complexity of this step is equal to $N + N \times 2^{13c}$.
2. Guess 3 cells $STK_{16}[2,5,7]$. Next, allocate a counter $N_1[W_0', Z_{15}]$ for all 2^{9c} possible value of $[W_0', Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Z_{16}, do the partial decryption to obtain Z_{15} and update the value $N_1[W_0', Z_{15}] = N_1[W_0', Z_{15}] + N_0[W_0', Z_{16}]$ for all 2^{6c} values of W_0'. The time complexity of this step is equal to $2^{(13c+3c)} \times 2^{6c} \times 2^{6c} = 2^{28c}$ memory access.
3. Guess 2 cells $STK_{15}[3,4]$. Next, allocate a counter $N_2[W_0', Z_{14}]$ for all 2^{8c} possible value of $[W_0', Z_{14}]$ and initialize it to zero. For all 2^{3c} possible values of Z_{15}, do the partial decryption to obtain Z_{14} and update the value $N_2[W_0', Z_{14}] = N_2[W_0', Z_{14}] + N_1[W_0', Z_{15}]$ for all 2^{6c} values of Y_0. The time complexity of this step is equal to $2^{(16c+2c)} \times 2^{3c} \times 2^{6c} = 2^{27c}$ memory access.

Fig. 12: ZC attack on 19 rounds of SKINNY-n-$2n$. #Involved key cells: 24.

4. We know $STK_{18}[4]$ and $STK_{16}[2]$, thus, we can determine the value of $STK_{14}[0]$. Next, allocate a counter $N_3[W'_0, Y_{13}]$ for all 2^{7c} possible value of $[W'_0, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Z_{14}, do the partial decryption to obtain Y_{13} and update the value $N_3[W'_0, Y_{13}] = N_3[W'_0, Y_{13}] + N_2[W'_0, Z_{14}]$ for all 2^{6c} values of Y_0. The time complexity of this step is equal to $2^{18c} \times 2^{7c} \times 2^{6c} = 2^{31c}$ memory access.
5. From the knowledge of $STK_{18}[4]$ and $STK_{16}[2]$, we can determine the value of $ETK[2, 6]$. Also, from $STK_{18}[1]$ and $STK_{16}[7]$, we determine the value of $ETK[8]$. Thus, we just guess 3 cells $ETK[0, 9, 15]$. Next, allocate a counter $N_4[Y_1, Y_{13}]$ for all 2^{4c} possible value of $[Y_1, Y_{13}]$ and initialize it to zero. For all 2^{6c} possible values of W'_0, do the partial decryption to obtain Y_1 and update the value $N_4[Y_1, Y_{13}] = N_4[Y_1, Y_{13}] + N_3[W'_0, Y_{13}]$ for all 2^c values of

Y_{13}. The time complexity of this step is equal to $2^{(18c+3c)} \times 2^{6c} \times 2^c = 2^{28c}$ memory access.

6. From the knowledge of $STK_{17}[6]$ and $STK_{15}[4]$, we determine the value of $STK_1[6]$. Therefore, we just guess 2 cells $STK_1[0,2]$. Next, allocate a counter $N_5[Y_2, Y_{13}]$ for all 2^{3c} possible value of $[Y_2, Y_{13}]$ and initialize it to zero. For all 2^{6c} possible values of Y_1, do the partial decryption to obtain Y_2 and update the value $N_5[Y_2, Y_{13}] = N_5[Y_2, Y_{13}] + N_4[Y_1, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{(21c+2c)} \times 2^{6c} \times 2^c = 2^{30c}$ memory access.
7. From the knowledge of $STK_{18}[2]$ and $ETK[0]$, we determine the value of $STK_2[2]$. Also, we determine the value of $STK_2[4]$ from $STK_{18}[4]$ and $STK_{16}[2]$. Next, allocate a counter $N_5[Y_3, Y_{13}]$ for all 2^{3c} possible value of $[Y_3, Y_{13}]$ and initialize it to zero. For all 2^{3c} possible values of Y_2, do the partial decryption to obtain Y_3 and update the value $N_5[Y_3, Y_{13}] = N_5[Y_3, Y_{13}] + N_4[Y_2, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{23c} \times 2^{3c} \times 2^c = 2^{27c}$ memory access.
8. Guess 1 cell $STK_3[6]$. Then, allocate a counter $N_6[X_4, Y_{13}]$ for all 2^{2c} possible value of $[X_4, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Y_3, do the partial decryption to obtain X_4 and update the value $N_6[X_4, Y_{13}] = N_6[X_4, Y_{13}] + N_5[Y_3, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{(23c+c)} \times 2^{2c} \times 2^c = 2^{27c}$ memory access.
9. To recover the secret key, allocate a counter $V[z]$ for $2c$-bit z. For 2^{2c} values of $[X_4, Y_{13}]$, evaluate all $2c$ basis ZC masks on $[X_4, Y_{13}]$ and get z. Update the counter $V[z]$ by $V[z] = V[z] + N_6[X_4, Y_{13}]$. Calculate the statistical value T (Equation 4), if $T < \tau$, the guessed key values are possible right key candidates. The time complexity of this step is equal to $2^{24c} \times 4c \times 2^{2c}$ times of reading the $4c$-bit memory.
10. Do an exhaustive search for all the right candidates. The time complexity of this step is $\beta \times 2^{32c}$.

Complexity. For $c = 4$, we set $\alpha = 2^{-2.7}$ and $\beta = 2^{-9}$, then $Z_{1-\alpha} = 1.01$, and $Z_{1-\beta} = 2.88$. Thus, based on the Equation 5; $N = 2^{62.89}$. The decision threshold is $\tau = \mu_0 + \sigma_0 Z_{1-\alpha}$. If we consider one memory access as one round encryption call, then the time complexity of our attack on 19-round SKINNY-64-128 is about $2^{62.89} + 2^{114.89} \times \frac{1}{19} + (2^{112} + 2^{108} + \cdots + 2^{108}) \times \frac{2}{19} + 2^{119} = 2^{119.15}$ 19-round encryptions. The required memory complexity is dominated by step 1, which needs about 2^{49} bytes.

For $c = 8$, by selecting $\alpha = 2^{-2.7}$ and $\beta = 2^{-16}$, our key recovery attack on 19-round SKINNY-128-256 requires $2^{122.9}$ known plaintexts, $2^{240.07}$ encryptions, and 2^{98} bytes memory. The success probability of the attacks is $1 - \alpha = 0.84$.

G.3 ZC Attack on SKINNY-n-3n

If the ZC linear approximation over 9-round SKINNY-n-3n in Figure 13 cover rounds 5 to 13, we can attack 21-round SKINNY-n-3n by adding 4 rounds before

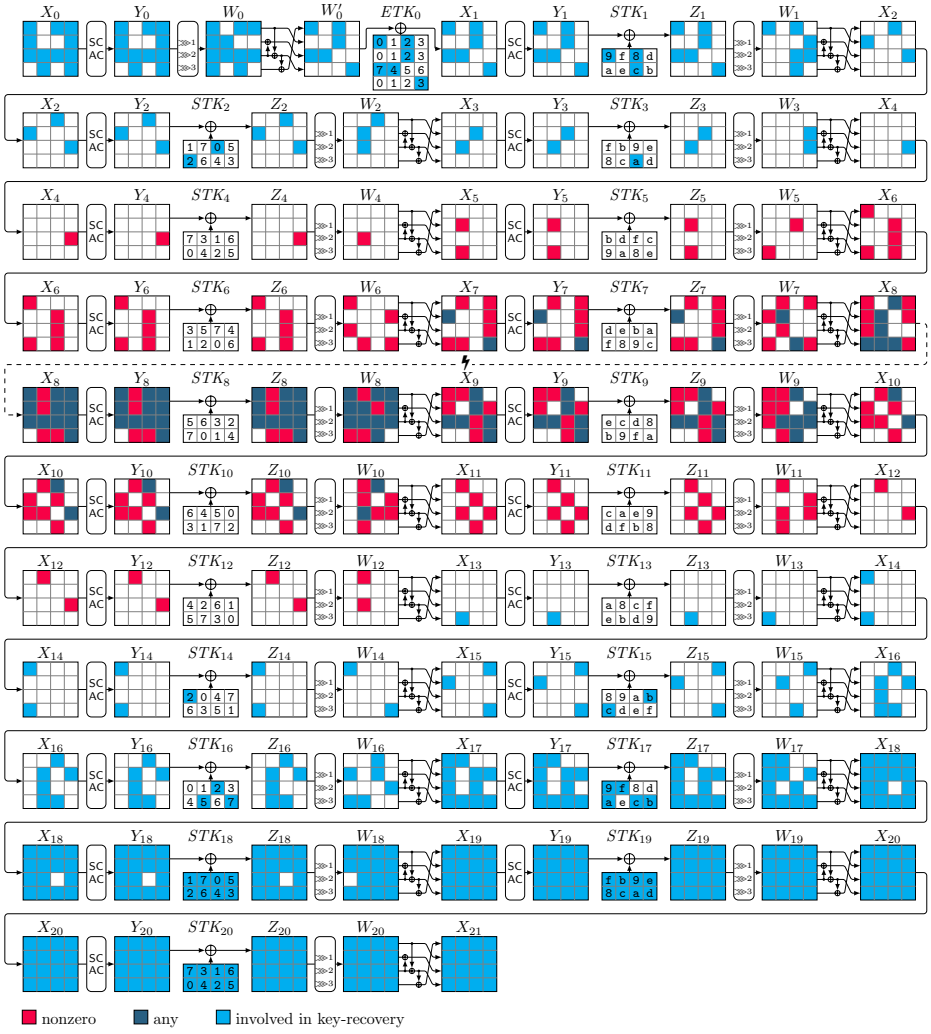

Fig. 13: ZC attack on 21 rounds of SKINNY-n-3n. #Involved key cells: 40

and 8 rounds after the linear approximations, as shown in Figure 13. We can divide the attack procedure into the following steps:

1. Collect N pairs of plaintexts and the corresponding ciphertexts. Guess 29 cells $STK_{20}[0-7], STK_{19}[0-7], STK_{18}[0-7]$, and $STK_{17}[0,1,4,6,7]$, do the partial decryption and calculate Z_{16} for each pair. Allocate a $4c$-bit counter $N_0[W_0', Z_{16}]$ for all 2^{12c} possible value of $[W_0', Z_{16}]$ and initialize it to zero. Next, compute the number of pairs of plaintext-ciphertext with given values W_0', and Z_{16} and store it in $N_0[W_0', Z_{16}]$. In this step, around 2^{16c} pairs are

divided into 2^{12c} distinct values of $[W'_0, Z_{16}]$, so $4c$-bit counter is sufficient. The time complexity of this step is $N + N \times 2^{29c}$.

2. Guess 5 cells $ETK[0, 2, 6, 8, 9, 15]$ ($ETK[6] = ETK[2]$). Next, allocate a counter $N_1[Y_1, Z_{16}]$ for all 2^{12c} possible value of $[Y_1, Z_{16}]$ and initialize it to zero. For all 2^{6c} possible values of Y_0, do the partial encryption to obtain Y_1 and update the value $N_1[Y_1, Z_{16}] = N_1[Y_1, Z_{16}] + N_0[W'_0, Z_{16}]$ for all 2^{6c} values of Z_{16}. The time complexity of this step is equal to $2^{(29c+5c)} \times 2^{6c} \times 2^{6c} = 2^{46c}$.
3. From the knowledge of $STK_{20}[0, 6]$, $STK_{18}[1, 4]$, and $ETK[2, 8]$, we can determine the values of $STK_{16}[2, 7]$. Thus, we guess just 1 cell $TK_{16}[5]$. Allocate a counter $N_2[Y_1, Z_{15}]$ for all 2^{9c} possible value of $[Y_1, Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Z_{16}, do the partial decryption to obtain Z_{15} and update the value $N_2[Y_1, Z_{15}] = N_2[Y_1, Z_{15}]+N_1[Y_1, Z_{16}]$ for all 2^{6c} values of Y_1. The time complexity of this step is equal to $2^{(34c+c)} \times 2^{6c} \times 2^{6c} = 2^{47c}$.
4. Guess 3 cells $STK_1[0, 2, 6]$. Allocate a counter $N_3[Y_2, Z_{15}]$ for all 2^{6c} possible value of $[Y_2, Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Y_1, do the partial encryption to obtain Y_2 and update the value $N_3[Y_2, Z_{15}] = N_3[Y_2, Z_{15}] + N_2[Y_1, Z_{15}]$ for all 2^{3c} values of Z_{15}. The time complexity of this step is equal to $2^{(35c+3c)} \times 2^{6c} \times 2^{3c} = 2^{47c}$.
5. We can determine the values of $STK_{15}[4]$ from the knowledge of $STK_{19}[5]$, $STK_{17}[6]$,and $STK_1[6]$. Therefore, guess just 1 cell $STK_{15}[3]$. Allocate a counter $N_4[Y_2, Z_{14}]$ for all 2^{5c} possible value of $[Y_2, Z_{14}]$ and initialize it to zero. For all 2^{3c} possible values of Z_{15}, do the partial decryption to obtain Z_{14} and update the value $N_4[Y_2, Z_{14}] = N_4[Y_2, Z_{14}]+N_3[Y_2, Z_{15}]$ for all 2^{3c} values of Y_2. The time complexity of this step is equal to $2^{(38c+c)} \times 2^{3c} \times 2^{3c} = 2^{45c}$.
6. We determine the values of $TK_{14}[0]$ from the knowledge of $STK_{20}[6]$, $STK_{18}[4]$, and $STK_{16}[2]$. Then, allocate a counter $N_5[Y_2, Y_{13}]$ for all 2^{4c} possible values of $[Y_2, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Z_{14}, do the partial decryption to obtain Y_{13} and update the value $N_5[Y_2, Y_{13}] = N_5[Y_2, Y_{13}] + N_4[Y_2, Z_{15}]$ for all 2^{3c} values of Y_2. The time complexity of this step is equal to $2^{39c} \times 2^{2c} \times 2^{3c} = 2^{44c}$.
7. From the knowledge of $STK_{20}[4, 6]$, $STK_{18}[2, 4]$,and $ETK[0, 2]$, we determine $STK_2[2, 4]$. Then, allocate a counter $N_6[Y_3, Y_{13}]$ for all 2^{3c} possible values of $[Y_3, Y_{13}]$ and initialize it to zero. For all 2^{3c} possible values of Y_2, do the partial encryption to obtain Y_3 and update the value $N_6[Y_3, Y_{13}] = N_6[Y_3, Y_{13}] + N_5[Y_2, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{39c} \times 2^{3c} \times 2^c = 2^{43c}$.
8. Guess 1 cell $STK_3[6]$, and then, allocate a counter $N_7[X_4, Y_{13}]$ for all 2^{2c} possible values of $[X_4, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Y_3, do the partial encryption to obtain X_4 and update the value $N_7[X_4, Y_{13}] = N_7[X_4, Y_{13}] + N_6[Y_3, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{(39c+c)} \times 2^{2c} \times 2^c = 2^{43c}$.
9. To recover the secret key, allocate a counter $V[z]$ for $2c$-bit z. For 2^{2c} values of $[X_4, Y_{13}]$, evaluate all $2c$ basis ZC masks on $[X_4, Y_{13}]$ and get z. Update the counter $V[z]$ by $V[z] = V[z]+N_7[X_4, Y_{13}]$. Calculate the statistical value T. If $T < \tau$, the guessed key values are possible right key candidates. The

time complexity of this step is equal to $2^{40c} \times 4c \times 2^{2c}$ times of reading the $4c$-bit memory.

1. Do an exhaustive search for all the right candidates. The time complexity of this step is equal to $\beta \times 2^{48c}$.

Complexity. For $c = 4$, we set $\alpha = 2^{-2.7}$ and $\beta = 2^{-7}$, then $Z_{1-\alpha} = 1.01$, and $Z_{1-\beta} = 2.41$. Thus, based on the Equation 5; $N = 2^{62.63}$. The decision threshold is $\tau = \mu_0 + \sigma_0 Z_{1-\alpha}$. If we consider one memory access as one round encryption call, then the time complexity of our attack on 21-round SKINNY-64-192 is about $2^{62.63} + 2^{178.63} \times \frac{4}{21} + (2^{184} + 2^{188} + \cdots + 2^{172} + 2^{172}) \times \frac{1}{21} + 2^{185} = 2^{185.83}$ 21-round encryptions. The required memory complexity is dominated by step 1, which needs about 2^{49} bytes.

For $c = 8$, by selecting $\alpha = 2^{-2.7}$ and $\beta = 2^{-14}$, our key recovery attack on 19-round SKINNY-128-384 requires $2^{122.81}$ known plaintexts, $2^{372.82}$ encryptions, and 2^{98} bytes memory. The success probability of the attack is $1 - \alpha = 0.84$.

H Integral Attacks on SKINNY

In this section, we transform ZC linear hulls into integral distinguishers to obtain integral attacks for SKINNY in the single-key/chosen-tweak setting.

H.1 ZC-Integral Key-Recovery Attack on 22-Round SKINNY-n-$2n$

Our tool finds a ZC distinguisher for 14 rounds (labelled as rounds 1 to 14 in Figure 14), combined with one free initial round (round 0) and a final key recovery phase over 7 rounds (15 to 21). In this distinguisher, the tweakey cell 8 is only active in at most $z = 2$ cells ('any' in STK_7, 'active' in STK_9). At the input to the distinguisher, 4 cells are active. Thus, we can convert it to an integral distinguisher [1] with data complexity $2^{4\cdot(16-4+2)} = 2^{56}$, where the values in the active input cells and the 2 tweakey cells with index 8 iterate over all values. The inactive input cells are constant, the other tweakey cells form the $4 \cdot 2 \cdot 15 = 120$-bit key. Then, the distinguisher's outputs in $W_{14}[14]$ sum to zero. We can trivially prepend 1 round because the addition of the equivalent tweakey does not change the input structure (key cell 8 is not involved), and all other operations in the first round are unkeyed.

Key recovery. For the key recovery, we separately recover the sums in $X_{15}[2]$ and $X_{15}[14]$ using the partial-sum technique [16] and merge the results following the meet-in-the-middle approach [38]. The procedures for both sums are summarized in Table 4. For each sum, we start with Step 0 by storing the obtained ciphertexts (after unwrapping the last linear layer, i.e., Z_{21}) together with their corresponding chosen tweakey values. For the tweakey, we either store the required subtweakey values (i.e., $STK_{21}[6]$) or, if the index is involved more than $z = 2$ times, the input tweakey values from which all subtweakeys can be reconstructed. In each of the following steps in round r, we guess one column of involved subtweakey STK_r and replace the stored values of this column in Z_r by those (potentially fewer) in W_{r-1}. If a subtweakey index is involved more than $z = 2$ times, we only guess the first z times and derive the remaining values afterwards. In each round, we reorder the column order if necessary to minimize the complexity of this round; that is, we first handle columns with fewer key guesses and a stronger reduction in the MixColumns step.

Complexity. The complexity of each step is determined by the number of guessed key cells so far and the number of new stored cells (for memory) or previously stored cells (for time). We use the number of S-box lookups as unit for the time complexity, as customary in previous attacks (although in reality, the memory accesses would likely be more expensive). Overall, we obtain a mapping from values of the sum in $X_{15}[2]$ to corresponding 2^{92} key candidates with complexity $2^{106.7}$, and for the sum in $X_{15}[14]$ for 2^{96} candidates with complexity $2^{102.8}$. These can be merged to obtain $2^{120-4} = 2^{116}$ key candidates that produce zero-sums. This remaining keyspace can either be brute-forced (complexity 2^{116}), or the attack can be repeated 3 times (complexity $3 \cdot 2^{106.7} = 2^{108.3}$ plus merging

plus $2^{120-3\cdot4} = 2^{108}$). As merging can be done efficiently, the total complexity is less than 2^{110} encryptions equivalents for 22-round SKINNY-64-128 with 120-bit keys. The same approach yields a complexity of $3 \cdot 2^{216-5.3} + 2^{240-3\cdot8} = 2^{216}$ for 22-round SKINNY-128-256 with 240-bit keys.

H.2 ZC-Integral Key-Recovery Attack on 26-Round SKINNY-n-3n

We follow the same approach as above with 4 active input cells, where tweakey cell index e is only active $z = 3$ times ('any' in STK_7, STK_9, 'active' in STK_{11}) in the 16 distinguisher rounds labelled 1 to 16 plus the free initial round 0. We append 9 rounds for key recovery for a total of 26 rounds. The distinguisher and key recovery are illustrated in Figure 15, with key-recovery details given in Table 5.

Complexity. The combined complexity of recovering $X_{17}[1]$ and $X_{17}[13]$ from $2^{4\cdot(16-4+3)} = 2^{60}$ data is $2^{172-4.8} + 2^{172-4.9} = 2^{168.2}$. We repeat the attack 2 times and then brute-force the remaining keyspace of $2^{180-2\cdot4} = 2^{172}$ candidates, which dominates the complexity for 26-round SKINNY-64-192 with 180-bit key. For 26-round SKINNY-128-384 with 360-bit key, the complexity is $2^{2\cdot172-4.8} + 2^{2\cdot172-4.9} = 2^{340.2}$ per repetition; with 2 repetitions, the brute-force complexity of $2^{360-2\cdot8} = 2^{344}$ encryptions dominates.

H.3 ZC-Integral Key-Recovery Attack on 30-Round SKINNYe-v2

SKINNYe-v2 is essentially SKINNY-n-4n with cell size $c = 4$, so we follow the same approach. The distinguisher (with target tweakey cell f) and key recovery are illustrated in Figure 16, with key-recovery details given in Table 6. The combined complexity of recovering $X_{19}[2]$ and $X_{19}[14]$ from $2^{4\cdot(16-4+4)} = 2^{64}$ data is $2^{232-5.7} + 2^{228-5.6} = 2^{226.3}$ (plus memory accesses). We repeat the attack 2 times and then brute-force the remaining keyspace of $2^{240-2\cdot4} = 2^{232}$ candidates, which dominates the complexity for 30-round SKINNYe-v2 with 240-bit key.

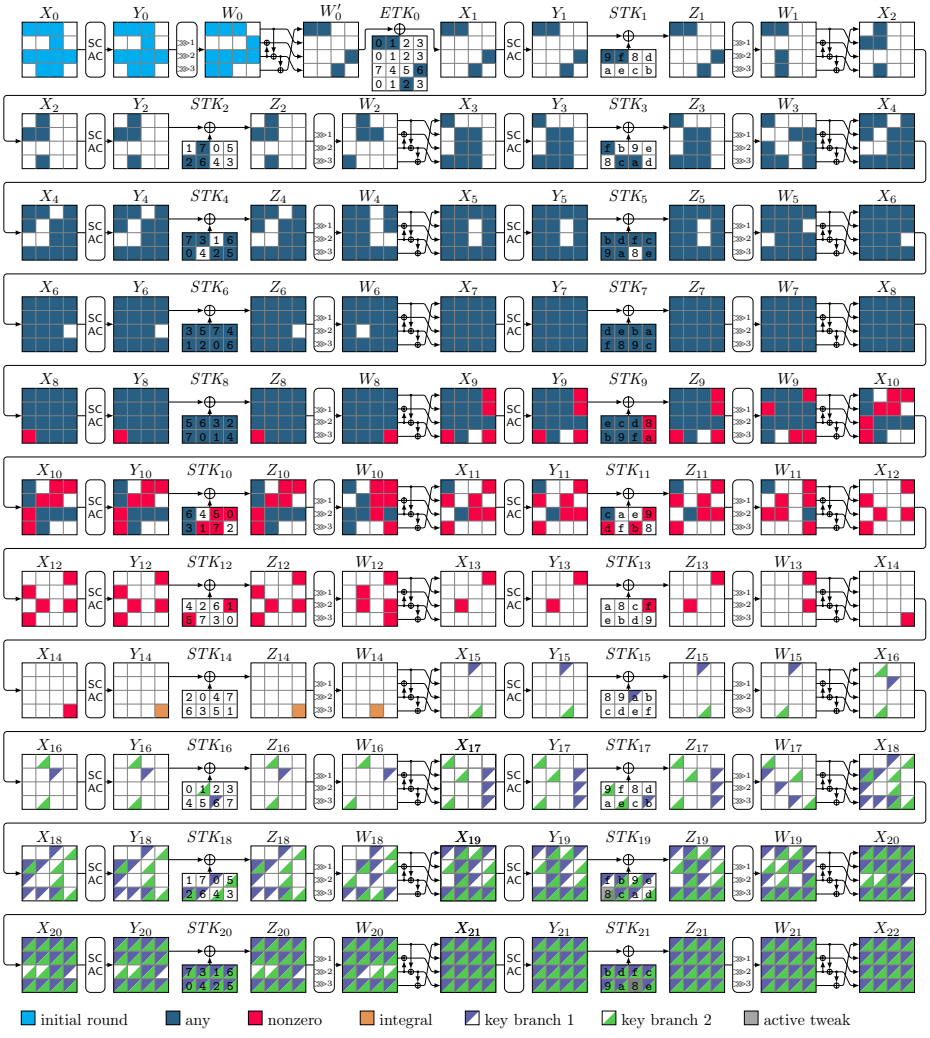

Fig. 14: ZC-based integral attack on 22 rounds of SKINNY-n-2n.

Table 4: Complexity of partial-sum key-recovery for 22 rounds of SKINNY-n-$2n$.

(a) Recovery of $X_{15}[2]$ (in Figure 14) with total complexity $2^{112-5.3} = 2^{106.7}$.

Step	Guessed	Keys×Data=Mem	Time	Stored Texts
0	–	$2^0 \times 2^{56} = 2^{56}$	2^{56}	$Z_{21}[0\text{–}15]$; $STK_{21}[6]$
1–4	$STK_{21}[0\text{–}5, 7]$	$2^{28} \times 2^{56} = 2^{84}$	$2^{84-6.5}$	$W_{20}[0\text{–}7, 9, 12\text{–}15]$
5	$STK_{20}[0, 4]$	$2^{36} \times 2^{56} = 2^{92}$	$2^{92-6.9}$	$Z_{20}[1\text{–}3, 5\text{–}7, 10, 11, 13\text{–}15]$; $W_{19}[0, 8, 12]$
6	$STK_{20}[1, 5]$	$2^{44} \times 2^{52} = 2^{96}$	$2^{100-6.9}$	$Z_{20}[2, 3, 6, 7, 10, 11, 14, 15]$; $W_{19}[0, 8, 12, 1, 13]$
7	$STK_{20}[2, 6]$	$2^{52} \times 2^{44} = 2^{96}$	$2^{104-6.5}$	$Z_{20}[3, 7, 11, 15]$; $W_{19}[0, 8, 12, 1, 13, 6, 14]$
8	$STK_{20}[3, 7]$	$2^{60} \times 2^{44} = 2^{104}$	$2^{104-6.5}$	$W_{19}[0, 8, 12, 1, 13, 6, 14, 3, 7, 11, 15]$
9	$STK_{19}[0]$	$2^{64} \times 2^{40} = 2^{104}$	$2^{108-7.5}$	$Z_{19}[1, 3, 5, 6, 9, 10, 13, 14, 15]$; $W_{18}[12]$
10	$STK_{19}[1, 5]$	$2^{72} \times 2^{32} = 2^{104}$	$2^{112-6.5}$	$Z_{19}[3, 6, 10, 14, 15]$; $W_{18}[12, 5, 13]$
11	$STK_{19}[6]$	$2^{76} \times 2^{32} = 2^{108}$	$2^{108-6.9}$	$Z_{19}[3, 15]$; $W_{18}[12, 5, 13, 2, 6, 10]$
12	$STK_{19}[3]$	$2^{80} \times 2^{28} = 2^{108}$	$2^{112-7.5}$	$W_{18}[12, 5, 13, 2, 6, 10, 15]$
13	$STK_{18}[4]$	$2^{84} \times 2^{20} = 2^{104}$	$2^{112-6.9}$	$Z_{18}[2, 5, 13, 14]$; $W_{17}[4]$
14	$STK_{18}[5]$	$2^{88} \times 2^{16} = 2^{104}$	$2^{108-7.5}$	$Z_{18}[2, 14]$; $W_{17}[4, 9]$
15	$STK_{18}[2]$	$2^{92} \times 2^{12} = 2^{104}$	$2^{108-7.5}$	$W_{17}[4, 9, 14]$
16	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{104-6.9}$	$W_{16}[7]$
17	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{96-8.5}$	$W_{15}[2]$
18	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{96-8.5}$	$X_{15}[2]$

(b) Recovery of $X_{15}[14]$ (in Figure 14) with total complexity $2^{108-5.2} = 2^{-102.8}$.

Step	Guessed	Keys×Data=Mem	Time	Stored Texts
0	–	$2^0 \times 2^{56} = 2^{56}$	2^{56}	$Z_{21}[0\text{–}15]$; $STK_{19}[4]$, $STK_{21}[6]$
1–4	$STK_{21}[0\text{–}5, 7]$	$2^{28} \times 2^{56} = 2^{84}$	$2^{84-6.5}$	$STK_{19}[4]$; $W_{20}[0\text{–}8, 10\text{–}15]$
5	$STK_{20}[0, 4]$	$2^{36} \times 2^{56} = 2^{92}$	$2^{92-6.5}$	$Z_{20}[1\text{–}3, 5\text{–}7, 9, 10, 13\text{–}15]$; $STK_{19}[4]$; $W_{19}[4, 12]$
6	$STK_{20}[2, 6]$	$2^{44} \times 2^{52} = 2^{96}$	$2^{100-6.5}$	$Z_{20}[1, 3, 5, 7, 9, 13, 15]$; $STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14]$
7	$STK_{20}[3, 7]$	$2^{52} \times 2^{48} = 2^{100}$	$2^{104-6.9}$	$Z_{20}[1, 5, 9, 13]$; $STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14, 11, 15]$
8	$STK_{20}[1, 5]$	$2^{60} \times 2^{48} = 2^{108}$	$2^{108-6.5}$	$STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14, 11, 15, 1, 5, 9, 13]$
9	–	$2^{60} \times 2^{40} = 2^{100}$	$2^{108-7.5}$	$Z_{19}[1, 2, 5, 7, 9, 11, 13, 14, 15]$; $W_{18}[8]$
10	$STK_{19}[1, 5]$	$2^{68} \times 2^{32} = 2^{100}$	$2^{108-6.5}$	$Z_{19}[2, 7, 11, 14, 15]$; $W_{18}[8, 5, 13]$
11	$STK_{19}[2]$	$2^{72} \times 2^{28} = 2^{100}$	$2^{104-7.5}$	$Z_{19}[7, 11, 15]$; $W_{18}[8, 5, 13, 14]$
12	$STK_{19}[7]$	$2^{76} \times 2^{24} = 2^{100}$	$2^{104-6.9}$	$W_{18}[8, 5, 13, 14, 3, 7]$
13	$STK_{18}[6]$	$2^{80} \times 2^{16} = 2^{96}$	$2^{104-6.9}$	$Z_{18}[3, 4, 15]$; $W_{17}[6]$
14	$STK_{18}[4]$	$2^{84} \times 2^{16} = 2^{100}$	$2^{100-8.5}$	$Z_{18}[3, 15]$; $W_{17}[6, 0]$
15	$STK_{18}[3]$	$2^{88} \times 2^{12} = 2^{100}$	$2^{104-7.5}$	$W_{17}[6, 0, 15]$
16	–	$2^{88} \times 2^{8} = 2^{96}$	$2^{100-7.5}$	$Z_{17}[5]$; $W_{16}[12]$
17	$STK_{17}[5]$	$2^{92} \times 2^{8} = 2^{100}$	$2^{100-8.5}$	$W_{16}[12, 1]$
18	$STK_{16}[1]$	$2^{96} \times 2^{4} = 2^{100}$	$2^{104-7.5}$	$W_{15}[13]$
19	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{100-8.5}$	$X_{15}[14]$

Fig. 15: ZC-based integral attack on 26 rounds of SKINNY-n-3n.

Table 5: Complexity of partial-sum key-recovery for 26 rounds of SKINNY-n-3n.

(a) Recovery of $X_{17}[1]$ (in Figure 15) with total complexity $2^{167.2}$.

Step	Guessed	Keys×Data=Mem	Time	Stored Texts
0	–	$2^0 \times 2^{60} = 2^{60}$	2^{60}	$Z_{25}[0\text{–}15]$; $STK_{23}[1], STK_{25}[0]$
1–4	$STK_{25}[1\text{–}7]$	$2^{28} \times 2^{60} = 2^{88}$	$2^{88-6.7}$	$STK_{23}[1]$; $W_{24}[0\text{–}15]$
5–8	$STK_{24}[0\text{–}7]$	$2^{60} \times 2^{60} = 2^{120}$	$2^{120-6.7}$	$STK_{23}[1]$; $W_{23}[0\text{–}15]$
9	$STK_{23}[0,4]$	$2^{68} \times 2^{60} = 2^{128}$	$2^{128-6.7}$	$Z_{23}[1\text{–}3,5\text{–}7,9\text{–}11,13\text{–}15]$; $STK_{23}[1]$; $W_{22}[0,4,8,12]$
10	$STK_{23}[5]$	$2^{72} \times 2^{60} = 2^{132}$	$2^{132-6.7}$	$Z_{23}[2,3,6,7,10,11,14,15]$; $W_{22}[0,4,8,12,1,5,13]$
11	$STK_{23}[2,6]$	$2^{80} \times 2^{56} = 2^{136}$	$2^{140-6.7}$	$Z_{23}[3,7,11,15]$; $W_{22}[0,4,8,12,1,5,13,2,6,14]$
12	$STK_{23}[3,7]$	$2^{88} \times 2^{56} = 2^{144}$	$2^{144-6.7}$	$W_{22}[0,4,8,12,1,5,13,2,6,14,3,7,11,15]$
13	$STK_{22}[0,4]$	$2^{96} \times 2^{52} = 2^{148}$	$2^{152-7.1}$	$Z_{22}[1,2,3,5,6,7,9,10,13,14,15]$; $W_{21}[0,12]$
14	$STK_{22}[1,5]$	$2^{104} \times 2^{44} = 2^{148}$	$2^{156-6.7}$	$Z_{22}[2,3,6,7,10,14,15]$; $W_{21}[0,12,5,13]$
15	$STK_{22}[2,6]$	$2^{112} \times 2^{44} = 2^{156}$	$2^{156-6.7}$	$Z_{22}[3,7,15]$; $W_{21}[0,12,5,13,2,6,10,14]$
16	$STK_{22}[3,7]$	$2^{120} \times 2^{44} = 2^{164}$	$2^{164-7.1}$	$W_{21}[0,12,5,13,2,6,10,14,3,11,15]$
17	$STK_{21}[0,4]$	$2^{128} \times 2^{36} = 2^{164}$	$2^{172-6.7}$	$Z_{21}[2,3,5,9,13,14,15]$; $W_{20}[4,12]$
18	$STK_{21}[5]$	$2^{132} \times 2^{36} = 2^{168}$	$2^{168-7.1}$	$Z_{21}[2,3,14,15]$; $W_{20}[4,12,1,5,9]$
19	$STK_{21}[2]$	$2^{136} \times 2^{32} = 2^{168}$	$2^{172-7.7}$	$Z_{21}[3,15]$; $W_{20}[4,12,1,5,9,14]$
20	$STK_{21}[3]$	$2^{140} \times 2^{28} = 2^{168}$	$2^{172-7.7}$	$W_{20}[4,12,1,5,9,14,15]$
21	$STK_{20}[4]$	$2^{144} \times 2^{24} = 2^{168}$	$2^{172-7.7}$	$Z_{20}[1,7,11,13,15]$; $W_{19}[8]$
22	$STK_{20}[1]$	$2^{148} \times 2^{20} = 2^{168}$	$2^{172-7.7}$	$Z_{20}[7,11,15]$; $W_{19}[8,13]$
23	$STK_{20}[7]$	$2^{152} \times 2^{12} = 2^{164}$	$2^{172-7.1}$	$W_{19}[8,13,7]$
24	–	$2^{152} \times 2^{4} = 2^{156}$	$2^{164-7.1}$	$W_{18}[6]$
25	$STK_{18}[5]$	$2^{156} \times 2^{4} = 2^{160}$	$2^{160-8.7}$	$W_{17}[1]$
26	–	$2^{156} \times 2^{4} = 2^{160}$	$2^{160-8.7}$	$X_{17}[1]$

(b) Recovery of $X_{17}[13]$ (in Figure 15) with total complexity $2^{167.1}$.

Step	Guessed	Keys×Data=Mem	Time	Stored Texts
0	–	$2^0 \times 2^{60} = 2^{60}$	2^{60}	$Z_{25}[0\text{–}15]$; $STK_{19}[3], STK_{21}[7], STK_{23}[1]$
1–4	$STK_{25}[1\text{–}7]$	$2^{28} \times 2^{60} = 2^{88}$	$2^{88-6.7}$	$STK_{19}[3], STK_{21}[7], STK_{23}[1]$; $W_{24}[0\text{–}15]$
5–8	$STK_{24}[0\text{–}7]$	$2^{60} \times 2^{60} = 2^{120}$	$2^{120-6.7}$	$STK_{19}[3], STK_{21}[7], STK_{23}[1]$; $W_{23}[0\text{–}15]$
9–12	$STK_{23}[0,2\text{–}7]$	$2^{88} \times 2^{60} = 2^{148}$	$2^{148-6.7}$	$STK_{19}[3], STK_{21}[7]$; $W_{22}[0\text{–}7,9\text{–}15]$
13	$STK_{22}[1,5]$	$2^{96} \times 2^{60} = 2^{156}$	$2^{156-6.7}$	$Z_{22}[0,2\text{–}4,6\text{–}8,11,12,14,15]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13]$
14	$STK_{22}[2,6]$	$2^{104} \times 2^{60} = 2^{164}$	$2^{164-7.1}$	$Z_{22}[0,3,4,7,8,11,12,15]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14]$
15	$STK_{22}[3,7]$	$2^{112} \times 2^{52} = 2^{164}$	$2^{172-6.7}$	$Z_{22}[0,4,8,12]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14,7,15]$
16	$STK_{22}[0,4]$	$2^{120} \times 2^{52} = 2^{172}$	$2^{172-6.7}$	$STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14,7,15,0,4,8,12]$
17	–	$2^{120} \times 2^{44} = 2^{164}$	$2^{172-7.7}$	$Z_{21}[0,1,4,6,8,10,12,13,14]$; $STK_{19}[3]$; $W_{20}[11]$
18	$STK_{21}[0,4]$	$2^{128} \times 2^{36} = 2^{164}$	$2^{172-6.7}$	$Z_{21}[1,6,10,13,14]$; $STK_{19}[3]$; $W_{20}[11,4,12]$
19	$STK_{21}[1]$	$2^{132} \times 2^{32} = 2^{164}$	$2^{168-7.7}$	$Z_{21}[6,10,14]$; $STK_{19}[3]$; $W_{20}[11,4,12,13]$
20	$STK_{21}[6]$	$2^{136} \times 2^{28} = 2^{164}$	$2^{168-7.1}$	$STK_{19}[3]$; $W_{20}[11,4,12,13,2,6]$
21	$STK_{20}[5]$	$2^{140} \times 2^{20} = 2^{160}$	$2^{168-7.1}$	$Z_{20}[2,7,14]$; $STK_{19}[3]$; $W_{19}[5]$
22	$STK_{20}[2]$	$2^{144} \times 2^{16} = 2^{160}$	$2^{164-7.7}$	$Z_{20}[7]$; $STK_{19}[3]$; $W_{19}[5,14]$
23	$STK_{20}[7]$	$2^{148} \times 2^{16} = 2^{164}$	$2^{164-8.7}$	$STK_{19}[3]$; $W_{19}[5,14,3]$
24	–	$2^{148} \times 2^{16} = 2^{164}$	$2^{164-8.7}$	$Z_{19}[3,15]$; $STK_{19}[3]$; $W_{18}[0]$
25	–	$2^{148} \times 2^{8} = 2^{156}$	$2^{164-7.7}$	$W_{18}[0,15]$
26	–	$2^{148} \times 2^{4} = 2^{152}$	$2^{156-7.7}$	$W_{17}[12]$
27	–	$2^{148} \times 2^{4} = 2^{152}$	$2^{152-8.7}$	$X_{17}[13]$

Fig. 16: ZC-based integral attack on 30 rounds of SKINNYe-v2.

Table 6: Complexity of partial-sum key-recovery for 30 rounds of SKINNYe-v2.

(a) Recovery of $X_{19}[2]$ (in Figure 16) with total complexity $2^{232-5.7} = 2^{226.3}$.

Step	Guessed	Keys×Data=Memo	Time	Stored Texts
0	–	$2^0 \times 2^{64} = 2^{64}$	2^{64}	$Z_{29}[0\text{–}15]$; $STK_{25}[6]$, $STK_{27}[5]$, $STK_{29}[3]$
1–4	$STK_{29}[0\text{–}2, 4\text{–}7]$	$2^{28} \times 2^{64} = 2^{92}$	$2^{92-6.9}$	$STK_{25}[6]$, $STK_{27}[5]$; $W_{28}[0\text{–}15]$
5–8	$STK_{28}[0\text{–}7]$	$2^{60} \times 2^{64} = 2^{124}$	$2^{124-6.9}$	$STK_{25}[6]$, $STK_{27}[5]$; $W_{27}[0\text{–}15]$
9–12	$STK_{27}[0\text{–}4, 6\text{–}7]$	$2^{88} \times 2^{64} = 2^{152}$	$2^{152-6.9}$	$STK_{25}[6]$; $W_{26}[0\text{–}15]$
13–16	$STK_{26}[0\text{–}7]$	$2^{120} \times 2^{64} = 2^{184}$	$2^{184-6.9}$	$STK_{25}[6]$; $W_{25}[0\text{–}15]$
17	$STK_{25}[0, 4]$	$2^{128} \times 2^{64} = 2^{192}$	$2^{192-6.9}$	$Z_{25}[1\text{–}3, 5\text{–}7, 9\text{–}11, 13\text{–}15]$; $STK_{25}[6]$; $W_{24}[0, 4, 8, 12]$
18	$STK_{25}[2]$	$2^{132} \times 2^{60} = 2^{192}$	$2^{196-6.9}$	$Z_{25}[1, 3, 5, 7, 9, 11, 13, 15]$; $W_{24}[0, 4, 8, 12, 2, 6, 14]$
19	$STK_{25}[3, 7]$	$2^{140} \times 2^{56} = 2^{196}$	$2^{200-6.9}$	$Z_{25}[1, 5, 9, 13]$; $W_{24}[0, 4, 8, 12, 2, 6, 14, 3, 7, 15]$
20	$STK_{25}[1, 5]$	$2^{148} \times 2^{56} = 2^{204}$	$2^{204-6.9}$	$W_{24}[0, 4, 8, 12, 2, 6, 14, 3, 7, 15, 1, 5, 9, 13]$
21	$STK_{24}[0, 4]$	$2^{156} \times 2^{56} = 2^{212}$	$2^{212-7.3}$	$Z_{24}[1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15]$; $W_{23}[0, 8, 12]$
22	$STK_{24}[1, 5]$	$2^{164} \times 2^{52} = 2^{216}$	$2^{220-7.3}$	$Z_{24}[2, 3, 6, 7, 10, 11, 14, 15]$; $W_{23}[0, 8, 12, 1, 13]$
23	$STK_{24}[2, 6]$	$2^{172} \times 2^{44} = 2^{216}$	$2^{224-6.9}$	$Z_{24}[3, 7, 11, 15]$; $W_{23}[0, 8, 12, 1, 13, 6, 14]$
24	$STK_{24}[3, 7]$	$2^{180} \times 2^{44} = 2^{224}$	$2^{224-6.9}$	$W_{23}[0, 8, 12, 1, 13, 6, 14, 3, 7, 11, 15]$
25	$STK_{23}[0]$	$2^{184} \times 2^{40} = 2^{224}$	$2^{228-7.9}$	$Z_{23}[1, 3, 5, 6, 9, 10, 13, 14, 15]$; $W_{22}[12]$
26	$STK_{23}[1, 5]$	$2^{192} \times 2^{32} = 2^{224}$	$2^{232-6.9}$	$Z_{23}[3, 6, 10, 14, 15]$; $W_{22}[12, 5, 13]$
27	$STK_{23}[6]$	$2^{196} \times 2^{32} = 2^{228}$	$2^{228-7.3}$	$Z_{23}[3, 15]$; $W_{22}[12, 5, 13, 2, 6, 10]$
28	$STK_{23}[3]$	$2^{200} \times 2^{28} = 2^{228}$	$2^{232-7.9}$	$W_{22}[12, 5, 13, 2, 6, 10, 15]$
29	$STK_{22}[4]$	$2^{204} \times 2^{20} = 2^{224}$	$2^{232-7.3}$	$Z_{22}[2, 5, 13, 14]$; $W_{21}[4]$
30	$STK_{22}[5]$	$2^{208} \times 2^{16} = 2^{224}$	$2^{228-7.9}$	$Z_{22}[2, 14]$; $W_{21}[4, 9]$
31	$STK_{22}[2]$	$2^{212} \times 2^{12} = 2^{224}$	$2^{228-7.9}$	$W_{21}[4, 9, 14]$
32	–	$2^{212} \times 2^{4} = 2^{216}$	$2^{224-7.3}$	$W_{20}[7]$
33	–	$2^{212} \times 2^{4} = 2^{216}$	$2^{216-8.9}$	$W_{19}[2]$
34	–	$2^{212} \times 2^{4} = 2^{216}$	$2^{216-8.9}$	$X_{19}[2]$

(b) Recovery of $X_{19}[14]$ (in Figure 16) with total complexity $2^{228-5.6} = 2^{222.4}$.

Step	Guessed	Keys×Data=Memo	Time	Stored Texts
0	–	$2^0 \times 2^{64} = 2^{64}$	2^{64}	$Z_{29}[0\text{–}15]$; $STK_{23}[4]$, $STK_{25}[6]$, $STK_{27}[5]$, $STK_{29}[3]$
1–4	$STK_{29}[0\text{–}2, 4\text{–}7]$	$2^{28} \times 2^{64} = 2^{92}$	$2^{92-6.9}$	$STK_{23}[4]$, $STK_{25}[6]$, $STK_{27}[5]$; $W_{28}[0\text{–}15]$
5–8	$STK_{28}[0\text{–}7]$	$2^{60} \times 2^{64} = 2^{124}$	$2^{124-6.9}$	$STK_{23}[4]$, $STK_{25}[6]$, $STK_{27}[5]$; $W_{27}[0\text{–}15]$
9–12	$STK_{27}[0\text{–}4, 6\text{–}7]$	$2^{88} \times 2^{64} = 2^{152}$	$2^{152-6.9}$	$STK_{23}[4]$, $STK_{25}[6]$; $W_{26}[0\text{–}15]$
13–16	$STK_{26}[0\text{–}7]$	$2^{120} \times 2^{64} = 2^{184}$	$2^{184-6.9}$	$STK_{23}[4]$, $STK_{25}[6]$; $W_{25}[0\text{–}15]$
17–20	$STK_{25}[0\text{–}5, 7]$	$2^{148} \times 2^{64} = 2^{212}$	$2^{212-6.9}$	$STK_{23}[4]$; $W_{24}[0\text{–}8, 10\text{–}15]$
21	$STK_{24}[0, 4]$	$2^{156} \times 2^{56} = 2^{212}$	$2^{220-6.9}$	$Z_{24}[1\text{–}3, 5\text{–}7, 9, 10, 13\text{–}15]$; $STK_{23}[4]$; $W_{23}[4, 12]$
22	$STK_{24}[2, 6]$	$2^{164} \times 2^{52} = 2^{216}$	$2^{220-6.9}$	$Z_{24}[1, 3, 5, 7, 9, 13, 15]$; $STK_{23}[4]$; $W_{23}[4, 12, 2, 6, 14]$
23	$STK_{24}[3, 7]$	$2^{172} \times 2^{48} = 2^{220}$	$2^{224-7.3}$	$Z_{24}[1, 5, 9, 13]$; $STK_{23}[4]$; $W_{23}[4, 12, 2, 6, 14, 11, 15]$
24	$STK_{24}[1, 5]$	$2^{180} \times 2^{48} = 2^{228}$	$2^{228-6.9}$	$STK_{23}[4]$; $W_{23}[4, 12, 2, 6, 14, 11, 15, 1, 5, 9, 13]$
25	–	$2^{180} \times 2^{40} = 2^{220}$	$2^{228-7.9}$	$Z_{23}[1, 2, 5, 7, 9, 11, 13, 14, 15]$; $W_{22}[8]$
26	$STK_{23}[1, 5]$	$2^{188} \times 2^{32} = 2^{220}$	$2^{228-6.9}$	$Z_{23}[2, 7, 11, 14, 15]$; $W_{22}[8, 5, 13]$
27	$STK_{23}[2]$	$2^{192} \times 2^{28} = 2^{220}$	$2^{224-7.9}$	$Z_{23}[7, 11, 15]$; $W_{22}[8, 5, 13, 14]$
28	$STK_{23}[7]$	$2^{196} \times 2^{24} = 2^{220}$	$2^{224-7.3}$	$W_{22}[8, 5, 13, 14, 3, 7]$
29	$STK_{22}[6]$	$2^{200} \times 2^{16} = 2^{216}$	$2^{224-7.3}$	$Z_{22}[3, 4, 15]$; $W_{21}[6]$
30	$STK_{22}[4]$	$2^{204} \times 2^{16} = 2^{220}$	$2^{220-8.9}$	$Z_{22}[3, 15]$; $W_{21}[6, 0]$
31	$STK_{22}[3]$	$2^{208} \times 2^{12} = 2^{220}$	$2^{224-7.9}$	$W_{21}[6, 0, 15]$
32	–	$2^{208} \times 2^{8} = 2^{216}$	$2^{220-7.9}$	$Z_{21}[5]$; $W_{20}[12]$
33	$STK_{21}[5]$	$2^{212} \times 2^{8} = 2^{220}$	$2^{220-8.9}$	$W_{20}[12, 1]$
34	$STK_{20}[1]$	$2^{216} \times 2^{4} = 2^{220}$	$2^{224-7.9}$	$W_{19}[13]$
35	–	$2^{216} \times 2^{4} = 2^{220}$	$2^{220-8.9}$	$X_{19}[14]$

I Application to SKINNYee

I.1 Specification

SKINNYee, proposed by Naito et al. [32], is a 64-bit tweakable block cipher with 128-bit key $K = K_0 \parallel K_1 \parallel K_2 \parallel K_3$ and 259-bit tweak. The design is closely related to SKINNY in a TK4 setting, where the tweak is split into a 256-bit tweak that serves as a SKINNY tweakey to generate 32-bit round subtweaks ST_r added to the top half of the state in round r, and a 3-bit domain separation tweak that influences the round constants. The domain separation tweak plays no role in our analysis and we assume it to be constant. The key generates 32-bit round keys $K_{r\%4}$ which are added to the bottom half of the state (see Figure 17).

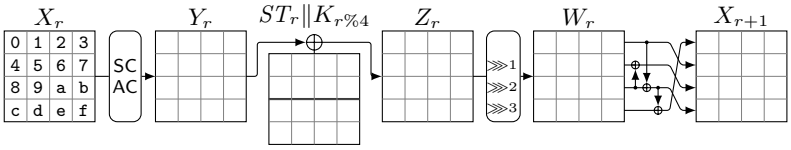

Fig. 17: Round function of SKINNYee

I.2 27-round Related-Tweak ID Attack on SKINNYee

This section explains our 27-round ID attack in the related-tweak setting on SKINNYee. Figure 18 illustrates the attack discovered by our tool. In this attack, we can obtain the equivalent key EK_0 by applying $EK_0 = \mathsf{MC}(\mathsf{SR}(K_{0\%4}))$ in the first round.

Pair Generation. We should build 2^x structures at W_0' and evaluate all possible values in 12 cells $W_0'[0, 1, 3\text{–}7, 9\text{–}11, 14, 15]$ for each structure, while the other cells assume a fixed value. By using 2^{x+48} plaintexts, we can have 2^{x+96-1} pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_B|-(n-|\Delta_F|)} = 2^{x+84}$ pairs. This step needs 2^{x+48+1} encryption calls.

Guess-and-Filter. For each of the N pairs we do the following steps:

a) *Satisfying round 27.* We guess $K_{26\%4}[3, 7]$ and compute the last column of ΔX_{26}. Checking if $\Delta X_{26}[3] = \Delta X_{26}[7] = \Delta X_{26}[15]$ will lead to two 4-bit filters (due to MC operation on W_{25}).We guess $K_{26\%4}[1, 5]$ and compute the second column of ΔX_{26}. Checking if $\Delta X_{26}[1] = \Delta X_{26}[13]$, and also $\Delta X_{26}[5] \oplus \Delta X_{26}[9] = \Delta X_{26}[1]$ will lead to two 4-bit filters.We guess $K_{26\%4}[0, 4]$ and compute the first column of ΔX_{26}. Checking if $\Delta X_{26}[4] = \Delta X_{26}[12]$ will lead to a 4-bit filter.We guess $K_{26\%4}[2, 6]$ and compute the

Fig. 18: ID attack on 27 rounds of SKINNYee in the related-tweak setting. $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 28 \cdot c$, $c_{\mathrm{B}} = 12 \cdot c$, $c_{\mathrm{F}} = 12 \cdot c$, $\Delta_{\mathrm{B}} = 12 \cdot c$, $\Delta_{\mathrm{F}} = 13 \cdot c$.

third column of ΔX_{26}. Now, we can determine ΔZ_{25} and Z_{25} as shown in Figure 18. Furthermore, checking if $\Delta Z_{25}[6] = \Delta ST_{25\%4}[6]$ will lead to a 4-bit filter. The time complexity of this step is about $N2^{12}$, and the number of tests left for the next step is $N2^8$.

b) *Satisfying round 26.* We guess $K_{25\%4}[5]$ and compute the second column of ΔX_{25}. Checking if $\Delta X_{25}[1] = \Delta X_{25}[5] = \Delta X_{25}[13]$ will lead to two 4-bit filters (due to MC operation on W_{24}). We guess $K_{25\%4}[3,7]$ and compute the last column of ΔX_{25}. Checking if $\Delta X_{25}[3] = \Delta X_{25}[11] = \Delta X_{25}[15]$ will lead to two 4-bit filters. We guess $K_{25\%4}[4]$. Now, we can determine ΔZ_{24} and Z_{24} as shown in Figure 18. The time complexity of this step is $N2^{12}$, and the number of tests left for the next step is $N2^8$.

c) *Satisfying round 25.* We guess $K_{24\%4}[5]$ and compute $\Delta X_{24}[1]$. Checking if $\Delta X_{24}[1] = \Delta X_{24}[13]$ will lead to a 4-bit filter (due to MC operation on the active cells in the second column of W_{23}). We guess $K_{24\%4}[1]$ and determine ΔZ_{23} and Z_{23} as shown in Figure 18. Furthermore, checking if $\Delta Z_{23}[4] = \Delta ST_{23\%4}[4]$ will lead to a 4-bit filter. The time complexity of this step is $N2^{12}$, and the number of tests left for the next step is $N2^8$.

d) *Satisfying round 1.* In this attack, we use the equivalent key EK_0 by applying $EK_0 = \mathsf{MC}(\mathsf{SR}(K_{0\%4}))$ in the first round. Therefore, $EK_0[0,1,2,3] = [K_{0\%4}[2] \oplus K_{0\%4}[5], K_{0\%4}[3] \oplus K_{0\%4}[6], K_{0\%4}[0] \oplus K_{0\%4}[7], K_{0\%4}[1] \oplus K_{0\%4}[4]]$, $EK_0[8,9,10,11] = K_{0\%4}[2,3,0,1]$, and $EK_0[12,13,14,15] = K_{0\%4}[2,3,0,1]$. We know the value of $K_{24\%4}[1](= K_{0\%4}[1])$ from the previous steps. Therefore, we will have $EK_0[11,15]$. The knowledge of $EK_0[11]$ enables to determine $\Delta Y_1[11]$ and thus $\Delta W_1[9]$. Also, without guessing any key, we can determine $\Delta Y_1[4]$ and so $\Delta W_1[5]$. Checking if $\Delta W_1[5] = \Delta W_1[9]$ will lead to a 4-bit filter (due to MC^{-1} operation on the active cells in the second column of X_2). Without guessing any key, we can determine $\Delta W_1[7]$. Next, we guess $EK_0[3,9]$ and determine $\Delta W_1[3,11]$. Here,by guessing $EK_0[9]$, we will have $EK_0[13]$ ($EK_0[9] = EK_0[13]$). Due to MC^{-1} operation on the active cells in the last column of X_2, we have $\Delta W_1[3] = \Delta W_1[7] = \Delta W_1[11]$. This will lead to two 4-bit filters. We guess $EK_0[0,10]$ and determine $\Delta W_1[0,8]$. Checking $\Delta W_1[0] = W_1[8]$ lead to a 4-bit filter. The knowledge of $EK_0[0]$ and $K_{24\%4}[5](= K_{0\%4}[5])$ allows us to determine $K_{0\%4}[2]$ (since $EK_0[0] = K_{0\%4}[5] \oplus K_{0\%4}[2]$) and so $EK_0[8,12]$. We guess $EK_0[1]$. The knowledge of $EK_0[14]$ (due to $EK_0[14] = EK_0[10]$) and $EK_0[1]$ lets us to determine $\Delta W_1[1,13]$. Also, without guessing any key, we can determine $\Delta W_1[5]$. Now, due to MC^{-1} operation on the active cells in the second column of X_2, we have $\Delta W_1[5] = \Delta W_1[1] \oplus \Delta W_1[13]$. This will lead to a 4-bit filter. Next, we guess $EK_0[2]$ and determine Y_1 and ΔY_1 as shown in Figure 18. The time complexity of this step is $N2^{12}$, and the number of tests left for the next step is $N2^{12}$.

e) *Satisfying rounds 2 and 3.* We guess $K_{1\%4}[0\text{–}2]$ and determine Y_2, and ΔY_2 as shown in Figure 18. We know $K_{2\%4}[0\text{–}5,7]$ from the previous steps. Therefore, we can determine ΔW_2, leading to three 4-bit filters (due to MC^{-1} on X_3). Therefore, we can determine ΔY_3 and the equality $\Delta Y_3[0] = \Delta ST_{3\%4}[0]$ will lead to a 4-bit filter. Due to MC^{-1} operation on X_4, we have $\Delta Y_3[6] =$

$\Delta Y_3[9] = \Delta Y_3[12]$ and so this will lead to two 4-bit filter. The time complexity of this step is $N2^{24}$, and the number of tests left for the next step is N.

f) *Satisfying round 4.* We guess $K_{3\%4}[0\text{–}2, 7]$ to obtain $\Delta Y_5[2]$ and here we will have a 4-bit filter. The time complexity of this step is $N2^{16}$, and the number of tests left to verify the impossible distinguisher is $N2^{12}$.

Complexity. Analyzing N pairs has a time complexity of about $N2^{24} \cdot \frac{2}{27}$ 27-round encryptions. The attack needs a data complexity of $D = N2^{n-|\Delta_B|-|\Delta_F|} = 2^{60} g \ln 2$ ($N = 2^{c_B+c_F} g \ln 2$). The total time complexity is $T = D + N2^{24}\frac{2}{27} + 2^{128-g}$. Hence, to optimize the time complexity of the attack, we select $g = 5$. Thus, the data, time, and memory complexities of the attack on SKINNYee are $2^{61.79}$, $2^{123.04}$, and 2^{108}, respectively.

I.3 26-Round ZC-Integral Attack on SKINNYee

We apply a similar approach as in Section H to derive a ZC distinguisher for 18 rounds (labelled as rounds 1 to 18 in Figure 19) with an initial free round and a final key recovery phase over 7 rounds (19 to 25). The target tweakey cell is `f`, only active in $z = 4$ cells. Thus, we can convert it to an integral distinguisher [1] with data complexity $2^{4\cdot(16-4+4)} = 2^{64}$, where the values in the active input cells and the 4 tweak cells with index `f` iterate over all values. The rest of the tweak is constant, as is the secret 128-bit key.

Key recovery and complexity Key recovery works as in Section H, except that the key is now added to the lower half of the state, while the upper half is updated with the constant tweak, which only needs to be stored once and thus does not impact the memory complexity. The procedures for both sums of $X_{19}[2]$ and $X_{19}[14]$ are summarized in Table 7. One repetition of the attack has an overall time complexity of about $2 \cdot 2^{108.9} = 2^{108.9}$ encryptions and reduces the key space to $2^{128-4} = 2^{124}$ candidates; with 4 repetitions, only 2^{112} candidates remain to brute-force. The intersection of these sets can be done relatively efficiently, so the overall complexity is about $4 \cdot 2^{109.9} + 2^{112} = 2^{113}$ encryption equivalents and $4 \cdot 2^{64} = 2^{66}$ data.

Fig. 19: ZC-based integral attack on 26 rounds of SKINNYee.

Table 7: Complexity of partial-sum key-recovery for 26 rounds of SKINNYee.

(a) Recovery of $X_{19}[2]$ (in Figure 19) with total complexity $2^{116-7.1} = 2^{108.9}$.

Step	Guessed	Keys × Data = Memo	Time	Stored Texts
0	–	$2^0 \times 2^{64} = 2^{64}$	2^{64}	$Z_{25}[0\text{–}15]$; $ST_{25}[6]$
1	$K_{25}[0,4]$	$2^8 \times 2^{64} = 2^{72}$	$2^{72-6.7}$	$Z_{25}[1\text{–}3,5\text{–}7,9\text{–}11,13\text{–}15]$; $ST_{25}[6]$; $W_{24}[0,4,8,12]$
2	$K_{25}[2,6]$	$2^{16} \times 2^{60} = 2^{76}$	$2^{80-6.7}$	$Z_{25}[1,3,5,7,9,11,13,15]$; $W_{24}[0,4,8,12,2,6,14]$
3	$K_{25}[3,7]$	$2^{24} \times 2^{56} = 2^{80}$	$2^{84-6.7}$	$Z_{25}[1,5,9,13]$; $W_{24}[0,4,8,12,2,6,14,3,7,15]$
4	$K_{25}[1,5]$	$2^{32} \times 2^{56} = 2^{88}$	$2^{88-6.7}$	$W_{24}[0,4,8,12,2,6,14,3,7,15,1,5,9,13]$
5	$K_{24}[4]$	$2^{36} \times 2^{56} = 2^{92}$	$2^{92-7.1}$	$Z_{24}[1,2,3,5,6,7,10,11,13,14,15]$; $W_{23}[0,8,12]$
6	$K_{24}[5]$	$2^{40} \times 2^{52} = 2^{92}$	$2^{96-7.1}$	$Z_{24}[2,3,6,7,10,11,14,15]$; $W_{23}[0,8,12,1,13]$
7	$K_{24}[2,6]$	$2^{48} \times 2^{44} = 2^{92}$	$2^{100-6.7}$	$Z_{24}[3,7,11,15]$; $W_{23}[0,8,12,1,13,6,14]$
8	$K_{24}[3,7]$	$2^{56} \times 2^{44} = 2^{100}$	$2^{100-6.7}$	$W_{23}[0,8,12,1,13,6,14,3,7,11,15]$
9	$K_{23}[4]$	$2^{60} \times 2^{40} = 2^{100}$	$2^{104-7.7}$	$Z_{23}[1,3,5,6,9,10,13,14,15]$; $W_{22}[12]$
10	$K_{23}[1,5]$	$2^{68} \times 2^{32} = 2^{100}$	$2^{108-6.7}$	$Z_{23}[3,6,10,14,15]$; $W_{22}[12,5,13]$
11	$K_{23}[7]$	$2^{72} \times 2^{28} = 2^{100}$	$2^{104-7.7}$	$Z_{23}[6,10,14]$; $W_{22}[12,5,13,15]$
12	$K_{23}[2,6]$	$2^{80} \times 2^{28} = 2^{108}$	$2^{108-7.1}$	$W_{22}[12,5,13,15,2,6,10]$
13	$K_{22}[0,4]$	$2^{88} \times 2^{20} = 2^{108}$	$2^{116-7.1}$	$Z_{22}[2,5,13,14]$; $W_{21}[4]$
14	$K_{22}[5]$	$2^{92} \times 2^{16} = 2^{108}$	$2^{112-7.7}$	$Z_{22}[2,14]$; $W_{21}[4,9]$
15	$K_{22}[6]$	$2^{96} \times 2^{12} = 2^{108}$	$2^{112-7.7}$	$W_{21}[4,9,14]$
16	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{108-7.1}$	$W_{20}[7]$
17	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{100-8.7}$	$W_{19}[2]$
18	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{100-8.7}$	$X_{19}[2]$

(b) Recovery of $X_{19}[14]$ (in Figure 19) with total complexity $2^{116-7.1} = 2^{108.9}$.

Step	Guessed	Keys × Data = Memo	Time	Stored Texts
0	–	$2^0 \times 2^{64} = 2^{64}$	2^{64}	$Z_{25}[0\text{–}15]$; $ST_{23}[4]$, $ST_{25}[6]$
1–4	$K_{25}[0\text{–}7]$	$2^{32} \times 2^{64} = 2^{96}$	$2^{96-6.7}$	$ST_{23}[4]$; $W_{24}[0\text{–}8,10\text{–}15]$
5	$K_{24}[0,4]$	$2^{40} \times 2^{56} = 2^{96}$	$2^{104-6.7}$	$Z_{24}[1\text{–}3,5\text{–}7,9,10,13\text{–}15]$; $ST_{23}[4]$; $W_{23}[4,12]$
6	$K_{24}[2,6]$	$2^{48} \times 2^{52} = 2^{100}$	$2^{104-6.7}$	$Z_{24}[1,3,5,7,9,13,15]$; $ST_{23}[4]$; $W_{23}[4,12,2,6,14]$
7	$K_{24}[7]$	$2^{52} \times 2^{48} = 2^{100}$	$2^{104-7.1}$	$Z_{24}[1,5,9,13]$; $ST_{23}[4]$; $W_{23}[4,12,2,6,14,11,15]$
8	$K_{24}[1,5]$	$2^{60} \times 2^{48} = 2^{108}$	$2^{108-6.7}$	$ST_{23}[4]$; $W_{23}[4,12,2,6,14,11,15,1,5,9,13]$
9	$K_{23}[4]$	$2^{64} \times 2^{40} = 2^{104}$	$2^{112-7.7}$	$Z_{23}[1,2,5,7,9,11,13,14,15]$; $W_{22}[8]$
10	$K_{23}[1,5]$	$2^{72} \times 2^{32} = 2^{104}$	$2^{112-6.7}$	$Z_{23}[2,7,11,14,15]$; $W_{22}[8,5,13]$
11	$K_{23}[6]$	$2^{76} \times 2^{28} = 2^{104}$	$2^{108-7.7}$	$Z_{23}[7,11,15]$; $W_{22}[8,5,13,14]$
12	$K_{23}[3,7]$	$2^{84} \times 2^{24} = 2^{108}$	$2^{112-7.1}$	$W_{22}[8,5,13,14,3,7]$
13	–	$2^{84} \times 2^{24} = 2^{108}$	$2^{108-8.7}$	$Z_{22}[3,6,10,14,15]$; $W_{21}[0]$
14	$K_{22}[2,6]$	$2^{92} \times 2^{16} = 2^{108}$	$2^{116-7.1}$	$Z_{22}[3,15]$; $W_{21}[0,6]$
15	$K_{22}[7]$	$2^{96} \times 2^{12} = 2^{108}$	$2^{112-7.7}$	$W_{21}[0,6,15]$
16	–	$2^{96} \times 2^{8} = 2^{104}$	$2^{108-7.7}$	$Z_{21}[5]$; $W_{20}[12]$
17	–	$2^{96} \times 2^{8} = 2^{104}$	$2^{104-8.7}$	$W_{20}[12,1]$
18	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{104-7.7}$	$W_{19}[13]$
19	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{100-8.7}$	$X_{19}[14]$

J Integral Distinguishers for SKINNY, SKINNYe-v2, and SKINNYee

Here, we propose new integral distinguishers for SKINNY, SKINNYe-v2 and SKINNYee. We use our new automatic tool to discover these distinguishers. Our model maximizes the number of active cells at the input of the ZC distinguisher to minimize the data complexity of the corresponding integral distinguisher. The Figures 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31 illustrate our discovered ZC-Integral distinguishers. We convert the ZC to an integral distinguisher by inverting the activeness pattern at the ZC distinguisher's input. More precisely, plaintext words with active linear masks take a fixed value, and plaintext words with zero linear masks take all possible values. Moreover, the tweakey cells involved in the attack take all possible values, and the remaining tweakey cells take a fixed value. Consequently, the linear combination of the active output words forms a balanced Boolean function over the input set.

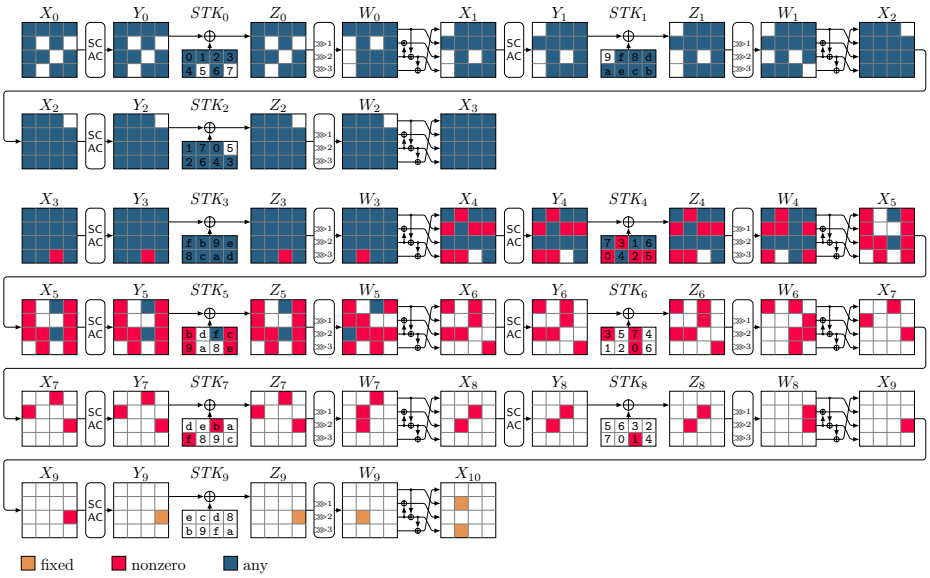

Fig. 20: ZC-Integral distinguisher for 10 rounds of SKINNY-n-n. $STK[5]$ is active at most one time. $X_{10}[5] \oplus X_{10}[13]$ is balanced. Data complexity: $2^{5 \cdot c}, c \in \{4, 8\}$.

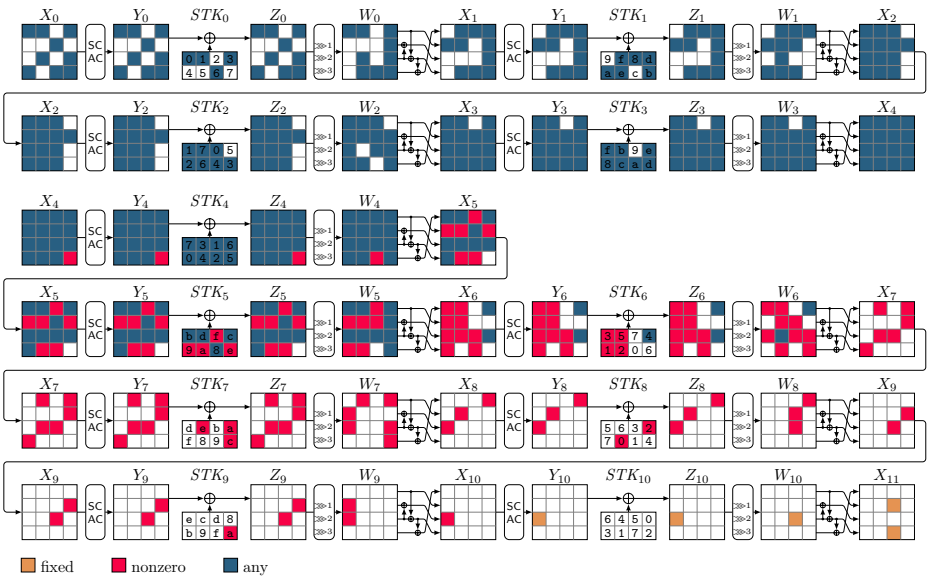

Fig. 21: ZC-Integral distinguisher for 11 rounds of SKINNY-n-n. $STK[9]$ is active at most one time. $X_{11}[6] \oplus X_{11}[14]$ is balanced. Data complexity: $2^{8 \cdot c}, c \in \{4, 8\}$.

Fig. 22: ZC-Integral distinguisher for 12 rounds of SKINNY-n-n. $STK[5]$ is active at most one time. $X_{12}[6] \oplus X_{12}[10] \oplus X_{12}[14]$ is balanced. Data complexity: $2^{13 \cdot c}, c \in \{4, 8\}$.

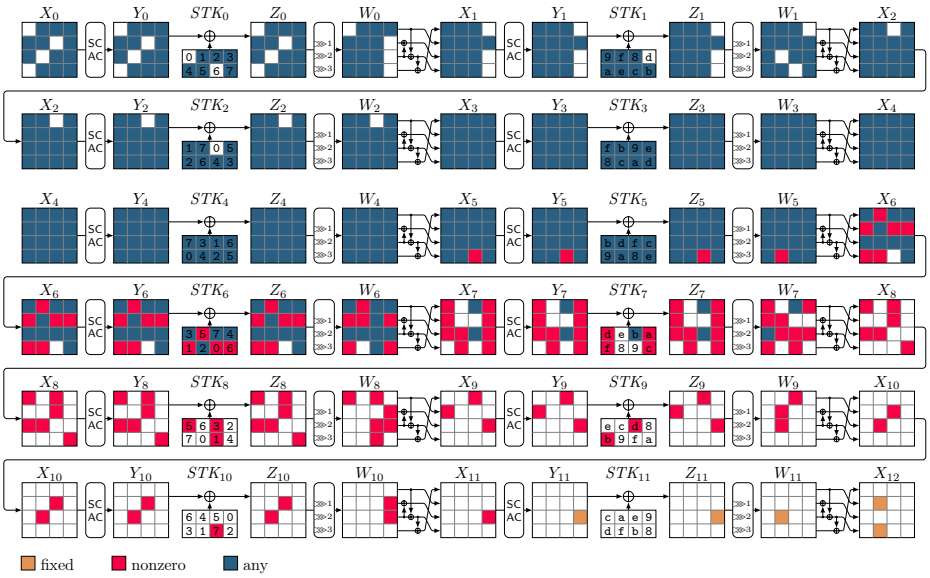

Fig. 23: ZC-Integral distinguisher for 12 rounds of SKINNY-n-$2n$. $STK[0]$ is active at most two times. $X_{12}[5] \oplus X_{12}[13]$ is balanced. Data complexity: $2^{6 \cdot c}, c \in \{4, 8\}$.

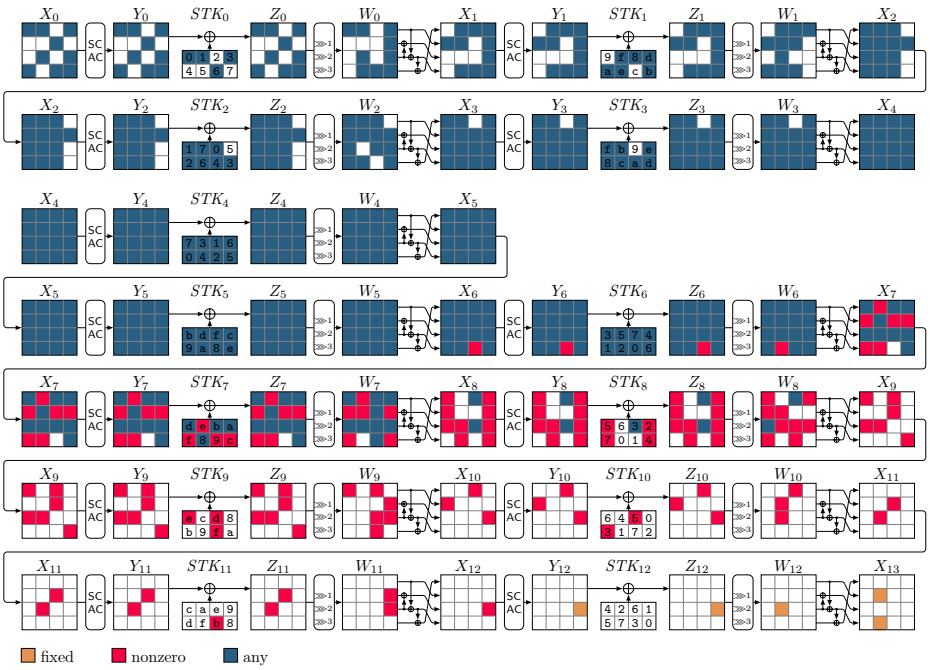

Fig. 24: ZC-Integral distinguisher for 13 rounds of SKINNY-n-$2n$. $STK[9]$ is active at most two times. $X_{13}[5] \oplus X_{13}[13]$ is balanced. Data complexity: $2^{9 \cdot c}, c \in \{4, 8\}$.

Fig. 25: ZC-Integral distinguisher for 14 rounds of SKINNY-n-$2n$. $STK[1]$ is active at most two times. $X_{14}[7] \oplus X_{14}[15]$ is balanced. Data complexity: $2^{14 \cdot c}, c \in \{4, 8\}$.

Fig. 26: ZC-Integral distinguisher for 14 rounds of SKINNY-n-$3n$. $STK[0]$ is active at most three times. $X_{14}[6] \oplus X_{14}[10] \oplus X_{14}[14]$ is balanced. Data complexity: $2^{7 \cdot c}$.

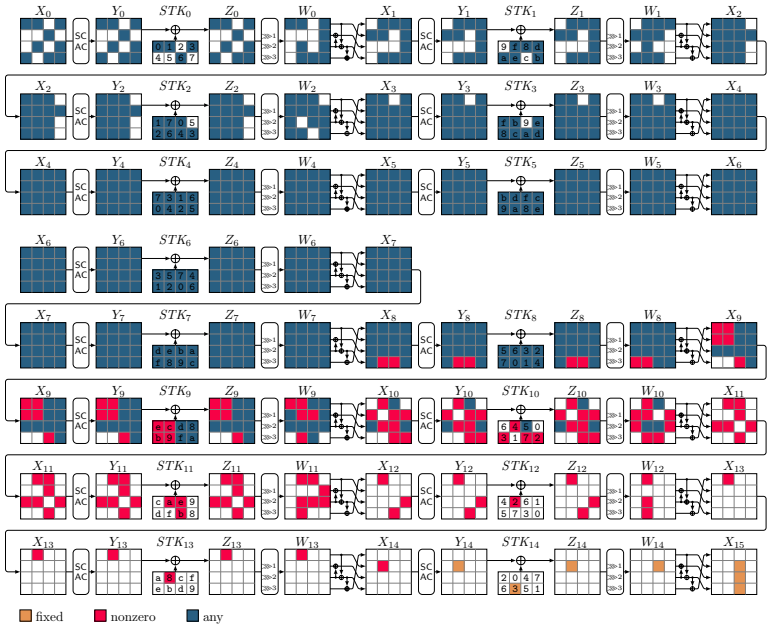

Fig. 27: ZC-Integral distinguisher for 15 rounds of SKINNY-n-$3n$. $STK[9]$ is active at most three times. $X_{15}[6] \oplus X_{15}[10] \oplus X_{15}[14]$ is balanced. Data complexity: $2^{10 \cdot c}$.

Fig. 28: ZC-Integral distinguisher for 16 rounds of SKINNY-n-$3n$. $STK[1]$ is active at most three times. $X_{16}[7] \oplus X_{16}[15]$ is balanced. Data complexity: $2^{15 \cdot c}, c \in \{4, 8\}$.

Fig. 29: ZC-Integral distinguisher for 16 rounds of SKINNYee/SKINNYe-v2. $STK[1]$ is active at most four times. $X_{16}[7] \oplus X_{16}[15]$ is balanced. Data complexity: 2^{32}.

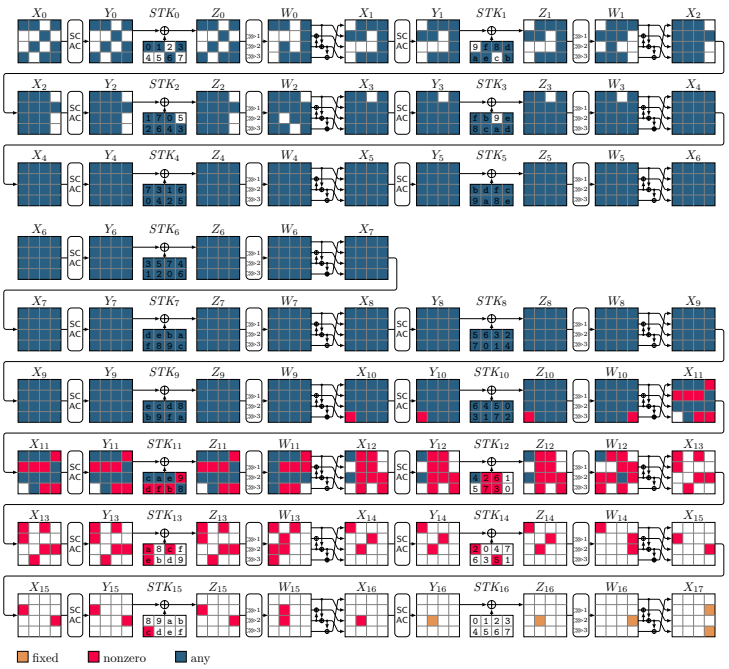

Fig. 30: ZC-Integral distinguisher for 17 rounds of SKINNYee/SKINNYe-v2. $STK[9]$ is active at most four times. $X_{17}[7] \oplus X_{17}[15]$ is balanced. Data complexity: 2^{44}.

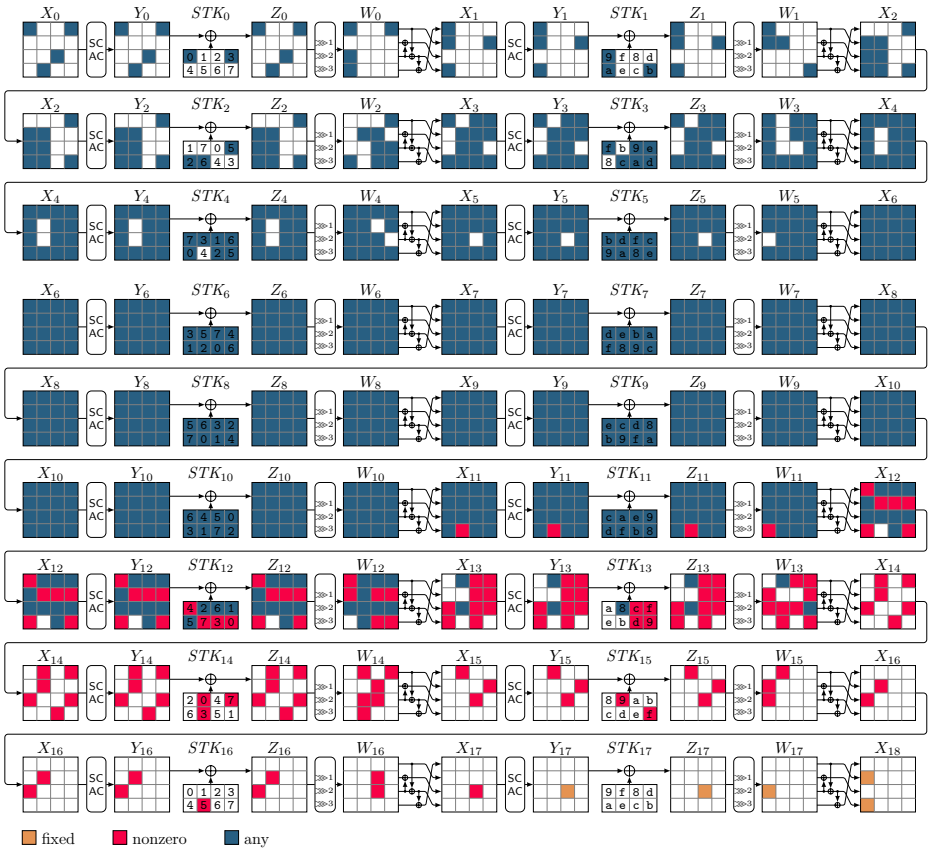

Fig. 31: ZC-Integral distinguisher for 18 rounds of SKINNYee/SKINNYe-v2. $STK[4]$ is active at most four times. $X_{18}[4] \oplus X_{18}[12]$ is balanced. Data complexity: 2^{64}.

K Application to CRAFT

K.1 Specification of CRAFT

CRAFT is a lightweight tweakable block cipher proposed in ToSC 2019 by Beirle et al. [4]. It receives a 64-bit plaintext, 128-bit key, plus a 64-bit tweak and applies 32 rounds to produce a 64-bit ciphertext. The internal state of CRAFT is arranged row-wise in a 4×4 array of nibbles. Figure 32 illustrates the round function of CRAFT which applies five basic operations to the internal state: MixColumns (MC), AddRoundConstant (ARC), AddTweakey (ATK), PermuteNibbles (PN), and S-box (SB).

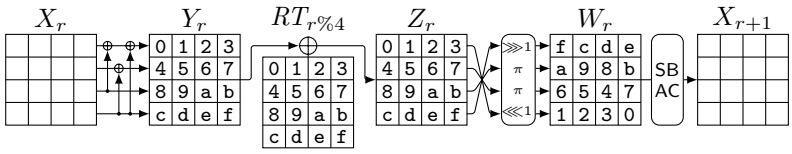

Fig. 32: Round function of CRAFT

The tweakey schedule of CRAFT receives a 128-bit key K and a 64-bit tweak T. It divides K into two halves $K_0||K_1$, and generates four round tweakeys $TK_0 = K_0 \oplus T$, $TK_1 = K_1 \oplus T$, $TK_2 = K_0 \oplus Q(T)$, and $TK_3 = K_1 \oplus Q(T)$, where Q is a nibble-wise permutation. Then, the ith round of CRAFT uses $TK_{i\%4}$ as the round tweakey. We refer the reader to [4] for more details.

K.2 Multidimensional ZC Attack on CRAFT

This section explains our multidimensional ZC attack on CRAFT. Our tool finds a ZC distinguisher for 13 rounds (labeled as rounds 3 (after MC) to 15 in Figure 33), combined with 3 initial rounds (0 to 2) and a final key recovery phase over 4 rounds (16 to 19). What follows explains the key recovery phase in detail.

Collect N pairs of plaintext and the corresponding ciphertexts. Guess 11 cells $RT_{0\%4}[0-2, 7, 9, 10, 14]$, and $RT_{1\%4}[3-5, 13]$, do the partial encryption and calculate $Y_3[5]$ for each pair. Allocate a 16-bit counter $N_0[Y_3, Z_{19}]$ for all 2^{48} possible value of $[Y_3, Z_{19}]$ and initialize it to zero. Next, compute the number of pairs of plaintext-ciphertext with given values Y_3, and Z_{19} and store it in $N_0[Y_3, Z_{19}]$. The time complexity of this step is equal to $N + (N \times 2^{44}) \times \frac{3}{20}$ 20-round encryptions.

Table 8 displays the details of each partial decryption step. In Table 8, the second column lists the guessed subkey cells in each step. The third column indicates the time complexity of the corresponding step. We calculate the values of the intermediate states and display them in the "Computed States" column. The counter $N[x_i]$ records the number of pairs that could produce the given

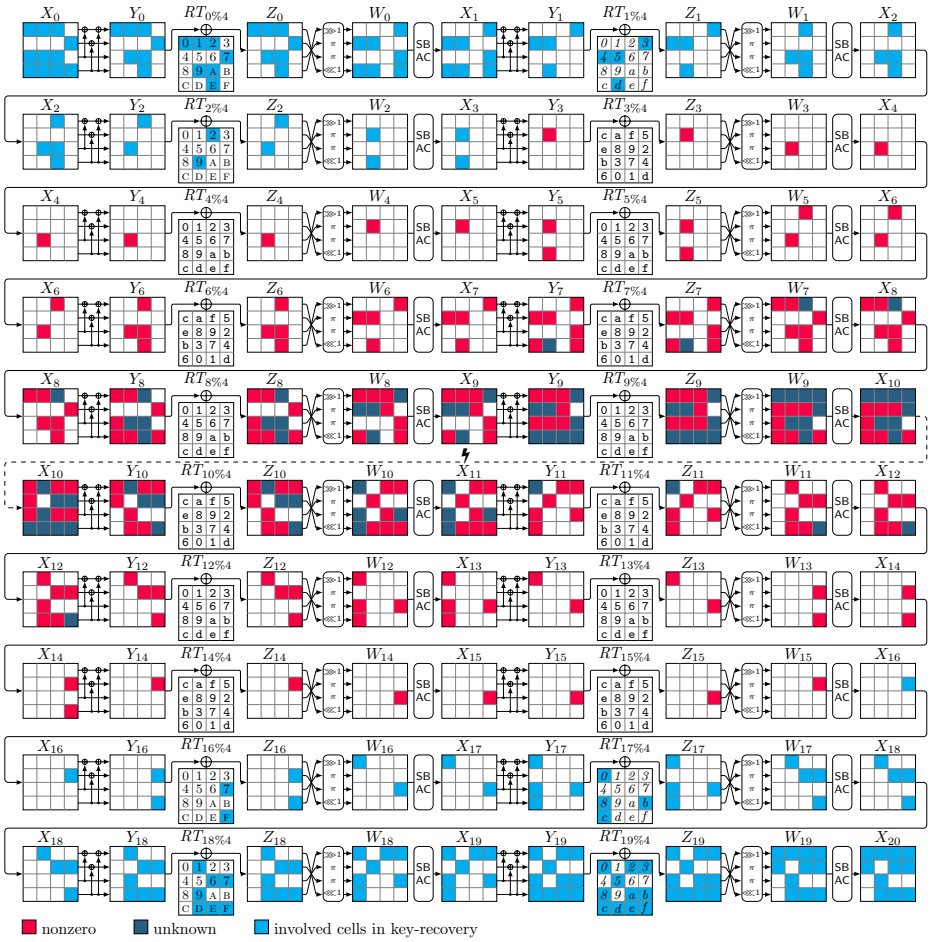

Fig. 33: ZC attack on 20 rounds of CRAFT. #Involved key cells: 22.

Table 8: Decryption procedure of the ZC attack on 20-round CRAFT.

Step	Guessed	Time	Computed States	Counter	Size
0	$RT_{0\%4}[0\text{–}2, 7, 9, 10, 14]$, $RT_{1\%4}[3\text{–}5, 13]$	$N \times 2^{44}$	$x_1 = Y_3[5] \vert Z_{19}[0, 2, 3, 5, 8, 10\text{–}15]$	$N_0[x_1]$	2^{48}
1	$RT_{19\%4}[0, 2, 8, 10\text{–}12, 14, 15]$	$2^{44+32} \times 2^{48}$	$x_2 = Y_3[5] \vert Z_{18}[1, 6, 7, 9, 13\text{–}15]$	$N_1[x_2]$	2^{32}
2	$RT_{18\%4}[6, 13, 15]$	$2^{76+12} \times 2^{32}$	$x_3 = Y_3[5] \vert Z_{17}[0, 8, 11, 12]$	$N_2[x_3]$	2^{20}
3	-	$2^{88} \times 2^{20}$	$x_4 = Y_3[5] \vert Z_{16}[7, 15]$	$N_3[x_4]$	2^{12}
4	-	$2^{88} \times 2^{12}$	$x_5 = Y_3[5] \vert X_{16}[7]$	$N_4[x_5]$	2^{8}

intermediate state for each possible value of x_i. In the last two columns, we show the counter and its size.

To recover the secret key, allocate a counter $V[z]$ for 8-bit z. For 2^8 values of $[Y_3, X_{16}]$, evaluate all 8 basis ZC masks on $[Y_3, X_{16}]$ and get z. Update the counter $V[z]$ by $V[z] = V[z] + N_4[Y_3, X_{16}]$. Calculate the statistical value T. If $T < \tau$, the guessed key values are possible right key candidates. The time complexity of this step is equal to $2^{88} \times 16 \times 2^8$ times of reading the 16-bit memory. Finally, do an exhaustive search for all the right candidates. The time complexity of this step is $\beta \times 2^{128}$.

Complexity. In this attack, we set the type-I error probability $\alpha = 2^{-2.7}$ and the type-II error probability $\beta = 2^{-9}$, then $Z_{1-\alpha} = 1.01$, and $Z_{1-\beta} = 2.88$. Thus, based on the Equation 5; $N = 2^{62.89}$. The decision threshold is $\tau = \mu_0 + \sigma_0 Z_{1-\alpha}$. If we consider one memory accesses as a one round, then the time complexity of our attack on 16-round CRAFT is about $2^{62.89} + (2^{106.89} \times \frac{3}{20}) + (2^{124} + 2^{120} + \cdots + 2^{100}) \times \frac{1}{20} + 2^{119} = 2^{120.43}$ 20-round encryptions. The required memory complexity is about 2^{49} bytes for counters.

K.3 21-round Impossible Differential Attack on CRAFT

Our tool finds an impossible differential distinguisher for 13 rounds (labeled as rounds 4 to 16 in Figure 34), combined with a key recovery phase over four initial rounds (rounds 0 to 3) and four final rounds (17 to 20).

Pair Generation. We should build 2^x structures at Y_0 and evaluate all possible values in eleven cells $Y_0[1\text{–}3, 7\text{–}10, 12\text{–}15]$ for each structure, while the other cells assume a fixed value. By using 2^{x+44} plaintexts, we can have 2^{x+88-1} pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_b|-1-(n-|\Delta_f|)} = 2^{x+67}$ pairs. This step needs 2^{x+44} encryption calls.

Guess-and-Filter. For each of the N pairs we do the following steps:

a) *Satisfying round 20.* we can determine ΔX_{20} as shown in Figure 34 without guessing any key. Therefore, we will have a 16-bit filter. Then, we guess $RT_{20\%4}[2, 6, 9, 10, 12, 14, 15]$ and determine $\Delta Y_{19}[0, 1, 5, 8, 13]$, and so $\Delta X_{19}[8, 13]$ that will lead to a 12-bit filters. Now, we guess $RT_{20\%4}[0, 1, 7, 8, 13]$ and determine Z_{19} and ΔZ_{19} as shown in Figure 34. The time complexity of this step is $N2^{20}$, and the number of tests left for the next step is $N2^{20}$.

b) *Satisfying round 1.* We know the values of $RT_{20\%4}[0\text{–}2, 6\text{–}10, 12\text{–}15]$ from the previous step. Therefore, we do not need to guess $RT_{0\%4}[0\text{–}2, 6\text{–}10, 12\text{–}15]$. Thus, we can determine $\Delta X_1[0\text{–}6, 11\text{–}13]$ as shown in Figure 34. Due to MC^{-1} operation on the active cells in the last three columns of Y_1, the equalities $\Delta X_1[1] = \Delta X_1[13]$, $\Delta X_1[2] = \Delta X_1[6]$, and $\Delta X_1[3] = \Delta X_1[11]$ will lead to a 12-bit filter. We guess $RT_{0\%4}[3]$ to determine $\Delta X_1[14]$ and filter a 4-bit (due to equality $\Delta X_1[14] = \Delta X_1[2]$). We guess $RT_{0\%4}[4]$ and determine Y_1

Fig. 34: ID attack on 21 rounds of CRAFT, $|k_B \cup k_F| = 25 \cdot c$, $c_B = 10 \cdot c$, $c_F = 10 \cdot c$, $\Delta_B = 11 \cdot c$, $\Delta_F = 11 \cdot c$

and ΔY_1 as shown in Figure 34. The time complexity of this step is $N2^{12}$, and the number of tests left for the next step is $N2^{12}$.

c) *Satisfying round 2.* We guess $RT_{1\%4}[0, 14]$ to determine $\Delta X_2[3, 15]$. Due to MC^{-1} operation on the active cells in the last column of Y_2, the equality $\Delta X_2[3] = \Delta X_2[15]$ will lead to a 4-bit filter. We guess $RT_{1\%4}[11]$ to determine $\Delta X_2[7]$. Due to MC^{-1} operation on the active cells in the last column of Y_2, the equality $\Delta X_2[7] = \Delta X_2[3]$ will lead to a 4-bit filter. We guess $RT_{1\%4}[4]$ to determine $\Delta X_2[10]$. Due to MC^{-1} operation on the ac-

tive cells in the third column of Y_2, we have $\Delta X_2[10] = \Delta X_2[2]$. From the knowledge of $\Delta X_2[2]$ and also $\Delta W_1[2]$, we can determine $W_1[2]$ and so $Z_1[13]$ by applying Lemma 1. Thus, we can determine $RT_{1\%4}[13]$ (due to $RT_{1\%4}[13] = Z_1[13] \oplus Y_1[13]$). The time complexity of this step is $N2^{20}$, and the number of tests left for the next step is $N2^{20}$.

d) *Satisfying round 3.* In this step, we can compute the cells $\Delta X_3[0, 4]$ (since we know $RT_{1\%4}[0, 4]$, and also $RT_{2\%4}[10, 15]$ based on the previous steps). On the other hand, We have $\Delta X_3[0] = \Delta X_3[4]$ due to the MC^{-1} operation on the active cells in the first column of Y_3. Checking if $\Delta X_3[0] = \Delta X_3[4]$ will lead to a 4-bit filter. The time complexity of this step is $N2^{20}$, and the number of tests left for the next step is $N2^{16}$.

e) *Satisfying round 20.* We guess $RT_{19\%4}[8]$. We also know the value of $RT_{19\%4}[13]$ from the previous steps. Hence, we can calculate $\Delta Y_{18}[2, 6]$. Due to MC operation on the active cells in the third column of X_{18}, the equality $\Delta Y_{18}[2] = \Delta Y_{18}[6]$ will lead to a 4-bit filter. The time complexity of this step is $N2^{20}$, and the number of tests left for the next step is $N2^{16}$.

f) *Satisfying round 20.* We guess $RT_{19\%4}[15]$. Due to MC operation on the active cells in the third column of X_{18}, we have $\Delta Y_{18}[2] = \Delta Y_{18}[14]$. Thus, we have $\Delta W_{18}[3] = \Delta W_{18}[13]$ and so we can determine $\Delta W_{18}[3]$ from the knowledge of $\Delta W_{18}[13]$. On the other hand, we can determine $\Delta X_{19}[3]$ from the knowledge of $\Delta Y_{19}[3]$ and $\Delta Y_{19}[11]$. Therefore, $\Delta W_{18}[3]$ and also $\Delta X_{19}[3]$, can help us to determine $X_{19}[3]$ by applying Lemma 1. Now, we can calculate $RT_{19\%4}[3]$ as $RT_{19\%4}[3] = Z_{19}[3] \oplus (Y_{19}[11] \oplus Y_{19}[15] \oplus X_{19}[3])$. The time complexity of this step is $N2^{20}$, and the number of tests left for the next step is $N2^{20}$.

g) *Satisfying rounds 20, 19, and 18.* We know the values of $RT_{19\%4}[0, 3, 8, 11, 13, 15]$ and also $RT_{18\%4}[7, 14, 15]$ from the previous steps. We guess $RT_{19\%4}[12]$ to determine $\Delta Y_{17}[3, 11]$. Due to MC operation on the active cells in the last column of X_{17}, the equality $\Delta Y_{17}[3] = \Delta Y_{17}[11]$ will lead to a 4-bit filter. The time complexity of this step is $N2^{24}$, and the number of tests left for the next step is $N2^{20}$.

h) *Satisfying rounds 2 and 3.* In this step, we can compute the cells $\Delta X_3[12]$ (since we know $RT_{2\%4}[1]$, and also $RT_{1\%4}[12]$ based on the previous steps). On the other hand, we have $\Delta X_3[0] = \Delta X_3[12]$ due to the MC^{-1} operation on the active cells in the first column of Y_3. Checking if $\Delta X_3[0] = \Delta X_3[12]$ will lead to a 4-bit filter (we also know $\Delta X_3[0]$ from the previous steps). We guess $RT_{1\%4}[2, 5]$. Now, we know the values of $RT_{1\%4}[0, 2\text{–}5, 11\text{–}14]$, $RT_{2\%4}[1, 2, 9, 10, 15]$, and also $RT_{3\%4}[5, 12]$ from the previous steps. Therefore, we can obtain $\Delta X_4[1, 9]$. Due to MC^{-1} operation on the active cells in the second column of Y_4, the equality $\Delta X_4[1] = \Delta X_4[9]$ will lead to a 4-bit filter. The time complexity of this step is $N2^{24}$, and the number of tests to verify the impossible distinguisher is $N2^{20}$.

Complexity. Analyzing N pairs has a time complexity of about $N2^{24}.\frac{5}{21}$ 21-round encryptions. The attack needs a data complexity of $D = 2^{57} g \ln 2$. The total time complexity is $T = D + N2^{24}\frac{5}{21} + 2^{128-g}$. Hence, to optimize the time

complexity of the attack, we select $g = 23$. Thus, the data, time, and memory complexities of the attack on CRAFT are $2^{60.99}$, $2^{106.53}$, and 2^{100}, respectively.

K.4 Integral Distinguishers for CRAFT

The Figures 35, 36, 37, and 38 represent our ZC-Integral distinguishers for 12 to 15 rounds of CRAFT. We use the same rule as in Section J to convert a ZC to an integral distinguisher. The behavior of the ZC distinguishers of CRAFT depends on the starting round. We denote the starting round by `Offset`, where `Offset` $\in \{0, 1, 2, 3\}$. For example, when `Offset` $= n$, the round tweakey TK_n is used in the first round of distinguisher.

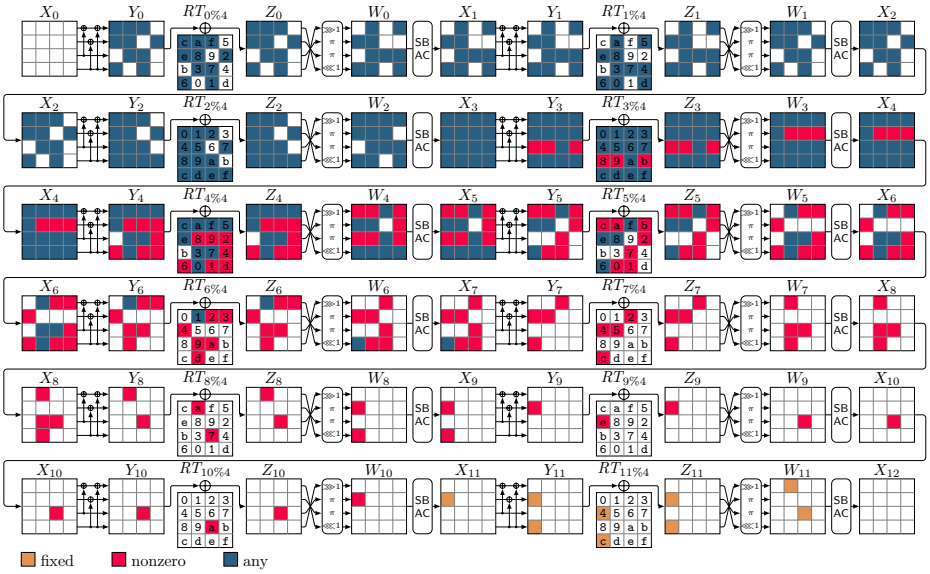

Fig. 35: ZC-Integral distinguisher for 12 rounds of CRAFT. `Offset` $= 2$. $RTK[\mathtt{b}]$ is active at most one time. $W_{11}[1] \oplus W_{11}[10]$ is balanced. Data complexity: 2^{28}.

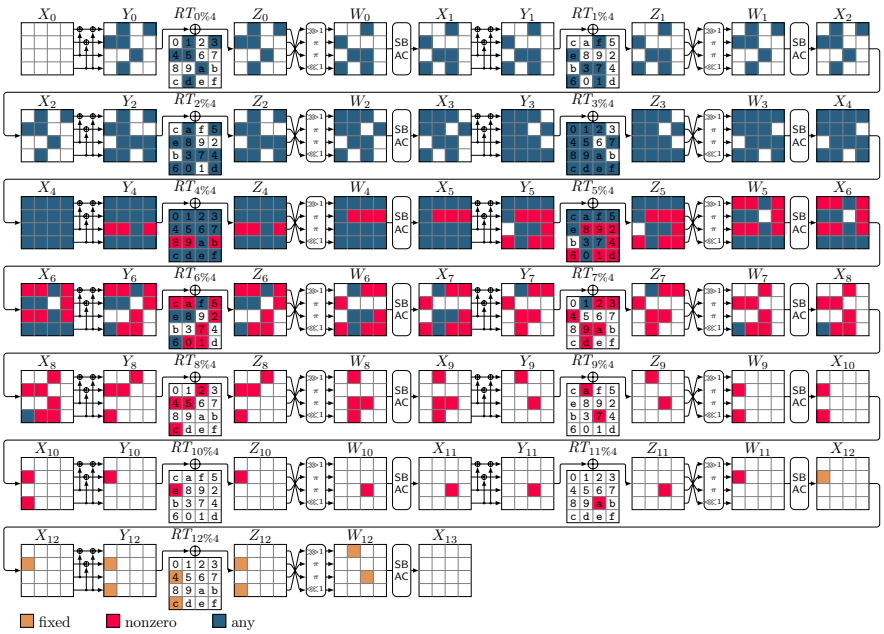

Fig. 36: ZC-Integral distinguisher for 13 rounds of CRAFT. `Offset` $= 1$. $RTK[\mathtt{b}]$ is active at most one time. $W_{12}[1] \oplus W_{12}[10]$ is balanced. Data complexity: 2^{44}.

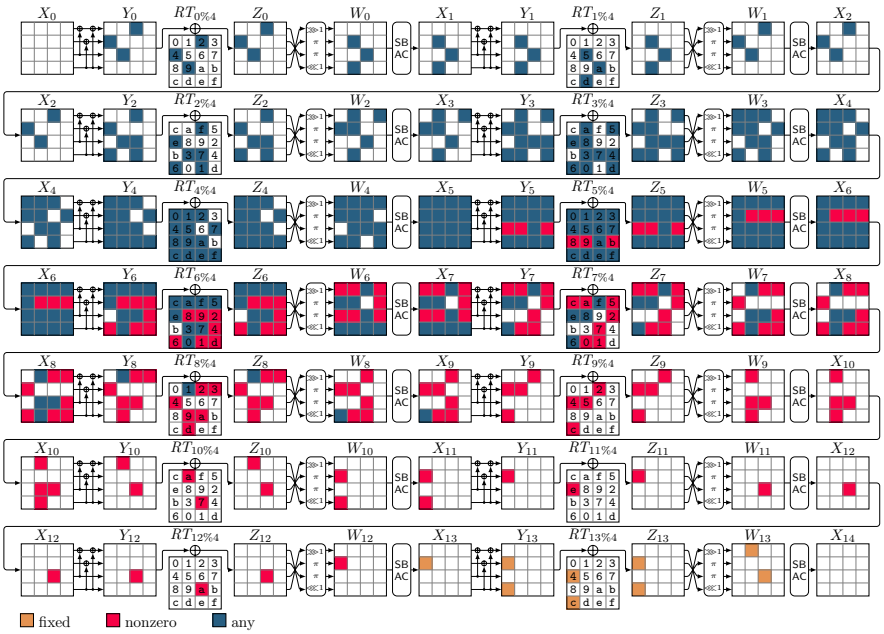

Fig. 37: ZC-Integral distinguisher for 14 rounds of CRAFT. `Offset` $= 0$. $RTK[\mathtt{b}]$ is active at most one time. $W_{13}[1] \oplus W_{13}[10]$ is balanced. Data complexity: 2^{56}.

Fig. 38: ZC-Integral distinguisher for 15 rounds of CRAFT. $\texttt{Offset} = 3$. $RTK[\texttt{b}]$ is active at most one time. $W_{14}[1] \oplus W_{14}[10]$ is balanced. Data complexity: 2^{64}.

L Application to Deoxys

Deoxys-BC is a tweakable block cipher based on the tweakey framework, which employs the round function of AES in the data path. Unlike the other ciphers in this paper, the internal state of Deoxys-BC is arranged **columnwise** in a 4×4 array of bytes. Deoxys-BC has two main versions: Deoxys-BC-256, and Deoxys-BC-384 that use 256-bit and 384-bit tweakey, respectively. We applied our tool to find the integral distinguisher to this block cipher. The Figures 39, 40, 41, and 42 illustrate the discovered ZC-based integral distinguishers for this cipher.

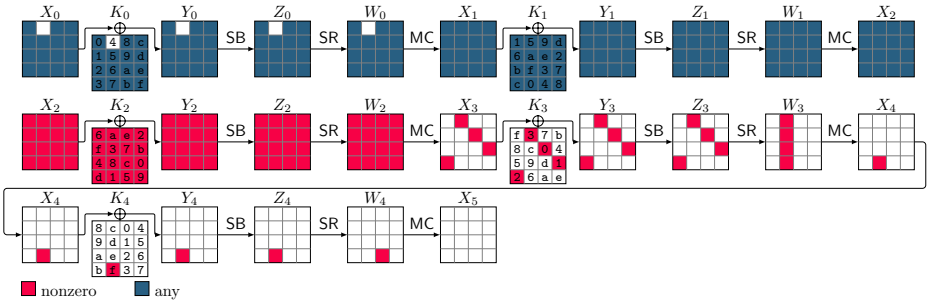

Fig. 39: ZC-Integral distinguisher for 5 rounds of Deoxys-BC-256. $RTK[4]$ is active at most two times. $W_4[11]$ is balanced. Data complexity: 2^{24}.

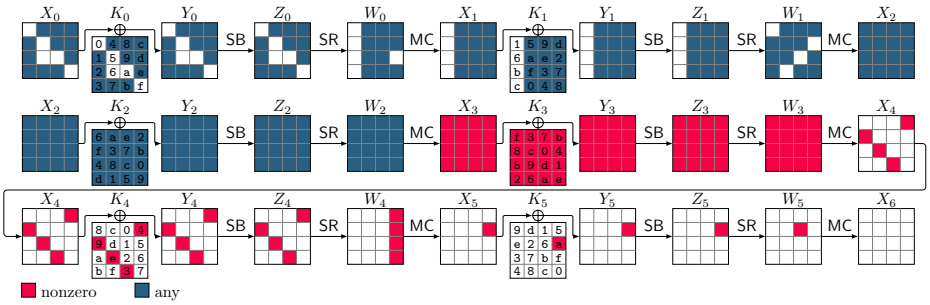

Fig. 40: ZC-Integral distinguisher for 6 rounds of Deoxys-BC-256. $RTK[6]$ is active at most two times. $W_5[9]$ is balanced. Data complexity: 2^{56}.

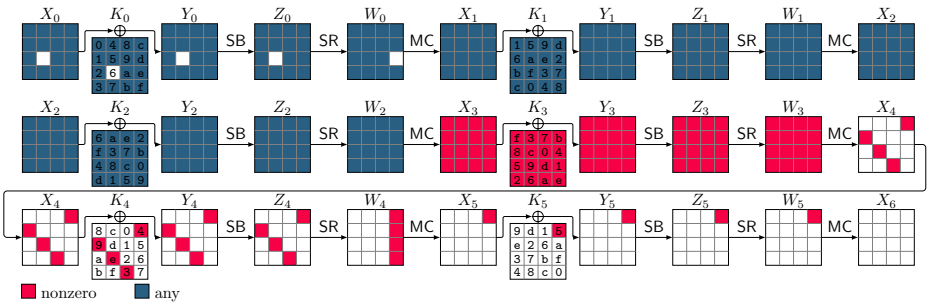

Fig. 41: ZC-Integral distinguisher for 6 rounds of Deoxys-BC-384. $RTK[6]$ is active at most three times. $W_5[12]$ is balanced. Data complexity: 2^{32}.

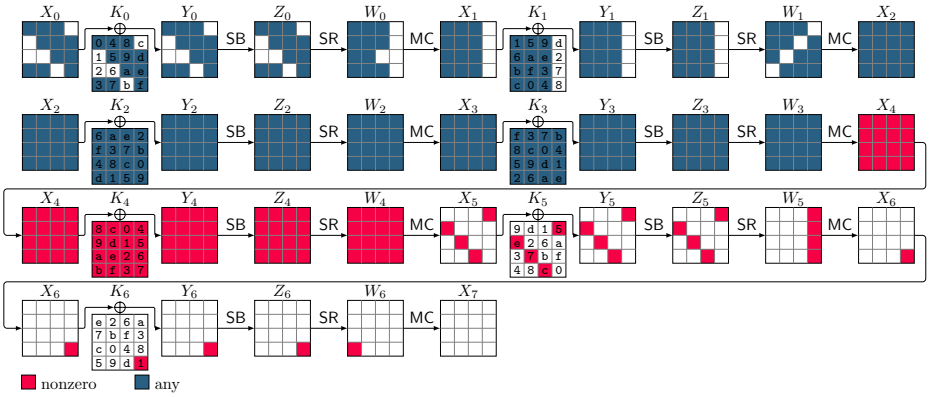

Fig. 42: ZC-Integral distinguisher for 7 rounds of Deoxys-BC-384. $RTK[2]$ is active at most three times. $W_6[3]$ is balanced. Data complexity: 2^{64}.

M Comparison of Our Method and Division Property

This section compares our method to search for integral distinguishers with division property (DP) or its enhanced version, i.e., monomial prediction (MP). We consider three aspects of automatic tools for finding integral distinguishers: The perfectness of the distinguisher model from the theoretical point of view (can it find all possible integral properties?), the efficiency of the model in practice, and the possibility of extending the distinguisher model to a unified model for finding the integral attack.

- As clarified in Section 2.3, the link between ZC and integral distinguisher is based on the definition of a balanced vectorial Boolean function. But DP/MP takes advantage of the algebraic degree/structure of the Boolean function to find integral distinguishers. Thus, we can not claim that our new automatic tool can find all the integral distinguishers theoretically discoverable by DP/MP. However, in what follows, we explain that its converse is also true in practice. Thus, both techniques are independently useful. Our tool can be even more useful for strongly aligned block ciphers.
- According to [22], MP is a theoretically perfect method to detect integral properties. But, it can be computationally hard to find a possible integral property with the DP/MP when it comes to practice. For example, while all output bits could be unbalanced, a linear combination of some bits/words can be balanced [26]. Still, checking all linear combinations for the input/output bits is computationally challenging. Thus, theoretically perfect methods such as DP/MP can miss the integral property due to limited search space.
- One of the significant advantages of our model is that it does not fix the input/output masks, and hence automatically considers all combinations of input/output words. We can also set an objective function to optimize a desired property for the input/output masks. In addition, taking the internal structure of S-boxes are not necessary for strongly aligned ciphers. Therefore, our word-oriented model seems sufficient to find a nearly optimum integral distinguisher/attack on strongly aligned block ciphers. As evidence, our 16-round (resp. 14-round) integral distinguisher for SKINNY-64-192 (resp. SKINNY-64-128) is better than the best previous one in [34] by a factor of 2^{12} in data complexity. In the previous models, such as [34] and DP, one has to (exhaustively) serach the space of input/output masks. For instance, if we want to find our integral distinguisher with DP or bit-wise ZC models in [34], the number of possible input masks with 60 active bits (48 bits in plaintext and 12 bits in tweak) is more than $\binom{64}{48} \approx 2^{48.79}$. Although some heuristic approaches have been proposed to search for minimum data complexity integral distinguishers based on DP, such as [15], the complexity of these methods still increases very quickly with the number of constant bits. Thus, we might have to try many configurations before finding the first one yielding the integral distinguisher. However, our method returns this low data complexity distinguisher in one run, taking less than a second on a regular laptop, without the need to check many different possible configurations.

– All of the SMT/SAT/MILP models for DP/MP rely on the unsatisfiability of the models. Conversely, the key recovery model is an optimization problem based on the underlying model's satisfiability. Consequently, building a unified SMT/SAT/MILP/CP model to find integral attacks using DP/MP is still impossible. However, our new model is based on the satisfiability of the model, i.e., any solution of our distinguisher model yields an integral distinguisher. This is our method's main advantage, letting us extend our model for integral distinguisher to a unified model to find an integral attack (optimized for key recovery) in Section 5. This key feature is still not achievable by DP/MP techniques.
– Note that the conventional division property can not consider the (twea)key schedule since XORing with constant does not change the division property. In contrast, our ZC-integral tool can consider the key schedule to find integral distinguishers. Thanks to this feature, we found many new low data-complexity integral distinguishers in the single-key and chosen tweak setting for SKINNY (see Table 3). Only the new variants of DP, e.g., MP, can consider the key schedule. However, DP/MP may not be efficient enough for ciphers with large S-boxes or dense linear layers, e.g., AES, Deoxys, and SKINNY-128, particularly if we want to check different linear combinations of input/output individually. For example, according to [26], $2^{32}-1$ inequalities are needed to model the MDS of AES in DP. Furthermore, taking the internal structure of S-boxes is unnecessary for strongly aligned block ciphers such as SKINNY, AES, Deoxys, MANTIS, and CRAFT. Therefore, compared to DP/MP, our technique can be even more useful in the integral analysis of strongly aligned block ciphers.