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
{hossein.hadipour,maria.eichlseder}@iaik.tugraz.at

[2] Department of Mathematics, 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

C. Hazay and M. Stam (Eds.): EUROCRYPT 2023, LNCS 14007, pp. 128–157, 2023.
https://doi.org/10.1007/978-3-031-30634-1_5

1 Introduction

The impossible differential (ID) attack, independently introduced by Biham et al. [5] and Knudsen [25], 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 [26] 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 [19,23]. 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	[21, 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	[21, F.3]
	$23^{\dagger}$	$2^{155.60}$	$2^{73.20}$	2^{138}	Int †	180,SK/CP,CT	[1]
	26	2^{172}	2^{61}	2^{172}	Int	180,SK/CP,CT	[21, 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	[21, F.4]
SKINNY-128-384	**21**	$2^{372.82}$	$2^{122.81}$	2^{98}	ZC	STK/KP	[21, 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	[21, F.3]
	26	2^{344}	2^{121}	2^{340}	Int	360,SK/CP,CT	[21, 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	[21, 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	[21, 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	[21, F.2]
	$20^{\dagger}$	$2^{97.50}$	$2^{68.40}$	2^{82}	Int †	120,SK/CP,CT	[1]
	22	2^{110}	$2^{57.58}$	2^{108}	Int	120,SK/CP,CT	[21, H.1]
SKINNY-128-256	**19**	$2^{240.07}$	$2^{122.90}$	2^{98}	ZC	STK/KP	[21, 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	[21, F.2]
	22	2^{216}	$2^{113.58}$	2^{216}	Int	240,SK/CP,CT	[21, 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	[21, 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	[21, F.1]
SKINNY-128-128	**16**	$2^{122.79}$	$2^{122.30}$	$2^{74.80}$	ZC	STK/KP	[21, 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	[21, F.1]
CRAFT	**20**	$2^{120.43}$	$2^{62.89}$	2^{49}	ZC	STK/KP	[21, K.2]
	21	$2^{106.53}$	$2^{60.99}$	2^{100}	ID	STK/CP	[21, K.3]
SKINNYee	**26**	2^{113}	2^{66}	2^{108}	Int	SK/CP,CT	[21, I.3]
	27	$2^{123.04}$	$2^{62.79}$	2^{108}	ID	RTK/CP	[21, I.2]
SKINNYe-v2	**30**	2^{232}	2^{65}	2^{228}	Int	240,SK/CP,CT	[21, H.3]

Outline. We recall the background on ID and ZC attacks and review the link between ZC and integral attacks in Sect. 2. In Sect. 3, we show how to convert the problem of searching for ID and ZC distinguishers to a CSP problem. In Sect. 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 Sect. 5, and finally conclude in Sect. 6. For detailed attack procedures of all analyzed ciphers, we refer to the full version of our paper [21].

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

Cipher	Attack	#R	Setting/Model	Ref.	Flaw
SKINNY-n-n	ID	18	STK/CP	[45]	Sect. 4.2
SKINNY-n-$2n$	ID	20	STK/CP	[45]	Sect. 4.2
	ZC/Int †	22	SK/CP, CT	[48]	Sect. 3
SKINNY-n-$3n$	ID	22	STK/CP	[45]	Sect. 4.2
	ZC/Int †	26	SK/CP, CT	[48]	Sect. 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 full version of our paper for the specification of SKINNY and SKINNYe [21, C], CRAFT [21, K.1], and SKINNYee [21, I.1].

2.1 Impossible Differential Attack

The impossible differential attack was independently introduced by Biham et al. [5] and Knudsen [25]. 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 Fig. 1, assume that there is an impossible differential $\Delta_{\text{U}} \nrightarrow \Delta_{\text{L}}$ 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 $|\Delta_{\text{B}}|$ ($|\Delta_{\text{F}}|$) denotes

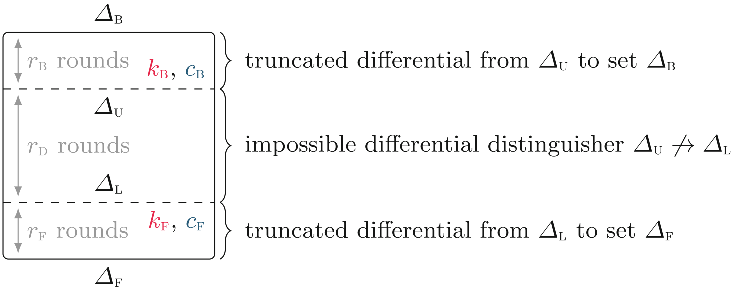
Δ_B
r_B rounds
k_B, c_B
truncated differential from Δ_U to set Δ_B
Δ_U
r_D rounds
Δ_L
impossible differential distinguisher $\Delta_U \not\to \Delta_L$
r_F rounds
k_F, c_F
truncated differential from Δ_L to set Δ_F
Δ_F

Fig. 1. Main parameters of the ID attack using an r_D-round impossible differential distinguisher $\Delta_U \not\to \Delta_L$. The distinguisher is extended with truncated differential propagation to sets $\Delta_U \to \Delta_B$ over r_B rounds backwards and $\Delta_L \to \Delta_F$ over r_F rounds forward. The inverse differentials $\Delta_B \to \Delta_U$ and $\Delta_F \to \Delta_L$ involve k_B, k_F key bits and have weight c_B, c_F, respectively.

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 $\Delta_B \to \Delta_U$ (resp. $\Delta_L \leftarrow \Delta_F$), i.e., $\Pr(\Delta_B \to \Delta_U) = 2^{-c_B}$ (resp. $\Pr(\Delta_L \leftarrow \Delta_F) = 2^{-c_F}$). 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 $x \oplus y \in \Delta_B$ and $E(x) \oplus E(y) \in \Delta_F$ and store them. This is a limited birthday problem, and according to [11] the complexity of this step is:

$$T_0 = \max\left\{\min_{\Delta \in \{\Delta_B, \Delta_F\}}\left\{\sqrt{N 2^{n+1-|\Delta|}}\right\}, N 2^{n+1-|\Delta_B|-|\Delta_F|}\right\} \quad (1)$$

- *Step 2: Guess-and-Filter.* The goal of this step is to discard all subkeys in $k_B \cup k_F$ which are invalidated by at least one of the generated pairs. Rather than guessing all subkeys $k_B \cup k_F$ at once and testing them with all pairs, we can optimize this step by using the *early abort* technique [29]: We divide $k_B \cup k_F$ 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]:

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

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

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 Eq. 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 [21, 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 of, Bogdanov et al. introduced the multidimensional ZC attack at ASIACRYPT 2012 [7]. We briefly recall the basics of multidimensional ZC attack.

Let E_{D} represent the reduced-round block cipher E with a block size of n bits. Assume that the correlation of m independent linear approximations $\langle u_i, x\rangle + \langle w_i, E_{\text{D}}(x)\rangle$ 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 $(x, y = E_{\text{D}}(x))$. 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 $z_i := \langle u_i, x\rangle + \langle w_i, E_{\text{D}}(x)\rangle$ 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., $y = E_{\text{D}}(x)$, 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, $E = E_{\text{F}} \circ E_{\text{D}} \circ E_{\text{B}}$. 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 $x = E_{\text{B}}(p)$ and output $y = E_{\text{F}}^{-1}(c)$ 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[1] 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 $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ 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*

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

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

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

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 [24] (see Fig. 2).

Theorem 2 (Ankele et al. [1]). *Let $E_K(T, P) : \mathbb{F}_2^{t\times n} \rightarrow \mathbb{F}_2^n$ 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.*

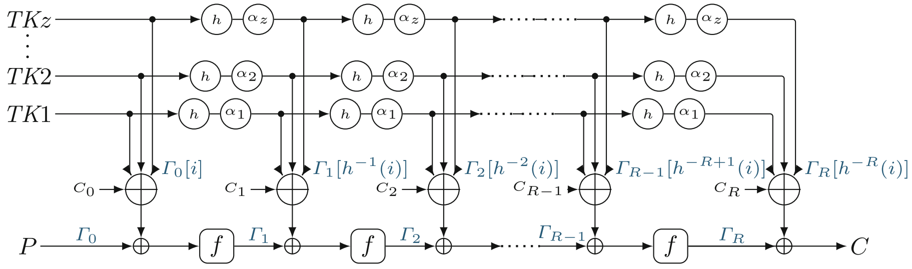

Fig. 2. The STK construction of the TWEAKEY framework.

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. [22] 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 $X_i \in \mathcal{D}_i$, $0 \leq i \leq n-1$; 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 $\mathtt{AX}[i]$ to encode the activeness pattern of $\Delta X[i]$ and another integer variable $\mathtt{DX}[i]$ to encode the actual c-bit difference value of $\Delta X[i]$:

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

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

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

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*

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

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

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

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

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

For encoding the MDS matrices, see [21, B]. To encode non-MDS matrices, such as the matrix employed in SKINNY, as described in [21, 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 Fig. 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 $XU_{r_{\text{U}}}$ (or XL_0) at the intersection of E_{U} and E_{L} is called the meeting point.

Let $\texttt{AXU}_r$ and $\texttt{AXL}_r$ denote the activeness pattern of the state variables XU_r and XL_r, as shown in Fig. 3. Let $\texttt{DXU}_r$ and $\texttt{DXL}_r$ 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 Sect. 2.5. We exclude trivial solutions by adding the constraints $\sum_{i=0}^{m-1} \texttt{AXU}_0[i] \neq 0$ and $\sum_{i=0}^{m-1} \texttt{AXL}_{r_{\text{L}}} \neq 0$. Let $\mathit{CSP}_{\text{U}}(\texttt{AXU}_0, \texttt{DXU}_0, \ldots, \texttt{AXU}_{r_{\text{U}}}, \texttt{DXU}_{r_{\text{U}}})$ be the model for propagation of deterministic truncated trails over E_{U} and $\mathit{CSP}_{\text{L}}(\texttt{AXL}_0, \texttt{DXL}_0, \ldots, \texttt{AXL}_{r_{\text{L}}}, \texttt{DXL}_{r_{\text{L}}})$ 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 additional constraints to ensure the incompatibility between the deterministic differential trails of E_{U} and E_{L} at the position of the meeting point:

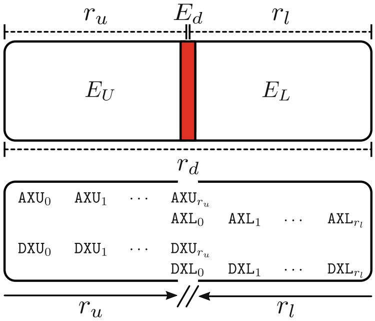
r_u
E_d
r_l
E_U
E_L
r_d
AXU$_0$ AXU$_1$ ··· AXU$_{r_u}$
AXL$_0$ AXL$_1$ ··· AXL$_{r_l}$
DXU$_0$ DXU$_1$ ··· DXU$_{r_u}$
DXL$_0$ DXL$_1$ ··· DXL$_{r_l}$
r_u
r_l

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

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

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 $\mathit{CSP}_\text{D} := \mathit{CSP}_\text{U} \wedge \mathit{CSP}_\text{L} \wedge \mathit{CSP}_M$, which is the union of all three CSPs. As a result, any feasible solution of CSP_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 CSP_U and CSP_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 CSP_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:

$$\mathit{CSP}_{TK}(\mathtt{ASTK}_0, \ldots, \mathtt{ASTK}_{r_{\text{D}}-1}) :=$$

$$\bigvee_{i=0}^{m-1} \left(\left(\begin{array}{c} \sum\limits_{r=0}^{r_{\text{D}}-1} \mathtt{bool2int}\left(\mathtt{ASTK}_r[h^{-r}(i)] \neq 0\right) \leq z \\ \wedge \bigvee\limits_{r=0}^{r_{\text{D}}-1} \left(\mathtt{ASTK}_r[h^{-r}(i)] = 1\right) \end{array} \right) \vee \left(\bigwedge_{r=0}^{r_{\text{D}}-1} \mathtt{ASTK}_r[h^{-r}(i)] = 0 \right) \right) \tag{7}$$

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 $\mathit{CSP}_{\text{D}} := \mathit{CSP}_{\text{U}} \wedge \mathit{CSP}_{\text{L}} \wedge \mathit{CSP}_{TK}$ to find ZC distinguishers of tweakable block ciphers taking the tweakey schedule into account. According to Eq. 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 [19,23], 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 [21, M].

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 Fig. 1, and outputs an optimized full ID attack for $r = r_{\text{B}} + r_{\text{U}} + r_{\text{L}} + r_{\text{F}}$ rounds of the targeted block cipher. To this end, we extend the CSP model for ID distinguishers in Sect. 3 to make a unified COP model for finding an optimized full ID attack taking all critical parameters affecting the final complexity into account.

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	$\mathbf{2^{5 \cdot c}}/\mathbf{2^{8 \cdot c}}/2^{13 \cdot c}$	[21, J]
SKINNY-n-$2n$	12/13/14	$\mathbf{2^{6 \cdot c}}/\mathbf{2^{9 \cdot c}}/2^{14 \cdot c}$	[21, J]
SKINNY-n-$3n$	14/15/16	$\mathbf{2^{7 \cdot c}}/2^{10 \cdot c}/2^{15 \cdot c}$	[21, J]
SKINNYe-v2/SKINNYee	16/17/18	$\mathbf{2^{32}}/2^{44}/2^{64}$	[21, J]
CRAFT	12/13/14/15	$\mathbf{2^{28}}/2^{44}/2^{56}/2^{64}$	[21, K.4]
Deoxys-BC-256	5/6	$\mathbf{2^{24}}/2^{56}$	[21, L]
Deoxys-BC-384	6/7	$\mathbf{2^{32}}/2^{64}$	[21, L]

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 Sect. 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 $\mathit{LG}(g) = \log_2(g) - 0.53$. Therefore, we can reformulate the complexity analysis of the ID attack as follows:

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

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 Eq. 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}, $|\Delta_{\mathrm{F}}|$, k_{F}. Moreover, to determine the actual size of $k_{\mathrm{B}} \cup k_{\mathrm{F}}$, 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 Sect. 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 $\Delta_{\text{B}} \xleftarrow{E_{\text{B}}^{-1}} \Delta_{\text{U}}$ and $\Delta_{\text{L}} \xrightarrow{E_{\text{F}}} \Delta_{\text{F}}$ 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 $\Delta_{\text{B}} \xrightarrow{E_{\text{B}}} \Delta_{\text{U}}$ and $\Delta_{\text{L}} \xleftarrow{E_{\text{F}}^{-1}} \Delta_{\text{F}}$.
- **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 Eq. 8 with the following constraints:

$$\begin{aligned}
&D[0] := min_{\Delta \in \{\Delta_{\text{B}}, \Delta_{\text{F}}\}}\{\tfrac{1}{2}(c_{\text{B}} + c_{\text{F}} + n + 1 - |\Delta| + LG(g))\},\\
&D[1] := c_{\text{B}} + c_{\text{F}} + n + 1 - |\Delta_{\text{B}}| - |\Delta_{\text{F}}| + LG(g),\\
&T[0] := max\{D[0], D[1]\}\,,\ \ T[0] < n,\\
&T[1] := c_{\text{B}} + c_{\text{F}} + LG(g),\ \ T[2] := |k_{\text{B}} \cup k_{\text{F}}| + LG(g),\ \ T[3] := k - g,\\
&T := max\{T[0], T[1], T[2], T[3]\},\ \ T < k.
\end{aligned} \tag{9}$$

  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 Eq. 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 *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 $E_{\text{B}}, E_{\text{U}}, E_{\text{L}}$, 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 [21, C] for the cipher specification). In what follows, given four integer numbers $r_{\text{B}}, r_{\text{U}}, r_{\text{L}}, r_{\text{F}}$, we model the full ID attack on $r = r_{\text{B}} + r_{\text{U}} + r_{\text{L}} + r_{\text{F}}$ rounds of SKINNY, where $r_{\text{D}} = r_{\text{U}} + r_{\text{L}}$ 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 $0 \le i \le 15$. 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.

Algorithm 1: CSP model for the tweakey schedule of SKINNY

```
Input: Four integer numbers $(r_B, r_U, r_L, r_F)$
Output: $CSP_{DTK}$
$R \leftarrow r_B + r_U + r_L + r_F - 1$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK1}[i] \in \{0, \ldots, 2^c - 1\} : 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK2}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \le r \le R, 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DTK3}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \le r \le R, 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{ASTK}_r[i] \in \{0, 1\} : 0 \le r \le R, 0 \le i \le 7\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DSTK}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \le r \le R, 0 \le i \le 7\}$;
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 $i \le 7$ 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 $\mathcal{M}$;
```

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 [21, algorithm 6], we build CSP_U

and CSP_{L}. We also build the CSP_{M} according to Eq. 6. The combined CSP model is $\mathit{CSP}_{\text{D}} := \mathit{CSP}_{\text{U}} \wedge \mathit{CSP}_{\text{L}} \wedge \mathit{CSP}_{M} \wedge \mathit{CSP}_{DTK}$. Hence, any feasible solution of CSP_{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 2: CSP_{U} for upper trail in distinguisher of SKINNY

```
   Input: $\mathit{CSP}_{DTK}$.var and the integer numbers $r_{\text{B}}, r_{\text{U}}$
   Output: $\mathit{CSP}_{\text{U}}$
1  $r_{\text{off}} \leftarrow r_{\text{B}}$;
2  Declare an empty CSP model $\mathcal{M}$;
3  $\mathcal{M}$.var $\leftarrow$ $\mathit{CSP}_{DTK}$.var;
4  $\mathcal{M}$.var $\leftarrow \{\mathtt{AXU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
5  $\mathcal{M}$.var $\leftarrow \{\mathtt{DXU}_r[i] \in \{-2,\ldots,2^c-1\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
6  $\mathcal{M}$.var $\leftarrow \{\mathtt{AYU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
7  $\mathcal{M}$.var $\leftarrow \{\mathtt{DYU}_r[i] \in \{-2,\ldots,2^c-1\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
8  $\mathcal{M}$.var $\leftarrow \{\mathtt{AZU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
9  $\mathcal{M}$.var $\leftarrow \{\mathtt{DZU}_r[i] \in \{-2,\ldots,2^c-1\} : \ 0 \le r \le r_{\text{U}}, \ 0 \le i \le 15\}$;
10 $\mathcal{M}$.con $\leftarrow \sum_{i=0}^{15} \mathtt{AXU}_0[i] + \sum_{i=0}^{15} \mathtt{DTK1}[i] + \sum_{i=0}^{15} \mathtt{DTK2}_0 + \sum_{i=0}^{15} \mathtt{DTK3}_0[i] \ge 1$;
11 for $r = 0,\ldots,r_{\text{U}}-1, \ i = 0,\ldots,15$ do
12     $\mathcal{M}$.con $\leftarrow$ $\mathit{Link}(\mathtt{AXU}_r[i],\mathtt{DXU}_r[i]) \wedge \mathit{Link}(\mathtt{AYU}_r[i],\mathtt{DYU}_r[i]) \wedge \mathit{Link}(\mathtt{AZU}_r[i],\mathtt{DZU}_r[i])$;
13 for $r = 0,\ldots,r_{\text{U}}-1, \ i = 0,\ldots,15$ do
14     $\mathcal{M}$.con $\leftarrow$ $\mathit{S\text{-}box}(\mathtt{AXU}_r[i], \mathtt{AYU}_r[i])$;
15 for $r = 0,\ldots,r_{\text{U}}-1, \ i = 0,\ldots,7$ do
16     $\mathcal{M}$.con $\leftarrow$ $\mathit{XOR}(\mathtt{AXU}_r[i], \mathtt{DXU}_r[i], \mathtt{ASTK}_{r_{\text{off}}+r}[i], \mathtt{DSTK}_{r_{\text{off}}+r}[i], \mathtt{AZU}_r[i], \mathtt{DZU}_r[i])$;
17     $\mathcal{M}$.con $\leftarrow$ $(\mathtt{AZU}_r[i+8] = \mathtt{AYU}_r[i+8]) \wedge (\mathtt{DZU}_r[i+8] = \mathtt{DYU}_r[i+8])$;
18 for $r = 0,\ldots,r_{\text{U}}-1, \ i = 0,\ldots,3$ do
19     $I_1 \leftarrow [\mathtt{AZU}_r[P[i]], \mathtt{AZU}_r[P[i+4]], \mathtt{AZU}_r[P[i+8]], \mathtt{AZU}_r[P[i+12]]]$;
20     $I_2 \leftarrow [\mathtt{DZU}_r[P[i]], \mathtt{DZU}_r[P[i+4]], \mathtt{DZU}_r[P[i+8]], \mathtt{DZU}_r[P[i+12]]]$;
21     $O_1 \leftarrow [\mathtt{AXU}_{r+1}[i], \mathtt{AXU}_{r+1}[i+4], \mathtt{AXU}_{r+1}[i+8], \mathtt{AXU}_{r+1}[i+12]]$;
22     $O_2 \leftarrow [\mathtt{DXU}_{r+1}[i], \mathtt{DXU}_{r+1}[i+4], \mathtt{DXU}_{r+1}[i+8], \mathtt{DXU}_{r+1}[i+12]]$;
23     $\mathcal{M}$.con $\leftarrow$ $\mathit{Mdiff}(I_1, I_2, O_1, O_2)$;
24 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 $\Delta_{\text{B}} \xleftarrow{E_{\text{B}}^{-1}} \Delta_{\text{U}}$, and $\Delta_{\text{L}} \xrightarrow{E_{\text{F}}} \Delta_{\text{F}}$, 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 ($\texttt{truncated} \xrightarrow{S} \texttt{fixed}$) 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 $\texttt{AX}, \texttt{AY}, \texttt{AZ} \in \{0, 1\}$ indicate whether the difference of X, Y, Z are zero. We define the new constraint XOR_1 to model the difference propagation with probability one through XOR:

$$\mathit{XOR}_1(\texttt{AX}, \texttt{AY}, \texttt{AZ}) := (\texttt{AZ} \geq \texttt{AX}) \wedge (\texttt{AZ} \geq \texttt{AY}) \wedge (\texttt{AZ} \leq \texttt{AX} + \texttt{AY}) \tag{10}$$

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 $0 \leq i \leq 19$. We also define the following constraint to encode the differential cancellation for each XOR operation:

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

Algorithm 3 and [21, algorithm 7] describe our model for difference propagation over E_{B} and E_{F}. We combine $\mathit{CSP}_{\mathrm{B}}^{dp}$ and $\mathit{CSP}_{\mathrm{F}}^{dp}$ into $\mathit{CSP}_{\mathrm{DP}} := \mathit{CSP}_{\mathrm{B}}^{dp} \wedge \mathit{CSP}_{\mathrm{F}}^{dp}$ 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 $\Delta_{\mathrm{B}} \rightarrow \Delta_{\mathrm{U}}$ and $\Delta_{\mathrm{L}} \leftarrow \Delta_{\mathrm{F}}$.

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 $0 \leq i \leq 3$ and $0 \leq r \leq r_u - 1$. 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 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

Algorithm 3: $\mathit{CSP}^{dp}_{\mathrm{B}}$ difference propagation through E_{B} for SKINNY

```
Input: $\mathit{CSP}_{DTK}.\mathtt{var}$, $\mathit{CSP}_{\mathrm{U}}.\mathtt{var}$ and the integer number $r_{\mathrm{B}}$
Output: $\mathit{CSP}^{dp}_{\mathrm{B}}$
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{DTK}.\mathtt{var}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AXB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}},\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AZB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{CB}_r[i] \in \{0,1\} :\ 0 \le r \le r_{\mathrm{B}} - 1,\ 0 \le i \le 19\}$;
for $i = 0, \ldots, 15$ do
    $\mathcal{M}.\mathtt{con} \leftarrow \textit{if}\ \mathtt{AXU}_0[i] \ge 1\ \textit{then}\ \mathtt{AXB}_{r_{\mathrm{B}}}[i] = 1\ \textit{else}\ \mathtt{AXB}_{r_{\mathrm{B}}}[i] = 0$;
for $r = 0, \ldots, r_{\mathrm{B}} - 1,\ i = 0, \ldots, 3$ do
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{Minvdiff}_1\left(\begin{pmatrix}\mathtt{AXB}_{r+1}[i]\\ \mathtt{AXB}_{r+1}[i+4]\\ \mathtt{AXB}_{r+1}[i+8]\\ \mathtt{AXB}_{r+1}[i+12]\end{pmatrix}, \begin{pmatrix}\mathtt{AZB}_r[P[i]]\\ \mathtt{AZB}_r[P[i+4]]\\ \mathtt{AZB}_r[P[i+8]]\\ \mathtt{AZB}_r[P[i+12]]\end{pmatrix}\right)$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AZB}_r[P[i+4]], \mathtt{AZB}_r[P[i+8]], \mathtt{AXB}_{r+1}[i+8], \mathtt{CB}_r[i])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AZB}_r[P[i]], \mathtt{AZB}_r[P[i+8]], \mathtt{AXB}_{r+1}[i+12], \mathtt{CB}_r[i+4])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AXB}_{r+1}[i+12], \mathtt{AZB}_r[P[i+12]], \mathtt{AXB}_{r+1}[i], \mathtt{CB}_r[i+8])$;
for $r = 0, \ldots, r_{\mathrm{B}} - 1,\ i = 0, \ldots, 7$ do
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_1(\mathtt{AZB}_r[i], \mathtt{ASTK}_r[i], \mathtt{AXB}_r[i])$;
    $\mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}_p(\mathtt{AXB}_r[i], \mathtt{ASTK}_r[i], \mathtt{AZB}_r[i], \mathtt{CB}_r[i+12])$;
    $\mathcal{M}.\mathtt{con} \leftarrow (\mathtt{AXB}_r[i+8] = \mathtt{AZB}_r[i+8])$;
return $\mathcal{M}$;
```

values of $X_r[i]$ and $Y_r[i]$ are needed. Then, we model the determination flow over the S-boxes as follows:

$$\textit{S-box}_{gd}(\mathtt{KXB}_r[i], \mathtt{KYB}_r[i], \mathtt{KDXB}_r[i]) := \begin{cases} (\mathtt{KYB}_r[i] \ge \mathtt{KXB}_r[i]) \wedge (\mathtt{KYB}_r[i] \ge \mathtt{KDXB}_r[i]) \wedge \\ (\mathtt{KYB}_r[i] \le \mathtt{KXB}_r[i] + \mathtt{KDXB}_r[i]) \end{cases}$$

We also model MixColumns according to [21, 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 $\mathtt{IKB}_r[i]$ be a binary variable that indicates whether the ith cell of subtweakey in the rth round of E_{B} is involved, where $0 \le r \le r_{\mathrm{B}} - 1$ and $0 \le i \le 15$. One can see that $\mathtt{IKB}_r[i] = 1$ if and only if $i \le 7$ and $\mathtt{KYB}_r[i] = 1$. Otherwise $\mathtt{IKB}_r[i] = 0$. We define binary variables $\mathtt{IKF}_r[i]$ to encode the involved subtweakey in E_{F} similarly. Algorithm 4 and [21, algorithm 8] describe our CSP models for the guess-and-determine through E_{B} and E_{F}. We refer to $\mathit{CSP}_{\mathrm{GD}} := \mathit{CSP}^{gd}_{\mathrm{B}} \wedge \mathit{CSP}^{gd}_{\mathrm{F}}$ 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

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

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

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 $\texttt{IK} \in \{0, \ldots, r_{\text{B}} + r_{\text{F}} - 1\}, \texttt{KE} \in \{0, 1, 2, 3\}$, and $\texttt{KS} \in \{0, \ldots, 48\}$. Then, assuming that $r_{\text{off}} = r_{\text{B}} + r_{\text{U}} + r_{\text{L}}$ and $z = 3$, we use the following constraints to model the key-bridging:

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

Model the Complexity Formulas. We now show how to combine all CSP models and model the complexity formulas. The variable KS in Eq. 12 determines the number of involved key cells, corresponding to $|k_{\text{B}} \cup k_{\text{F}}| = \texttt{c} \cdot \texttt{KS}$ 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 ([21, Figure 8], details in [21, F.2]) is 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 $2^{15.39}$ [21, 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], $c_{\text{B}} + c_{\text{F}}$ is supposed to be $18c$ for 22-round ID attack on SKINNY-n-$3n$ with cell size c. However, our tool returns $c_{\text{B}} = 6c$ and $c_{\text{F}} = 15c$, and thus $c_{\text{B}} + c_{\text{F}} = 21c$. Accordingly,

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

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

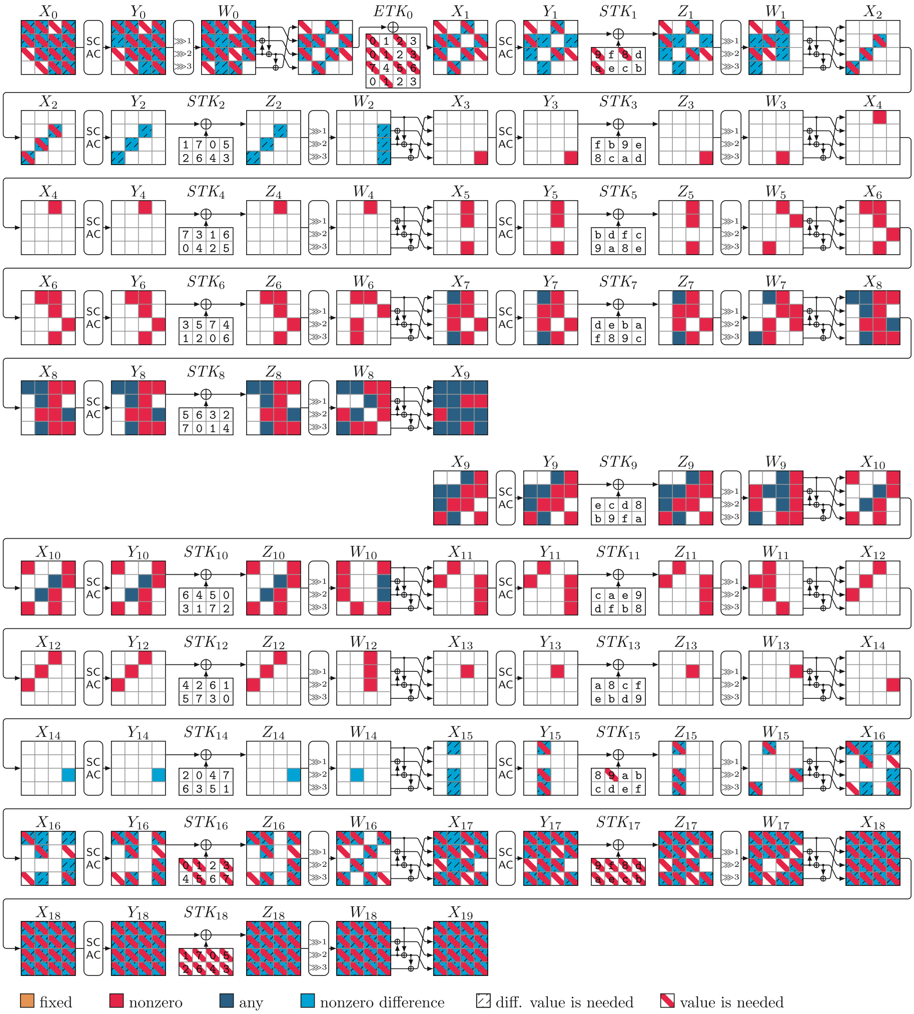

Fig. 4. ID attack on 19 rounds of SKINNY $[n\text{-}2n]$, $|k_{\text{B}} \cup k_{\text{F}}| = 26 \cdot c$, $c_{\text{B}} = 6 \cdot c$, $c_{\text{F}} = 15 \cdot c$, $\Delta_{\text{B}} = 7 \cdot c$, $\Delta_{\text{F}} = 16 \cdot c$

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 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 Sect. 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 [21, G, H, I.3]. As can be seen in [21, Figures 14, 15, 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 Trans. Symmetric Cryptol. **2019**(1), 192–235 (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, Yu., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (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: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (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: Knudsen, L. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48519-8_10
7. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: Wang, X., Sako, K. (eds.)

ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer, Heidelberg (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 Crypt. **70**(3), 369–383 (2012). https://doi.org/10.1007/s10623-012-9697-z
9. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 29–48. Springer, Heidelberg (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. J. Cryptol. **31**(1), 101–133 (2017). 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: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 179–199. Springer, Heidelberg (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 Cryptol, ePrint Archive, Report 2016/689 (2016), https://eprint.iacr.org/2016/689
13. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (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: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 157–184. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_6
15. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In: SAC. LNCS, vol. 11349, pp. 115–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-10970-7_6
16. Ferguson, N., et al.: Improved cryptanalysis of Rijndael. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001). 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, Cham (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., Eichlseder, M.: Finding the impossible: Automated search for full impossible differential, zero-correlation, and integral attacks. IACR Cryptology ePrint Archive, Report 2022/1147, p. 92 (2022). https://eprint.iacr.org/2022/1147
22. 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

23. 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: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_15
24. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_15
25. Knudsen, L.: Deal-a 128-bit block cipher. Complexity **258**(2), 216 (1998)
26. Lai, X.: Higher order derivatives and differential cryptanalysis, pp. 227–233 (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
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: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_22
29. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_24
30. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34704-7_5
31. Naito, Y., Sasaki, Yu., Sugawara, T.: Lightweight authenticated encryption mode suitable for threshold implementation. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 705–735. Springer, Cham (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, Cham (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: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74970-7_38
34. Niu, C., Li, M., Sun, S., Wang, M.: Zero-correlation linear cryptanalysis with equal treatment for plaintexts and tweakeys. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 126–147. Springer, Cham (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, Yu., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 185–215. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7

38. Sasaki, Yu., Wang, L.: Meet-in-the-middle technique for integral attacks against feistel ciphers. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 234–251. Springer, Heidelberg (2013). 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: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 3–34. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_1
40. Song, L., et al.: Optimizing rectangle attacks: A unified and generic framework for key recovery. In: ASIACRYPT 2022. LNCS, vol. 13791, pp. 410–440. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-22963-3_14
41. Sun, B., et al.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 95–115. Springer, Heidelberg (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 Trans. Symmetric Cryptol. **2020**(3), 262–287 (2020). https://doi.org/10.13154/tosc.v2020.i3.262-287
43. Sun, S., et al.: Analysis of aes, skinny, and others with constraint programming. IACR Trans. Symmetric Cryptol. **2017**(1), 281–306 (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, Cham (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: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 2017. LNCS, vol. 10239, pp. 117–134. Springer, Cham (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: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (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 Comput. Sci. **17**(174808 (2023)), 377–385 (2022). https://doi.org/10.1007/s11704-022-2206-2