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@iaik.tugraz.at, maria.eichlseder@iaik.tugraz.at
[2] Institute for Advanced Studies in Basic Sciences, Zanjan, Iran
s.sadeghi.khu@gmail.com,

Abstract. Impossible differential (ID) and zero-correlation (ZC) 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 distinguishers. However, previous CP-based methods are focused on finding only the ID or ZC distinguishers, and often only in a limited search space. Notably, none of them 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 and ZC 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 apply it to the ISO standard block cipher SKINNY and improve all of the existing ID, ZC, and integral attacks on it. 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-2n and SKINNY-n-n by 1 and 2 rounds, respectively. Applying our tool to discover ID attacks, we improve the ID attacks on all variants of SKINNY in the single-tweakey setting. Particularly, we improve the time complexity of the best previous single key ID attack on SKINNY-128-256 by a factor of $2^{22.57}$, while keeping the data and memory complexities much smaller. We also improve the ID attack on SKINNY-n-3n in the related-tweakey setting. Our method is generic and applicable to other word-oriented block ciphers.

Keywords: Cryptanalysis · Impossible differential attacks · Zero-correlation attacks · Integral attacks · SKINNY

1 Introduction

The impossible differential (ID) attack, independently introduced by Biham et al. [4] and Knudsen [21], 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 [24]. The ID attack exploits an impossible differential in a block cipher that usually originates from the slow diffusion to retrieve the master key. The zero correlation (ZC) attack, firstly introduced by Bogdanov and Rijmen [7], 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 [22] and as a practical attack by Daemen et al. [12]. The core idea of integral attack 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 [6]. Sun et al. at CRYPTO 2015 [34] developed further the links among the ID, ZC, and integral attacks. Thanks to the link developed between the ZC and integral attacks, we can use the techniques for finding the ZC distinguishers to find integral distinguishers. As another development of the ZC attack, Ankele et al. studied the influence of the tweakey schedule in ZC analysis of tweakable block ciphers at ToSC 2019 [1]. In particular, they showed that when searching for ZC approximations, taking the tweakey schedule into account can result in a longer ZC distinguisher. For instance, they used the links between the ZC and integral attacks and provided integral attacks on 20 and 23 rounds of SKINNY-64-128 and SKINNY-64-192, respectively, which have been the best single key attacks on these variants of SKINNY so far.

The search for ID, ZC, and integral attacks on a block cipher contain two main phases: finding a distinguisher and mounting a key recovery based on the discovered distinguisher. One of the main techniques to find the ID and ZC distinguishers is the miss-in-the-middle technique [4, 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. 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 enough and the number of possible solutions is extensive. 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 stem from cryptanalytic attacks is developing dedicated algorithms. For instance, in CRYPTO 2016 Derbez and Fouque proposed a dedicated algorithm [13] to find the $\mathcal{DS}$-MITM and ID attacks. However, developing efficient algorithms is difficult and implies a hard programming task to implement the approach as efficiently as possible. 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 the 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 [18, 26, 31, 33, 37]. For example, at EUROCRYPT 2017, Sasaki and Todo proposed a new automatic tool based on mixed integer linear programming (MILP) solvers to find the ID distinguishers [31]. Almost at the same time, Cui et al. proposed a similar approach to find the ID and ZC distinguishers [11]. As another development in this direction, Sun et al. recently proposed a new CP-based method to search for the ID and ZC distinguishers at ToSC 2020 [35].

Although the automatic methods to search for the ID and ZC attacks had significant advances over the past years, they have some basic limitations, which are summarized as follows:

- The CP models for finding the ID and ZC distinguishers proposed in [11,31] and [36] are the CP-based methods to search for differential (or liner) trails with an extra condition fixing the input/output differences (resp. linear masks). This approach requires an exhaustive search over all possible combinations for the input/output differences (or linear masks) to find all distinguishers. Thus, it is infeasible when the block size is large enough. Notably, this approach can not be extended to making a unified optimization problem to optimize the key recovery of the full ID and ZC attacks, since the input/output differences (or linear masks) are fixed in each instance of this model.
- The CP-based approach proposed in [35] employs a miss-in-the-middle-like technique to find the ID and ZC distinguishers. The main advantage of this approach is that it does not fix the input/output differences (or linear masks) for finding the ID (resp. ZC) distinguishers. Thus, it resolves the issue of incomplete search by transforming the exhaustive search into an inherent feature of the model. However, the compatibility between the two parts of the distinguisher is checked outside of the CSP model by iterating over a loop where the activeness pattern of a state cell at the meeting point should be fixed in each iteration. Notably, the method proposed in [35] is only focused on finding the longest distinguishers and does not consider the key recovery.
- The previous CP-based approaches regarding the ID, ZC, and integral attacks are only focused 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 in [13] is the only tool to find the full ID attack. However, this tool is based on a dedicated algorithm implemented in C/C++ and is not as generic and easy to use as the CP-based methods. In addition, this tool can not take all critical parameters of the ID attacks into account. Besides these limitations, this tool is not applicable for the ZC attacks and integral attacks based on the ZC distinguishers.
- None of the previous automatic tools takes the relationship between the ZC and integral attack into account to find ZC distinguishers suitable for the integral key recovery. Particularly, there is no automatic tool to take the meet-in-the-middle technique in integral key recovery into account for the integral attack based on the ZC distinguishers.

Our contributions. We propose a new generic and easy-to-use automatic method to address the above limitations to find the full ID and ZC attacks. We also propose a new method to automatically find the integral attacks based on ZC distinguishers taking the meet-in-the-middle technique into account. Unlike the method proposed in [13], our method is a CP-based method that can take advantage of the state-of-the-art CP/MILP/SAT solvers to solve multiple cryptanalytic problems. To show the usefulness of our method, we apply it to SKINNY and significantly improve all of the previous ZC, ID, and integral attacks on this cipher. Table 1 summarizes our discoveries regarding ID, ZC, and integral attacks on SKINNY. Notably, 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 so far. Regarding the ZC attacks, we improve the ZC attacks on SKINNY-n-n by 2 rounds. We also provide the first 19-round ZC attack on SKINNY, which covers the same number of rounds as the ID attack on this variant. Regarding the ID attacks, we could improve all previous ID attacks on all variants of SKINNY in the single tweakey setting. We also improved the related-tweakey ID attack on SKINNY-n-$3n$. The source code of our tool will be publicly available in the following Github repository: `https://github.com/hadipourh`.

Outline. We recall the background on the ID and ZC attacks and briefly review the link between the ZC and integral attacks in Section 2. We also briefly review the specification of SKINNY in Section 2. In Section 3, we show how to convert

Table 1: Summary of our cryptanalytic results. ID/ZC/Int = impossible differential, zero-correlation, integral. SK/RK = single/related key with given keysize, CP/KP = chosen/known plaintext, CT = chosen tweak.

Cipher	#R	Time	Data	Memory	Attack	Setting / Model	Ref.
SKINNY-64-192	**21**	$2^{185.83}$	$2^{62.63}$	2^{49}	ZC	SK / KP	C.2
	21	$2^{180.50}$	2^{62}	2^{170}	ID	SK / CP	[38]
	21	$\mathbf{2^{174.42}}$	$2^{62.43}$	2^{168}	ID	SK / CP	C.1
	23	$2^{155.60}$	$2^{73.20}$	2^{138}	Int	180,SK / CP,CT	[1]
	26	2^{172}	2^{61}	2^{172}	Int	180,SK / CP,CT	C.3
	27	2^{189}	$2^{63.53}$	2^{184}	ID	RK / CP	[23]
	27	$\mathbf{2^{183.83}}$	$2^{63.64}$	2^{110}	ID	RK / CP	C.1
SKINNY-128-384	**21**	$2^{372.82}$	$2^{122.81}$	2^{98}	ZC	SK / KP	C.2
	21	$2^{353.60}$	2^{123}	2^{341}	ID	SK / CP	[38]
	21	$\mathbf{2^{347.35}}$	$2^{122.89}$	2^{336}	ID	SK / CP	C.1
	26	2^{344}	2^{121}	2^{340}	Int	360,SK / CP,CT	C.3
	27	2^{378}	$2^{126.03}$	2^{368}	ID	RK / CP	[23]
	27	$\mathbf{2^{364.49}}$	$2^{124.86}$	2^{360}	ID	RK / CP	C.1
SKINNY-64-128	18	2^{126}	$2^{62.68}$	2^{64}	ZC	SK / KP	[30]
	19	$2^{119.12}$	$2^{62.89}$	2^{49}	ZC	SK / KP	C.2
	19	$2^{119.80}$	2^{62}	2^{110}	ID	SK / CP	[38]
	19	$\mathbf{2^{110.34}}$	$2^{60.86}$	2^{104}	ID	SK / CP	C.1
	20	$2^{97.50}$	$2^{68.40}$	2^{82}	Int	120,SK / CP,CT	[1]
	22	2^{110}	$2^{57.58}$	2^{108}	Int	120,SK / CP,CT	C.3
SKINNY-128-256	**19**	$2^{240.07}$	$2^{122.90}$	2^{98}	ZC	SK / KP	C.2
	19	$2^{241.80}$	2^{123}	2^{221}	ID	SK / CP	[38]
	19	$\mathbf{2^{219.23}}$	$2^{117.86}$	2^{208}	ID	SK / CP	C.1
	22	2^{216}	$2^{113.58}$	2^{216}	Int	240,SK / CP,CT	C.3
SKINNY-64-64	14	2^{62}	$2^{62.58}$	2^{64}	ZC	SK / KP	[30]
	16	$2^{62.71}$	$2^{61.35}$	$2^{37.80}$	ZC	SK / KP	C.2
	17	$2^{61.80}$	$2^{59.50}$	$2^{49.60}$	ID	SK / CP	[38]
	17	$\mathbf{2^{59}}$	$2^{58.79}$	2^{40}	ID	SK / CP	C.1
SKINNY-128-128	**16**	$2^{122.79}$	$2^{122.30}$	$2^{74.80}$	ZC	SK / KP	C.2
	17	$\mathbf{2^{116.51}}$	$2^{116.137}$	2^{80}	ID	SK / CP	C.1

the problem of searching for the ID and ZC distinguishers to a CSP problem whose feasible solutions correspond to ID and ZC distinguishers, respectively. In Section 4 we show how to extend our distinguisher models to create a unified model for finding optimum ID attacks. We discuss about the extension of our models for ZC and integral attacks in Section 5, and lastly conclude in Section 6.

2 Background

Here, we recall the basics of ID and ZC attacks and briefly review the link between the ZC and integral attacks. Moreover, we briefly review the specification of SKINNY and introduce the notations we use in the rest of this paper.

2.1 Impossible Differential Attack

The impossible differential attack was independently introduced by Biham et al. [4] and Knudsen [21]. 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. Thus, we can derive (some bits of) the correct key by discarding all the keys leading to the impossible differential. The goal is to discard as many wrong keys as possible. Lastly, we uniquely retrieve the key by exhaustively searching the remaining candidates.

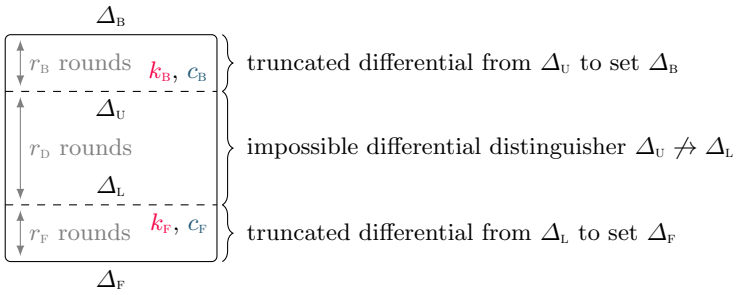

Fig. 1: Main parameters of the ID attack using an r_D-round impossible differential distinguisher $\Delta_U \not\to \Delta_L$. The distinguisher is extended with truncated differential propagation to sets $\Delta_U \to \Delta_B$ over r_B rounds backwards and $\Delta_L \to \Delta_F$ over r_F rounds forward. 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.

In what follows, we recall the complexity analysis of the ID attack based on [9,10]. Let E be a block cipher with n-bit block size and k-bit key. As illustrated in Figure 1, assume that there is an impossible differential $\Delta_U \nrightarrow \Delta_L$ 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 $|\Delta_B|$ ($|\Delta_F|$) denotes the dimension of vector space Δ_B (resp. Δ_F). Let c_B (c_F) be the number of bit-conditions that should be satisfied for $\Delta_B \to \Delta_U$ (resp. $\Delta_L \leftarrow \Delta_F$), i.e.,

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

Step 1: Pair Generation. In this step, given access to the encryption oracle (and possibly the decryption oracle), we generate N pairs $(x, y) \in \{0,1\}^{2n}$ such that $x \oplus y \in \Delta_{\mathrm{B}}$ and $E(x) \oplus E(y) \in \Delta_{\mathrm{F}}$. This is a limited birthday problem, and according to [10] the complexity of this step is:

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

We store the N generated pairs in memory for the next step.
Step 2: Guess-and-Filter. The goal of this step is to discard, among all the subkeys $k_{\mathrm{B}} \cup k_{\mathrm{F}}$, those which are invalidated by at least one of the generated pairs. Rather than guessing all subkeys $k_{\mathrm{B}} \cup k_{\mathrm{F}}$ at once and testing them with all pairs, we can optimize this step by using the *early abort* technique [25]. Accordingly, we divide $k_{\mathrm{B}} \cup k_{\mathrm{F}}$ into smaller subsets, typically the round keys, and guess them step by step. At each step, we reduce the size of the remaining pairs by checking if they satisfy the required conditions of truncated differential trail through E_{B} and E_{F}. The minimum number of partial encryptions/decryptions in this step is as follows [9]:

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

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

Suppose that $T_2 = N$, $T_3 = 2^{|k_{\mathrm{B}} \cup k_{\mathrm{F}}|} \frac{N}{2^{c_{\mathrm{B}}+c_{\mathrm{F}}}}$, and $T_3 = 2^k P$. 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 should have $T_0 < 2^n$. In addition, to retrieve at least one bit of key information in the guess-and-filter step, $P < \frac{1}{2}$ should hold. It is important to recall that Equation 2 is the average time complexity of the guess-and-filter step. Therefore, for each ID attack, we must compute the complexity of the guess-and-filter step accurately to ensure we meet this bound in practice. To see the complexity analysis of the ID attack in the related (twea)key setting see Section A.

2.2 Multidimensional Zero-Correlation Attack

Zero-correlation attack, firstly introduced by Bogdanov and Rijmen [7], is the dual of the ID attack in the context of linear analysis and exploits 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 [8], which utilizes multiple ZC linear approximations. However, multiple ZC attack relies on the assumption that all involved ZC linear approximations are independent, which limits its applications. To overcome the assumption of independent ZC linear approximations, Bogdanov et al. introduced the multidimensional ZC attack at ASIACRYPT 2012 [6]. We briefly recall the basics of multidimensional ZC attack.

Let E_{D} represents the reduced round of block cipher E with a block size of n bits. Assume that the correlation of m independent linear approximations $\langle u_i, x\rangle + \langle w_i, E_{\mathrm{D}}(x)\rangle$, and all their nonzero linear combinations are zero, where $u_i, w_i, x \in \mathbb{F}_2^n$, for $i = 0, \ldots, m-1$. We denote by l the number of ZC linear approximations, i.e., $l = 2^m$. In addition, let we are given N input/output pairs $(x, y = E_{\mathrm{D}}(x))$. Then, we can construct a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ which maps x to $z(x) = (z_0, \ldots, z_{m-1})$, where $z_i := \langle u_i, x\rangle + \langle w_i, E_{\mathrm{D}}(x)\rangle$ for all i. The idea of the multidimensional ZC distinguisher is that the output of this function follows the *multivariate hypergeometric distribution*, while the m-tuples of bits drawn at random from a uniform distribution on $\mathbb{F}_2^m$ follows *multinomial distribution* [6]. For sufficiently large N, we can distinguish E_{D} from a random permutation as follows.

We initialize 2^m counters $V[z]$ to zero for all $z \in \mathbb{F}_2^m$. Then, for each pair (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 the counter $V[z]$, where $z = (z_0, \ldots, z_{m-1})$. We repeat this process for all N pairs. Finally, we compute the following statistic:

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

For the pairs (x, y) derived from E_{D}, i.e., $y = E_{\mathrm{D}}(x)$, the statistic T follows a χ^2-distribution with mean $\mu_0 = (l-1)\frac{2^n - N}{2^n - 1}$ and variance $\sigma_0^2 = 2(l-1)(\frac{2^n - N}{2^n - 1})^2$ However, it follows a χ^2-distribution with mean $\mu_1 = (l-1)$ and variance $\sigma_1^2 = 2(l-1)$ for a random permutation [6]. 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'.

The above 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, respectively. Then, the number of required pairs N, to successfully distinguish E_{D} from a random permutation is [6]:

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

where $Z_{1-\alpha}$, and $Z_{1-\beta}$ are respective quantiles of the standard normal distribution. As can be seen in Equation 5, the data complexity of the multidimensional ZC attack depends on the number of ZC linear 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. Let denote by $E = E_{\mathrm{F}} \circ E_{\mathrm{D}} \circ E_{\mathrm{B}}$ the extended cipher. Suppose that we are given N plaintext/ciphertext pairs $(p, c = E(p))$. We can divide the key recovery into two steps:

> *Step1: 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 and output of E_{D}, where $x = E_{\mathrm{B}}(p)$, and $y = E_{\mathrm{F}}^{-1}(c)$. Assuming that wrong keys yield pairs (x, y) randomly chosen from $\mathbb{F}_2^{2n}$, and the correct key yields the actual input and output of E_{D}, we use the statistic T to discard all keys for which $T \leq \tau$.
> *Step2: 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, i.e., 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 [14] 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 [6] showed that an integral distinguisher[3] always implies a ZC distinguisher, but its converse is true only if the input and output linear masks of the ZC distinguisher are independent. Later, Sun et al. [34] proposed Theorem 1 showing that the conditions for deriving an integral distinguisher from a ZC linear hull in [6] can be removed.

Theorem 1 (Sun et al. [34]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a vectorial Boolean function. In addition, suppose that 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 linear 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 following 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 tweakey on the ZC distinguishers of tweakable block ciphers. They showed that taking the tweakey

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

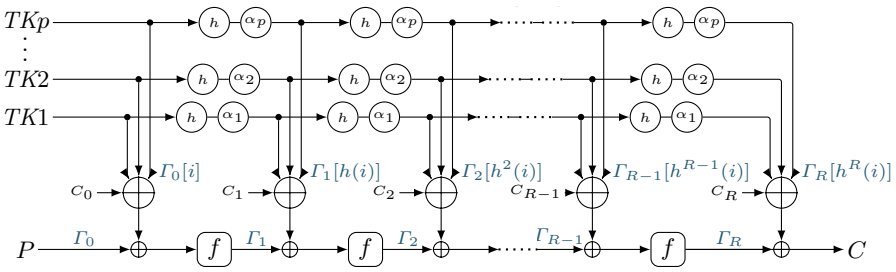

Fig. 2: The STK construction of the TWEAKEY framework.

schedule into account can lead to a longer ZC distinguisher and thus a longer integral distinguisher. Particularly, they proposed Theorem 2 which provides a miss-in-the-middle like algorithm to find ZC linear hulls for tweakable block ciphers following the super-position tweakey (STK) construction of the TWEAKEY framework [20] (see Figure 2).

Theorem 2 (Ankele et al. [1]). *Suppose that $E_K(T,P) : \mathbb{F}_2^{t\times n} \to \mathbb{F}_2^n$, be a tweakable block cipher following the STK construction of the* TWEAKEY *framework. Assume that the tweakey schedule of E_K employs p parallel paths and applies a permutation h on the positions of 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 (see Figure 2). If there is a cell position i such that any possible sequence $\Gamma_0[i], \Gamma_1[h(i)], \Gamma_2[h^2(i)], \ldots \Gamma_r[h^r(i)]$ has at most p linearly active cells, then (Γ_0, Γ_r) yields a ZC linear hull for r rounds of E.*

Ankele et al. used Theorem 2 to manually find ZC linear hulls for several twekable block ciphers including SKINNY, QARMA [2], and MANTIS [3]. Later, Hadipour et al. [19] proposed a bit-wise 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. [28] to revisit the ZC linear hulls of SKINNY-64-128 and SKINNY-64-192.

The following theorem is a natural generalization of Theorem 1 to the case of tweakable block ciphers.

Theorem 3. *Let $E_K(T,P) : \mathbb{F}_2^{t\times n} \to \mathbb{F}_2^n$, be a tweakable block cipher. Moreover, suppose that A is a subspace of $\mathbb{F}_2^{t\times n}$, and $\beta \in \mathbb{F}_2^n \setminus \{0\}$, such that $((\alpha_1,\alpha_2),\beta)$ is a ZC linear approximation for any $(\alpha_1,\alpha_2) \in A$. then, for any $(\lambda_1,\lambda_2) \in \mathbb{F}_2^{t\times n}$, $\langle \beta, E_K(t+\lambda_1, x+\lambda_2)\rangle$ is balanced over the following set:*

$$A^{\perp} = \{(t,x) \in \mathbb{F}_2^{t\times n} \mid \langle(\alpha_1,\alpha_2),(t,x)\rangle = 0 \text{ for all } (\alpha_1,\alpha_2) \in A\}.$$

According to Theorem 3, the data complexity of the resulting integral distinguisher is 2^{n+t-d}, where t, and n are the tweak and block sizes, respectively, and d is the dimension of linear subspace $A \subseteq \mathbb{F}_2^{n+t}$.

2.4 Constraint Satisfaction and Constraint Optimization Problems

Constraint satisfaction problem (CSP) is a mathematical problem including a set of constraints over a set of variables that should be satisfied. CSP is formally defined as follows:

Definition 1. *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* $X_i \in \mathcal{D}_i$ *for all* $0 \leq i \leq n-1$*.*
- $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{n-1}\}$ *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 assignments 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, a function of variables that should be minimized (or maximized). The objective function is a function of variables that should be minimized (or maximized). Searching for the solution of a CSP or COP problem is usually referred to as constraint programming (CP), and the solvers performing the search are called CP solvers.

We use MiniZinc [27] 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 [29], and for CSP/COP problems over real numbers, we employ Gurobi [16] as the solver.

2.5 Specification of the SKINNY family of tweakable block ciphers

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

As illustrated in Figure 3, each round of SKINNY applies five basic operations to the internal state: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR), and MixColumns (MC). The SC operation applies a 4-bit (or an 8-bit) S-box on each cell. AC combines the round constant with the internal state using the bitwise exclusive-or (XOR). In ART layer, the cells in the

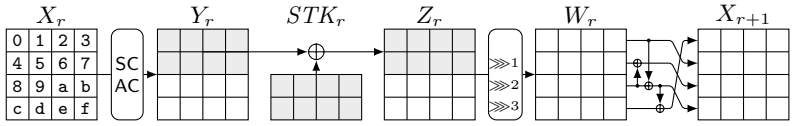

Fig. 3: Round function of SKINNY

first and the second rows of subtweakey are XORed to the corresponding cells in the internals state. SR applies a permutation P on the position of the state cells, where $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. MC multiplies each column of the internal state by a non-MDS matrix M. M and its inverse are as follows:

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

Figure 3 represents the variables we use to denote the internals states of SKINNY after r rounds. We also use ΔX_r (ΓX_r) to represent the difference (resp. linear mask) of state X_r. To denote the ith cell of state X_r we use $X_r[i]$, where $0 \leq i \leq 15$. STK_r indicates the subtweakey in after r rounds, and $ETK_r = \texttt{MC} \circ \texttt{SR}(STK_r)$ which is called the equivalent subtweakey in round r.

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

$$\begin{cases} STK_r = TK1_r & \text{if } t = n \\ STK_r = TK1_r \oplus TK2_r & \text{if } t = 2n \\ STK_r = TK1_r \oplus TK2_r \oplus TK3_r & \text{if } t = 3n, \end{cases} \tag{6}$$

where $TK1_r, TK2_r, TK3_r$, denote the tweakey arrays in round r and are generated as follows. First, a permutation h is applied to each tweakey array, such that $TKm_r[n] \leftarrow TKm_{r-1}[h(n)]$ for all $0 \leq n \leq 15$, and $m \in \{1, 2, 3\}$. Next, an LFSR is applied to each cell of the first and the second rows of $TK2_r$ and $TK3_r$. For more details on the specification of SKINNY we refer the reader to [3].

For any fixed $i \in \{0, \ldots, 15\}$, and $R \in \mathbb{N}$, we define the ith lane of tweakey $TK1$, $TK2$, and $TK3$ as follows:

$$\begin{aligned} Lane_1[i] &= [TK1[h^r(i)] : \; 0 \leq r \leq R-1], \\ Lane_2[i] &= [L_2^r(TK2[h^r(i)]) : \; 0 \leq r \leq R-1], \\ Lane_3[i] &= [L_3^r(TK3[h^r(i)]) : \; 0 \leq r \leq R-1], \end{aligned} \tag{7}$$

where L_2 and L_3 are the linear transformations corresponding to the LFSRs in tweakey paths $TK2$, and $TK3$, respectively.

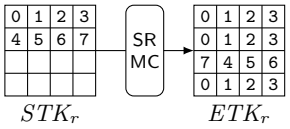

Fig. 4: Relation between the subtweakey and the equivalent subtweakey

2.6 Encoding Deterministic Truncated Trails

Here, we recall the method proposed in [35] 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 $\Delta X[i]$ which is defined as follows:

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

We also use an integer variable $\mathtt{DX}[i]$ to encode the actual c-bit difference value of $\Delta X[i]$, which is defined as follows:

$$\mathtt{DX}[i] \in \begin{cases} \{0\}, & \text{if } \mathtt{AX}[i] = 0 \\ \{1, \ldots, 2^c - 1\}, & \text{if } \mathtt{AX}[i] = 1 \\ \{-1\}, & \text{if } \mathtt{AX}[i] = 2 \\ \{-2\}, & \text{if } \mathtt{AX}[i] = 3 \end{cases}$$

Then, we include the following constraints to link $\mathtt{AX}[i]$ and $\mathtt{DX}[i]$ for all $i = 0, \ldots, m-1$:

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

Note that, MiniZinc supports conditional expression '*if-then-else-endif*', and we have not to convert it to integer inequalities. In what follows, we briefly explain the propagation rules of deterministic truncated differential trails through the building blocks of block ciphers.

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

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

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

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

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

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

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

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

For non-MDS matrices, such as the matrix employed in SKINNY, as illustrated in Section B, 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. The miss-in-the-middle technique was firstly used by Biham et

al. [4,5] to build the ID distinguishers. 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 the halfway deterministic propagations results in an impossible differential (resp. unbiased linear hull).

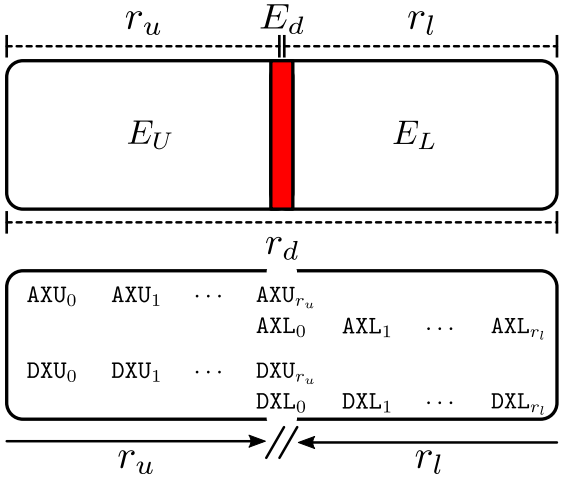

Fig. 5: Modeling the miss-in-the-middle technique as a CSP problem

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 find the ID and ZC distinguishers automatically. To do so, as illustrated in Figure 5 we first divide E_{D} into two parts: E_{U}, and E_{L} covering r_{U}, and r_{L} rounds, respectively. We refer to E_{U} and E_{L} as the upper and lower parts of the distinguisher, respectively. The trails discovered for E_{U} (E_{L}) are referred to as upper (resp. lower) trails, hereafter. We also use XU_r (XL_r) to represent the internals state of E_{U} (E_{L}) after r rounds. The internal state, which is in the intersection of E_{U} and E_{L}, i.e., $XU_{r_{\text{U}}}$ (or XL_0), is called the meeting point.

As Figure 5 shows, let $\texttt{AXU}_r$ and $\texttt{AXL}_r$ indicate the activeness pattern of the state variables XU_r and XL_r, respectively. Moreover, let $\texttt{DXU}_r$ and $\texttt{DXL}_r$ denote the actual difference values of state variables in round r of E_{U} and E_{L}, respectively. We encode the deterministic truncated differential trails propagation through E_{U} and E_{L} in opposite directions as two independent CSP problems. To encode the deterministic truncated trails, we use the rules described in Section 2.6. We also exclude the trivial solutions by adding the constraints $\sum_i^{m-1} \texttt{AXU}_0[i] \neq 0$ and $\sum_i^{m-1} \texttt{AXL}_{r_{\text{L}}} \neq 0$, for the upper and lower trails, respectively. 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 $\mathit{CSP}_{\mathrm{L}}(\mathtt{AXL}_0, \mathtt{DXL}_0, \ldots, \mathtt{AXL}_{r_{\mathrm{L}}}, \mathtt{DXL}_{r_{\mathrm{L}}})$ be the model for propagation of deterministic truncated trails over E_{L}^{-1}. Thus, any feasible solutions of $\mathit{CSP}_{\mathrm{U}}$ ($\mathit{CSP}_{\mathrm{L}}$) corresponds to a deterministic truncated differential trail through E_{U} (resp. E_{L}^{-1}).

Note that, the last internal state in E_{U} and the first internal state of E_{L} overlap at the meeting point, i.e., 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, as follows:

$$\begin{aligned}&\mathit{CSP}_M(\mathtt{AXU}_{r_{\mathrm{L}}}, \mathtt{DXU}_{r_{\mathrm{L}}}, \mathtt{AXL}_0, \mathtt{DXL}_0) \coloneqq \\ &\bigvee_{i=0}^{m-1} \begin{pmatrix} (\mathtt{AXU}_{r_{\mathrm{U}}}[i] + \mathtt{AXL}_0[i] > 0) \wedge \\ (\mathtt{AXU}_{r_{\mathrm{U}}}[i] + \mathtt{AXL}_0[i] < 3) \wedge \\ \mathtt{AXU}_{r_{\mathrm{U}}}[i] \neq \mathtt{AXL}_0[i] \end{pmatrix} \vee \bigvee_{i=0}^{m-1} \begin{pmatrix} \mathtt{AXU}_{r_{\mathrm{U}}}[i] = 1 \wedge \\ \mathtt{AXL}_0[i] = 1 \wedge \\ \mathtt{DXU}_{r_{\mathrm{U}}}[i] \neq \mathtt{DXL}_0[i] \end{pmatrix} = \mathit{True}\end{aligned} \tag{8}$$

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

Although we encode the deterministic truncated trails in the same way as [35], our method to search for distinguishers has some important differences. Sun et al. [35] solves $\mathit{CSP}_{\mathrm{U}}$ and $\mathit{CSP}_{\mathrm{L}}$ separately through a loop where the activeness pattern of a cell at the meeting point is fixed in each iteration. The main advantage of our model is that any solutions of $\mathit{CSP}_{\mathrm{D}}$ corresponds to an ID (or ZC) distinguisher. In addition, we do not constrain the value of our model at the input/output or at meeting point. These key feature enables us to extend our model for the key recovery and build a unified COP for finding the nearly optimum ID and ZC attacks in the next sections.

We showed how to encode and detect the contradiction in the meeting point. However, the contradiction may occur in other positions, such as in the tweakey schedule (see Theorem 2). Our method to model the distinguishers is not limited to the basic miss-in-the-middle technique. One can extend this model to encode the contradiction beyond the meeting point to find longer distinguishers. In what follows, 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.

Suppose that the block cipher E follows the STK construction of the TWEAKEY framework with p parallel independent paths in the tweakey schedule. Moreover, 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, where $i = 0, \ldots, m-1$. For all $i = 0, \ldots, m-1$, we define the integer variable $\mathtt{ASTK}_r[i] \in \{0, 1, 2, 3\}$, to indicate the activeness pattern of $STK_r[i]$. Then we define the following constraints to ensure that there is a contradiction

in the tweakey schedule and the condition of Theorem 2 holds:

$$
\begin{aligned}
&\mathit{CSP}_{TK}(\mathtt{ASTK}_0, \ldots, \mathtt{ASTK}_{r_\mathrm{D}-1}) \coloneqq \\
&\bigvee_{i=0}^{m-1} \left(\left(\begin{array}{l} \displaystyle\sum_{r=0}^{r_\mathrm{D}-1} \mathtt{bool2int}\left(\mathtt{ASTK}_r[h^{(r)}(i)] \neq 0\right) \leq p \\ \wedge \quad \displaystyle\bigvee_{r=0}^{r_\mathrm{D}-1} \left(\mathtt{ASTK}_r[h^{(r)}(i)] = 1\right) \end{array} \right) \vee \left(\bigwedge_{r=0}^{r_\mathrm{D}-1} \mathtt{ASTK}_r[h^{(r)}(i)] = 0 \right) \right) \\
&= \mathit{True}
\end{aligned} \tag{9}
$$

As you can see, Equation 9 guarantees that at least one lane of tweakey schedule has at most p active cells, or it is totally inactive. Finally, we make the CSP problem $\mathit{CSP}_\mathrm{D} \coloneqq \mathit{CSP}_\mathrm{U} \wedge \mathit{CSP}_\mathrm{L} \wedge \mathit{CSP}_{TK}$ to find the ZC distinguishers of tweakable block ciphers taking the tweakey schedule into account.

We assume that the round keys are independent. Thus, our method regards even those differential or linear propagations over multiple rounds that can not occur due to the global dependency between the round keys as possible propagations. We also consider the S-box 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 impossible differentials (ZC linear hulls) discovered by our method are always valid.

4 Modeling the Key Recovery of Impossible Differential Attack

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

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

the complexity analysis of the ID attack as follows:

$$
\begin{aligned}
&T_0 = \max \left\{ \begin{array}{l} \min\limits_{\Delta \in \{\Delta_{\rm B}, \Delta_{\rm F}\}} \{2^{\frac{c_{\rm B}+c_{\rm F}+n+1-|\Delta|+LG(g)}{2}}\}, \\ 2^{c_{\rm B}+c_{\rm F}+n+1-|\Delta_{\rm B}|-|\Delta_{\rm F}|+LG(g)} \end{array} \right\},\ T_0 < 2^n, \\
&T_1 = 2^{c_{\rm B}+c_{\rm F}+LG(g)},\ T_2 = 2^{|k_{\rm B} \cup k_{\rm F}|+LG(g)},\ T_3 = 2^{k-g}, \\
&T_{tot} = (T_0 + (T_1 + T_2)\, C_{E'} + T_3)\, C_E,\ T_{tot} < 2^k, \\
&M_{tot} = \min \left\{ 2^{c_{\rm B}+c_{\rm F}+LG(g)}, 2^{|k_{\rm B} \cup k_{\rm F}|} \right\},\ M_{tot} < 2^k.
\end{aligned}
\tag{10}
$$

When searching for an optimal full ID attack, we aim to minimize the total time complexity while keeping the memory and data complexities under the threshold values. As can be seen in Equation 10, $c_{\rm B}, c_{\rm F}, |\Delta_{\rm B}|, |\Delta_{\rm F}|$, and $|k_{\rm B} \cup k_{\rm F}|$, are the critical parameters which directly affect the final complexity of the ID attack. To determine $(c_{\rm B}, |\Delta_{\rm B}|)$ $((c_{\rm F}, |\Delta_{\rm F}|))$, we need to model the propagation of truncated differential trails through $E_{\rm B}$ (resp. $E_{\rm F}^{-1}$), taking the probability of all differential cancellations into account. To determine $k_{\rm B}$ ($k_{\rm F}$), we need to detect the state cells whose difference or data values are needed through the partial encryption (resp. decryption) over $E_{\rm B}$ (resp. $E_{\rm F}$). Moreover, to determine the actual size of $k_{\rm B} \cup k_{\rm F}$, we should take the (twea)key schedule and key-bridging technique into account. For these purposes, we present a genic COP model including the following main components:

- **Model the distinguisher.** In this component, we model the distinguisher part as we described in Section 3. Unlike the previous methods, our model does not have any constraints for the input/output of the distinguisher.
- **Model the difference propagation in outer parts.** This component models the propagation of truncated differential trails $\Delta_{\rm B} \xleftarrow{E_{\rm B}^{-1}} \Delta_{\rm U}$, and $\Delta_{\rm L} \xrightarrow{E_{\rm F}} \Delta_{\rm F}$ with probability one. Unlike our model for the distinguisher part, where we use integer variables with domain $\{0, \ldots, 3\}$, here, we use binary variables to only encode active/inactive cells through propagation. We also model the number of filters, i.e., $c_{\rm B}$, and $c_{\rm F}$. To do so, we use some additional binary variables and new constraints to encode the probability of $\Delta_{\rm B} \xrightarrow{E_{\rm B}} \Delta_{\rm U}$ and $\Delta_{\rm L} \xleftarrow{E_{\rm F}^{-1}} \Delta_{\rm F}$.
- **Model the guess-and-determine in outer parts.** In this component, we model the determination relationships over $E_{\rm B}$ and $E_{\rm F}$ to detect the state cells whose difference or data values must be known for verifying the differences $\Delta_{\rm U}$, and $\Delta_{\rm L}$. 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 $E_{\rm B}$, and $E_{\rm F}$.
- **Model the key bridging.** In this component, taking the (twea)key schedule into account, we model the relations between the sub-(twea)keys to determine the actual number of involved sub-(twea)keys in the key recovery. For this component, we can use the general CP-based model for key-bridging proposed by Hadipour and Eichlseder in [17], or we can use cipher dedicated models.

– **Model the complexity formulas.** In this component, we model the complexity formulas in Equation 10. To do so, we include the following constraints to the model:

$$\begin{aligned}
&D[0] \coloneqq \mathit{min}_{\Delta \in \{\Delta_{\mathrm{B}}, \Delta_{\mathrm{F}}\}} \{\frac{c_{\mathrm{B}} + c_{\mathrm{F}} + n + 1 - |\Delta| + LG(g)}{2}\},\\
&D[1] \coloneqq c_{\mathrm{B}} + c_{\mathrm{F}} + n + 1 - |\Delta_{\mathrm{B}}| - |\Delta_{\mathrm{F}}| + LG(g),\\
&T[0] \coloneqq \mathit{max}\{D[0], D[1]\},\ \ T[0] < n,\\
&T[1] \coloneqq c_{\mathrm{B}} + c_{\mathrm{F}} + LG(g),\ \ T[2] \coloneqq |k_{\mathrm{B}} \cup k_{\mathrm{F}}| + LG(g),\ \ T[3] \coloneqq k - g,\\
&T \coloneqq \mathit{max}\{T[0], T[1], T[2], T[3]\},\ \ T < k.
\end{aligned} \tag{11}$$

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

Note that all variables in our model are binary or integer variables with a limited domain except for D, and $T[i]$ for $i \in \{0, 1, 2, 3\}$, in Equation 11 which are real numbers. MiniZinc and many MILP solvers such as Gurobi support max, and min operators. We also precompute the values of $LG(g)$ with 3 floating point precision for all $g \in \{2, \ldots, k\}$, and use the $\mathtt{table}$ feature of MiniZinc to model $LG(g)$. As a result, our COP model considers all the critical parameters of the ID attacks. We recall that the only input of our tool is four integer numbers to specify the lengths of $E_{\mathrm{B}}, E_{\mathrm{U}}, E_{\mathrm{L}}$, and E_{F}. So, one can try different lengths for these four parts to find a nearly optimal attack. We can also modify the objective function of our model to minimize the data or memory complexities where time or any other parameter is constrained. Although we discussed our model in the single (twea)key setting, one can extend it for the related (twea)key setting as will show in Section 4.

Another application of our method, which the previous methods can not cover, is checking the validity of the ID attack with predefined attack parameters. For instance, one can remove the objective function and limit the attack parameters, such as the time and data complexities, to a specific bound to check if there is an ID attack with the given parameters. If the resulting CSP model becomes infeasible, the claimed ID attack will likely be invalid.

4.1 Application to SKINNY

In the previous section, we provided a high-level view of our method. Here, we show in more detail how to perform each step. To this end, we build the COP model for finding related-tweakey full ID attacks on SKINNY. We choose the largest variant of SKINNY, i.e., SKINNY-n-$3n$ as our example. Thus, one can model the more minor variants of SKINNY by simplifying this model. In what follows, given four integer numbers $r_{\mathrm{B}}, r_{\mathrm{U}}, r_{\mathrm{L}}, r_{\mathrm{F}}$, we model the full ID attack on $r = r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}} + r_{\mathrm{F}}$ rounds of SKINNY, where $r_{\mathrm{D}} = r_{\mathrm{U}} + r_{\mathrm{L}}$ is the length of the distinguisher and r_{B}, and r_{F} are the lengths of extended parts in backward and forward directions, respectively. Moreover, we assume that the cell size is $c \in \{4, 8\}$.

Model the distinguisher We first model the difference propagation through the tweakey schedule of SKINNY according to algorithm 1. 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 $\texttt{DTK1}[i]$ to encode the c-bit difference in the ith cell of $TK1$. For tweakey paths $TK2$, and $TK3$, we define the integer variables $\texttt{DTK}m_r[i]$ to encode the c-bit difference value in the ith cell of TKm_r, where $0 \leq i \leq 15$, and $m \in \{2, 3\}$. We also define the integer variables $\texttt{ASTK}_r[i]$, and $\texttt{DSTK}_r[i]$ to encode the activeness pattern as well as the c-bit difference value in the ith cell of STK_r. As you can see in algorithm 1, our CSP model for the tweakey schedule of SKINNY is a bit-wise model.

Algorithm 1: CSP Model for the Tweakey Schedule of SKINNY

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

$R \leftarrow r_{\text{B}} + r_{\text{U}} + r_{\text{L}} + r_{\text{F}} - 1$;
Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DTK1}[i] \in \{0, \ldots, 2^c - 1\} : \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DTK2}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DTK3}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 15\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{ASTK}_r[i] \in \{0, 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 7\}$;
$\mathcal{M}.\texttt{var} \leftarrow \{\texttt{DSTK}_r[i] \in \{0, \ldots, 2^c - 1\} : 0 \leq r \leq R, \ 0 \leq i \leq 7\}$;
for $r = 0, \ldots, R; \ i = 0, \ldots, 7$ **do**
    $\mathcal{M}.\texttt{con} \leftarrow \mathit{Link}(\texttt{ASTK}_r[i], \texttt{DSTK}_r[i])$;
for $r = 1, \ldots, R; \ i = 0, \ldots 15$ **do**
    if $i \leq 7$ **then**
        $\mathcal{M}.\texttt{con} \leftarrow \mathit{table}([\texttt{DTK2}_{r-1}[h(i)], \texttt{DTK2}_r[i]], \mathit{lfsr2})$;
        $\mathcal{M}.\texttt{con} \leftarrow \mathit{table}([\texttt{DTK3}_{r-1}[h(i)], \texttt{DTK3}_r[i]], \mathit{lfsr3})$;
    else
        $\mathcal{M}.\texttt{con} \leftarrow \texttt{DTK2}_r[i] = \texttt{DTK2}_{r-1}[h(i)]$;
        $\mathcal{M}.\texttt{con} \leftarrow \texttt{DTK3}_r[i] = \texttt{DTK3}_{r-1}[h(i)]$;
for $r = 0, \ldots, R; \ i = 0, \ldots 7$ **do**
    $\mathcal{M}.\texttt{con} \leftarrow \texttt{DSTK}_r[i] = \texttt{DTK1}[h^r(i)] \oplus \texttt{DTK2}_r[i] \oplus \texttt{DTK3}_r[i]$;
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 algorithm 3, we build the CSP_{U} and CSP_{L}, respectively. We also build the CSP_M according to Equation 8. Lastly, we combine the generated CSP models as follows:

$$\mathit{CSP}_{\text{D}} \coloneqq \mathit{CSP}_{\text{U}} \wedge \mathit{CSP}_{\text{L}} \wedge \mathit{CSP}_M \wedge \mathit{CSP}_{DTK}.$$

Hence, any feasible solution of $\mathit{CSP}_{\mathrm{D}}$ yields a related-tweakey ID distinguisher for SKINNY-n-$3n$. By setting $\mathtt{DTK3}_0$ in algorithm 1 to zero we can find the 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 the single tweaeky ID distinguishers of SKINNY.

Algorithm 2: $\mathit{CSP}_{\mathrm{U}}$ Model for SKINNY

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

1 $r_{\mathrm{off}} \leftarrow r_{\mathrm{B}}$;
2 Declare an empty CSP model $\mathcal{M}$;
3 $\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{DTK}.\mathtt{var}$;
4 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AXU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
5 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DXU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
6 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AYU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
7 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DYU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
8 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AZU}_r[i] \in \{0,1,2,3\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
9 $\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{DZU}_r[i] \in \{-2,\dots,2^c-1\} : \ 0 \le r \le r_{\mathrm{U}}, \ 0 \le i \le 15\}$;
10 $\mathcal{M}.\mathtt{con} \leftarrow \sum_{i=0}^{16} \mathtt{AXU}_0[i] + \sum_{i=0}^{16} \mathtt{DTK1}[i] + \sum_{i=0}^{16} \mathtt{DTK2}_0 + \sum_{i=0}^{16} \mathtt{DTK3}_0[i] \ge 1$;
11 **for** $r = 0,\dots,r_{\mathrm{U}}-1, \ i = 0,\dots,15$ **do**
12 $\quad \mathcal{M}.\mathtt{con} \leftarrow \mathit{Link}(\mathtt{AXU}_r[i],\mathtt{DXU}_r[i]) \wedge \mathit{Link}(\mathtt{AYU}_r[i],\mathtt{DYU}_r[i]) \wedge \mathit{Link}(\mathtt{AZU}_r[i],\mathtt{DZU}_r[i])$;
13 **for** $r = 0,\dots,r_{\mathrm{U}}-1, \ i = 0,\dots,15$ **do**
14 $\quad \mathcal{M}.\mathtt{con} \leftarrow \mathit{S\text{-}box}(\mathtt{AXU}_r[i], \mathtt{AYU}_r[i])$;
15 **for** $r = 0,\dots,r_{\mathrm{U}}-1, \ i = 0,\dots,7$ **do**
16 $\quad \mathcal{M}.\mathtt{con} \leftarrow \mathit{XOR}(\mathtt{AXU}_r[i], \mathtt{DXU}_r[i], \mathtt{ASTK}_{r_{\mathrm{off}}+r}[i], \mathtt{DSTK}_{r_{\mathrm{off}}+r}[i], \mathtt{AZU}_r[i], \mathtt{DZU}_r[i])$;
17 $\quad \mathcal{M}.\mathtt{con} \leftarrow (\mathtt{AZU}_r[i+8] = \mathtt{AYU}_r[i+8]) \wedge (\mathtt{DZU}_r[i+8] = \mathtt{DYU}_r[i+8])$;
18 **for** $r = 0,\dots,r_{\mathrm{U}}-1, \ i = 0,\dots,3$ **do**
19 $\quad I_1 \leftarrow [\mathtt{AZU}_r[P[i]], \mathtt{AZU}_r[P[i+4]], \mathtt{AZU}_r[P[i+8]], \mathtt{AZU}_r[P[i+12]]]$;
20 $\quad I_2 \leftarrow [\mathtt{DZU}_r[P[i]], \mathtt{DZU}_r[P[i+4]], \mathtt{DZU}_r[P[i+8]], \mathtt{DZU}_r[P[i+12]]]$;
21 $\quad O_1 \leftarrow [\mathtt{AXU}_{r+1}[i], \mathtt{AXU}_{r+1}[i+4], \mathtt{AXU}_{r+1}[i+8], \mathtt{AXU}_{r+1}[i+12]]$;
22 $\quad O_2 \leftarrow [\mathtt{DXU}_{r+1}[i], \mathtt{DXU}_{r+1}[i+4], \mathtt{DXU}_{r+1}[i+8], \mathtt{DXU}_{r+1}[i+12]]$;
23 $\quad \mathcal{M}.\mathtt{con} \leftarrow \mathit{Mdiff}(I_1, I_2, O_1, O_2)$;
24 **return** $\mathcal{M}$;

The first operation in the round function of SKINNY is the 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 the AddRoundTweakey or MixColumns. In this case, we should skip the constraints in line 14 of algorithm 2 for the first round, i.e., $r = 0$.

Algorithm 3: CSP_{L} Model for SKINNY

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

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

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 $\Delta_{\text{B}} \xrightarrow{E_{\text{B}}} \Delta_{\text{U}}$, and $\Delta_{\text{L}} \xleftarrow{E_{\text{F}}^{-1}} \Delta_{\text{F}}$, we note that there are two types of probabilistic transitions. The first type is differential cancellation through the XOR operation. The second type is any differential transition ($\texttt{truncated} \xrightarrow{S} \texttt{fixed}$) over the S-boxes. The second type only happens at the boundary of the distinguisher, i.e., the first S-box layer of E_{F}, or the last S-box layer of E_{B}, where a state cell has a fixed difference value at the input/output of distinguisher.

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

through the XOR operation as follows:

$$\mathit{XOR}_1(\mathtt{AX}, \mathtt{AY}, \mathtt{AZ}) \coloneqq (\mathtt{AZ} \geq \mathtt{AX}) \wedge (\mathtt{AZ} \geq \mathtt{AY}) \wedge (\mathtt{AZ} \leq \mathtt{AX} + \mathtt{AY}) \tag{12}$$

We define a binary variable $\mathtt{CB}_r[i]$ ($\mathtt{CF}_r[i]$) 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 differntial cancellation for each XOR operation:

$$\mathit{XOR}_p(\mathtt{AX}, \mathtt{AY}, \mathtt{AZ}, \mathtt{CB}) \coloneqq \mathit{if}\ (\mathtt{AX} + \mathtt{AY} = 2 \wedge \mathtt{AZ} = 0)\ \mathit{then}\ \mathtt{CB} = 1\ \mathit{else}\ \mathtt{CB} = 0 \tag{13}$$

The algorithm 4, and algorithm 5 describes our model for difference propagation over E_{B}, and E_{F}, respectively. We combine $\mathit{CSP}_{\mathrm{B}}^{dp}$ and $\mathit{CSP}_{\mathrm{F}}^{dp}$ into $\mathit{CSP}_{\mathrm{DP}} \coloneqq \mathit{CSP}_{\mathrm{B}}^{dp} \wedge \mathit{CSP}_{\mathrm{F}}^{dp}$ to model the difference propagation through the outer parts.

Algorithm 4: $\mathit{CSP}_{\mathrm{B}}^{dp}$ Model for SKINNY

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

Declare an empty CSP model $\mathcal{M}$;
$\mathcal{M}.\mathtt{var} \leftarrow \mathit{CSP}_{DTK}.\mathtt{var}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AXB}_r[i] \in \{0,1\} :\ 0 \leq r \leq r_{\mathrm{B}},\ 0 \leq i \leq 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{AZB}_r[i] \in \{0,1\} :\ 0 \leq r \leq r_{\mathrm{B}} - 1,\ 0 \leq i \leq 15\}$;
$\mathcal{M}.\mathtt{var} \leftarrow \{\mathtt{CB}_r[i] \in \{0,1\} :\ 0 \leq r \leq r_{\mathrm{B}} - 1,\ 0 \leq i \leq 19\}$;
for $i = 0, \ldots, 15$ **do**
  $\mathcal{M}.\mathtt{con} \leftarrow \mathit{if}\ \mathtt{AXU}_0[i] \geq 1\ \mathit{then}\ \mathtt{AXB}_{r_{\mathrm{B}}}[i] = 1\ \mathit{else}\ \mathtt{AXB}_{r_{\mathrm{B}}}[i] = 0$;
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 $\mathcal{M}$;

Model the guess-and-determine in outer parts In this section we show how to detect the state cells whose difference or data value are needed to verify the filters in differenece propagations $\Delta_{\mathrm{B}} \rightarrow \Delta_{\mathrm{U}}$, and $\Delta_{\mathrm{L}} \leftarrow \Delta_{\mathrm{F}}$.

Algorithm 5: CSP_{F}^{dp} Model for SKINNY

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

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

We first discuss detecting the state cells whose difference values are needed. The difference value in a state cell is needed if the corresponding state cell contributes to a filter, i.e., a differential cancellation. We know that AddRoundTweakey and MixColumns are the only places where a differential cancellation may occur. So, we 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, respectively. We recall that the difference cancellation through each XOR over E_b is already encoded by $\texttt{CB}_r[i]$ variables. As a result, if $\texttt{CB}_r[i] = 1$, then the difference value in the states cells contributing to this diferential cancellation is needed. For instance, if $\texttt{CB}_r[i] = 1$, then $\texttt{KDZB}_r[P[i+4]] = 1$, and $\texttt{KDZB}_r[P[i+4]] = 1$, where $0 \le i \le 3$, and $0 \le r \le r_u - 1$. Besides detecting the new state cells whose difference values are needed in each round, we should encode the propagation of this property from the previous rounds. The lines line 12 to line 15 of algorithm 6 describes how to do this. We also define new constraint according to line 9 of algorithm 6 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, respectively. Then, we follow a similar approach to model the determination of difference values.

When modeling the determination of data values, the SubCells comes into effect. We explain modeling the determination of data values over S-boxes in E_{B}. However, 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 $X_r[i]$ 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 $\mathtt{KXB}_r[i]$ and $\mathtt{KYB}_r[i]$ to indicate whether the values of $X_r[i]$ and $Y_r[i]$ are needed, respectively. Then, we model the determination flow over the S-boxes as follows:

$$\mathit{S\text{-}box}_{gd}(\mathtt{KXB}_r[i], \mathtt{KYB}_r[i], \mathtt{KDXB}_r[i]) \coloneqq \begin{cases} (\mathtt{KYB}_r[i] \geq \mathtt{KXB}_r[i]) \wedge (\mathtt{KYB}_r[i] \geq \mathtt{KDXB}_r[i]) \wedge \\ (\mathtt{KYB}_r[i] \leq \mathtt{KXB}_r[i] + \mathtt{KDXB}_r[i]) \end{cases} \tag{14}$$

We also model the MixColumns according to Equation 18 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 invloved, where $0 \leq r \leq r_{\mathrm{B}} - 1$, and $0 \leq i \leq 15$. One can see that $\mathtt{IKB}_r[i] = 1$ if and only if $i \leq 7$, 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. The algorithm 6 and algorithm 7 describes our CSP models for the guess-and-determine through E_{B}, and E_{F}, respectively. We refer to $\mathit{CSP}_{\mathrm{GD}} \coloneqq \mathit{CSP}^{gd}_{\mathrm{B}} \wedge \mathit{CSP}^{gd}_{\mathrm{F}}$ as our CSP model for the guess-and-determine through the outer parts.

Model the key bridging In the previous section, we modeled the involved subtweakey cells using the variables $\mathtt{IKB}_r[i]$, and $\mathtt{IKF}_r[i]$ for E_{B}, and E_{F}, respectively. Although the subtweakeys used in E_{B} and E_{F} are separated by r_{D} rounds, they may have some relations due to the tweakey schedule. So, 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 introduced in [17] into our model. The model introduced in [17] is generic and can be used for nonlinear (twea)key schedules. 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-$3n$, but it can be adapted for minor variants of SKINNY.

For the ith cell of subtweakey after r rounds, i.e., $STK_r[i]$, we have $STK_r[i] = TK1_r[i] \oplus TK2_r[i] \oplus TK3_r[i]$. Moreover, $TK1_r[i]$, $TK2_r[i]$, and $TK3_r[i]$ are in the ith lane of $TK1$, $TK2$, and $TK3$, respectively. According to Equation 7, knowing $STK_r[h^r(i)]$ in three rounds yields three independent linear 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

Algorithm 6: CSP_{B}^{gd} Model for SKINNY

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

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

more than three different rs. To take this fact into account, we first define new integer variables $\mathtt{IK} \in \{0, \ldots, r_{\mathrm{B}} + r_{\mathrm{F}} - 1\}$, $\mathtt{KE} \in \{0,1,2,3\}$, and $\mathtt{KS} \in \{0, \ldots, 48\}$. Then, assuming that $r_{\mathrm{off}} = r_{\mathrm{B}} + r_{\mathrm{U}} + r_{\mathrm{L}}$, and $z = 3$, we use the following constraints

Algorithm 7: $\mathit{CSP}_{\mathrm{F}}^{gd}$ Model for SKINNY

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

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

to model the key-bridging:

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

We can model the key-bridging for SKINNY-n-$2n$, and SKINNY-n-n by choosing $z = 2$, and $z = 1$ in Equation 15, respectively.

Model the complexity formulas Here, we show how to model the complexity formulas based on the CSP models described in the previous sections. The variable KS in Equation 15 determines the actual number of involved key cells. Assuming that the cells size is c, the actual number of involved key bits is $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = s \cdot \mathtt{KS}$. We can model the other critical parameters of the ID attack as shown in algorithm 8. Lastly, as illustrated in algorithm 8 we cobmine all previous CSP problems into a unified model and define an ojective function to minimize the final time complexity of the ID attack.

Algorithm 8: COP Model for the Full ID Attack on SKINNY

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

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

We applied our method to find the full ID attacks on all variants of SKINNY. Our model includes integer and real variables, so we used Gurobi to solve the

resulting COP problems. Table 1 shows our results. The time, data and memory complexity of our ID attacks are smaller than the best previous ID attack. Particularly, the time complexity of our 19-round single-tweakey ID attack on SKINNY-128-256 is smaller by a factor of $2^{22.57}$ compared to the best previous one [38]. Our tool can produce all the reported results on a laptop in a few seconds. As a result, besides improving the security evaluation against ID attacks, our tool can significantly reduce human effort/error in finding ID attacks.

5 Modeling the Key Recovery of ZC and Integral Attacks

We can extend our models for the ZC and ZC-Integral distinguishers to make a unified model for finding full ZC and ZC-Integral attacks, respectively. One of the critical parameters in the key recovery of the ZC and ZC-Integral attacks is the number of involved key cells in the outer parts. Another parameter that affects the time complexity of these attacks is the hamming weight of internal states in each round. Thus, we should consider these parameters when modeling the key recovery of the ZC and ZC-Integral attacks. Moreover, the meet-in-the-middle and partial sum techniques are essential to reduce the time complexity of ZC-Integral attacks. Therefore, taking these techniques into account, we provide a generic model for key recovery of ZC and ZC-Integral attacks as follows:

- **Model the distinguisher.** We model the distinguisher as described in Section 3.
- **Model the guess-and-determine.** In this module, we model the value paths in the outer part and detect the key cells whose values are needed in key recovery.
- **Model the key bridging.** This module aims at modeling the key bridging in the key recovery.
- **Model the meet-in-the-middle technique.** For the key recovery of integral attack, we model the meet-in-the-middle technique. To this end, we model the path values for each output cell separately, where we define a new integer variable to capture the number of involved key cells in each path. This way, we can detect the most critical path determining the time complexity of the guess-and-filter step. Note that we integrate the meet-in-the-middle technique into our CSP model and do not consider it a post-processing step. As a result, the result derived from our model is optimum in terms of the meet-in-the-middle technique.
- **Model the partial sum technique.** We also consider the partial sum technique for the key recovery of ZC and ZC-Integral attacks.
- **Set the objective function.** The objective of our model is minimizng the final time complexity. To this aim, we try to minimize the number of involved key cells keeping the data and memory complexities under the theresholds.

We applied our unified framework for finding full ZC and integral attacks to all variants of SKINNY, and could significantly improve all previous ZC and integral attacks on this cipher. Notably, we improved the integral attack on

SKINNY-n-3n (SKINNY-n-2n) by 3 (resp. 2) rounds. We also improved the ZC attack on SKINNY-n-n (SKINNY-n-2n) by 2 (resp. 1) rounds. Our COP models for ZC and integral attacks use 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. As you can see in Figure 13, and Figure 14, the input of 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 vector with at most one active cell can not find such a distinguisher.

6 Conclusion

In this paper, we presented a unified CP model to find the full ID, ZC, and integral attacks. 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 an ISO standard block cipher, SKINNY, and improved all of the best previous ID, ZC, and integral attacks on this cipher in the single key setting. We also improved the related-tweakey ID attack on SKINNY-n-3n. 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 integral attacks, more accurately and efficiently.

References

1. Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y.: Zero-correlation attacks on tweakable block ciphers with linear tweakey expansion. IACR Transactions on Symmetric Cryptology **2019**(1), 192–235 (Mar 2019). https://doi.org/10.13154/tosc.v2019.i1.192-235
2. Avanzi, R.: The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/tosc.v2017.i1.4-44
3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016. pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
4. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: EUROCRYPT 1999. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer (1999). https://doi.org/10.1007/3-540-48910-X_2
5. Biham, E., Biryukov, A., Shamir, A.: Miss in the middle attacks on IDEA and khufu. In: FSE 1999. Lecture Notes in Computer Science, vol. 1636, pp. 124–138. Springer (1999)
6. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 244–261. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_16

7. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Des. Codes Cryptogr. **70**(3), 369–383 (2014). https://doi.org/10.1007/s10623-012-9697-z
8. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 29–48. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_3
9. Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. Journal of Cryptology **31**(1), 101–133 (2018). https://doi.org/10.1007/s00145-016-9251-7
10. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to clefia, camellia, lblock and simon. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 179–199. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_10
11. Cui, T., Chen, S., Jia, K., Fu, K., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. Cryptology ePrint Archive, Paper 2016/689 (2016), https://eprint.iacr.org/2016/689, https://eprint.iacr.org/2016/689
12. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer (1997). https://doi.org/10.1007/BFb0052343
13. Derbez, P., Fouque, P.: Automatic search of meet-in-the-middle and impossible differential attacks. In: CRYPTO 2016. Lecture Notes in Computer Science, vol. 9815, pp. 157–184. Springer (2016)
14. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of rijndael. In: FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer (2000). https://doi.org/10.1007/3-540-44706-7_15
15. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of Rijndael. In: FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer (2000). https://doi.org/10.1007/3-540-44706-7_15
16. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022), https://www.gurobi.com
17. Hadipour, H., Eichlseder, M.: Autoguess: A tool for finding guess-and-determine attacks and key bridges. In: ACNS 2022. Lecture Notes in Computer Science, vol. 13269, pp. 230–250. Springer (2022). https://doi.org/10.1007/978-3-031-09234-3_12
18. Hadipour, H., Nageler, M., Eichlseder, M.: Throwing boomerangs into feistel structures: Application to clefia, warp, lblock, lblock-s and twine (2022)
19. 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
20. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: ASIACRYPT 2014. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer (2014). https://doi.org/10.1007/978-3-662-45608-8_15
21. Knudsen, L.: Deal-a 128-bit block cipher. complexity **258**(2), 216 (1998)
22. Lai, X.: Higher order derivatives and differential cryptanalysis pp. 227–233 (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
23. 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

24. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In: INDOCRYPT 2008. Lecture Notes in Computer Science, vol. 5365, pp. 279–293. Springer (2008)
25. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In: CT-RSA 2008. Lecture Notes in Computer Science, vol. 4964, pp. 370–386. Springer (2008)
26. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011). https://doi.org/10.1007/978-3-642-34704-7_5
27. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard CP modelling language. In: CP 2007. Lecture Notes in Computer Science, vol. 4741, pp. 529–543. Springer (2007)
28. Niu, C., Li, M., Sun, S., Wang, M.: Zero-correlation linear cryptanalysis with equal treatment for plaintexts and tweakeys. In: CT-RSA 2021. Lecture Notes in Computer Science, vol. 12704, pp. 126–147. Springer (2021). https://doi.org/10.1007/978-3-030-75539-3_6
29. Perron, L., Furnon, V.: OR-Tools, https://developers.google.com/optimization/
30. 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
31. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017. pp. 185–215. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7
32. Sasaki, Y., Wang, L.: Meet-in-the-middle technique for integral attacks against Feistel ciphers. In: SAC 2012. LNCS, vol. 7707, pp. 234–251. Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_16
33. Shi, D., Sun, S., Derbez, P., Todo, Y., Sun, B., Hu, L.: Programming the demirciselçuk meet-in-the-middle attack with constraints. In: ASIACRYPT 2018. Lecture Notes in Computer Science, vol. 11273, pp. 3–34. Springer (2018). https://doi.org/10.1007/978-3-030-03329-3_1
34. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., AlKhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: CRYPTO 2015. Lecture Notes in Computer Science, vol. 9215, pp. 95–115. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_5
35. Sun, L., Gerault, D., Wang, W., Wang, M.: On the usage of deterministic (related-key) truncated differentials and multidimensional linear approximations for spn ciphers. IACR Transactions on Symmetric Cryptology **2020**(3), 262–287 (Sep 2020). https://doi.org/10.13154/tosc.v2020.i3.262-287
36. Sun, S., Gerault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., Hu, L.: Analysis of aes, skinny, and others with constraint programming. IACR Transactions on Symmetric Cryptology **2017**(1), 281–306 (Mar 2017). https://doi.org/10.13154/tosc.v2017.i1.281-306
37. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24

38. 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

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

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

$$T_0 = \max\left\{ \min_{\Delta \in \{\Delta_{\rm B}, \Delta_{\rm F}\}} \left\{ 2\sqrt{N 2^{n+1-|\Delta|}} \right\}, N 2^{n+1-|\Delta_{\rm B}|-|\Delta_{\rm F}|} \right\}. \tag{16}$$

We can reformulate the complexity analysis to a CSP-friendly form as follows:

$$\begin{aligned}
&T_0 = \max\left\{ \begin{matrix} \min\limits_{\Delta \in \{\Delta_{\rm B}, \Delta_{\rm F}\}} \{2^{\frac{c_{\rm B}+c_{\rm F}+n+1-|\Delta|+\mathit{LG}(g)}{2}+1}\}, \\ 2^{c_{\rm B}+c_{\rm F}+n+1-|\Delta_{\rm B}|-|\Delta_{\rm F}|+\mathit{LG}(g)} \end{matrix} \right\},\ T_0 < 2^n,\\
&T_1 = 2^{c_{\rm B}+c_{\rm F}+\mathit{LG}(g)},\ T_2 = 2^{|k_{\rm B} \cup k_{\rm F}|+\mathit{LG}(g)},\ T_3 = 2^{k-g},\\
&T_{tot} = (T_0 + (T_1 + T_2)\, C_{E'} + T_3)\, C_E,\ T_{tot} < 2^k,\\
&M_{tot} = \min\left\{ 2^{c_{\rm B}+c_{\rm F}+\mathit{LG}(g)}, 2^{|k_{\rm B} \cup k_{\rm F}|} \right\},\ M_{tot} < 2^k,
\end{aligned} \tag{17}$$

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

B Encoding the Matrix of SKINNY

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} \mathit{Mdata}(\mathtt{KX},\mathtt{KY}) &\coloneqq \begin{cases} \mathtt{KY}[0] = \mathtt{KX}[3] \wedge \mathtt{KY}[2] = \mathtt{KX}[1] \wedge \mathit{XOR}_1(\mathtt{KX}[1],\mathtt{KX}[2],\mathtt{D}) \wedge \\ \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[0],\mathtt{KY}[1]) \wedge \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[3],\mathtt{KY}[3]) \end{cases} \\ \mathit{Minvdata}(\mathtt{KX},\mathtt{KY}) &\coloneqq \begin{cases} \mathtt{KY}[1] = \mathtt{KX}[2] \wedge \mathtt{AY}[3] = \mathtt{AX}[0] \wedge \mathit{XOR}_1(\mathtt{KX}[0],\mathtt{KX}[3],\mathtt{D}) \wedge \\ \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[1],\mathtt{KY}[0]) \wedge \mathit{XOR}_1(\mathtt{D},\mathtt{KX}[2],\mathtt{KY}[2]) \end{cases} \end{aligned} \tag{18}$$

C Application to SKINNY

C.1 Impossible Differential Attack on SKINNY

As we have previously stated, the part of time complexity formula in *Guess-and-Filter* step (Equation 2) is only a lower-bound approximation of the time complexity; to determine the complexity precisely, one must perform the detailed attack step by step. As a result, in this paper, we executed the detailed attack step by step to calculate the time complexity. We use Lemma 1 in our attacks.

Lemma 1. *For any given SKINNY S-box, S and any input-output difference $\delta_i, \delta_o \neq 0$, the equation $S(x \oplus \delta_i) \oplus S(x) = \delta_o$ has one solution x on average Similar result holds for the inverse S-box.*

17-round Impossible Differential Attack on SKINNY-n-n In this section, we detail a 17-round attack on SKINNY-n-n that takes the 11-round impossible differential trail shown in Figure 6 and extends it by 3 rounds in both directions. Since there is no tweakey used before W_0', the plaintext P can be recovered by applying $\mathsf{MC}^{-1}, \mathsf{SR}^{-1}, \mathsf{AC}^{-1}$, and SC^{-1} layers on W_0'.

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

Guess-and-Filter. For each of the N pairs

a) *Satisfying round 17.* Calculate $\Delta X_{16}[11, 15]$ using the values of the ciphertext pairs. There is no requirement for any tweaked information to compute these cells here. We get $\Delta X_{16}[3] = \Delta X_{16}[11] = \Delta X_{16}[15]$ because of the MC operation on the active cells in the fourth column of W_{15}. Checking if $\Delta X_{17}[11] = \Delta X_{17}[15]$ will lead to a $c-$bit filter. From the knowledge of $\Delta Y_{16}[3]$ and $\Delta X_{16}[3]$, we can determine $Y_{16}[3]$ by applying Lemma 1. Thus, we can determine $STK_{16}[3]$ (due to $STK_{16}[3] = Z_{16}[3] \oplus Y_{16}[3]$). Now, we can calculate $\Delta X_{16}[13]$ from ciphertext values. Based on the MC operation on the active cells in the second column of W_{15}, we have $\Delta X_{16}[13] = \Delta X_{16}[5] = \Delta X_{16}[1]$. Similarly, the knowledge of this information and Lemma 1 helps us to derive the tweakey cells $STK_{16}[1, 5]$. The time complexity of this step is N, and the number of tests left for the next step is $N.2^{-c}$.

b) *Satisfying round 1.* From the knowledge of $STK_{16}[1, 3, 5]$ (from the previous step), we can uniquely determine $ETK[7, 10, 13]$. Therefore, we determine $\Delta Y_1[7, 10, 13]$. Due to the MC^{-1} operation on the active cells in the first column of X_2, we have $\Delta Y_1[7] = \Delta Y_1[10] = \Delta Y_1[13]$ that will lead to two $c-$bit filters. The time complexity of this step is $N.2^{-c}$, and the number of tests left for the next step is $N.2^{-3c}$.

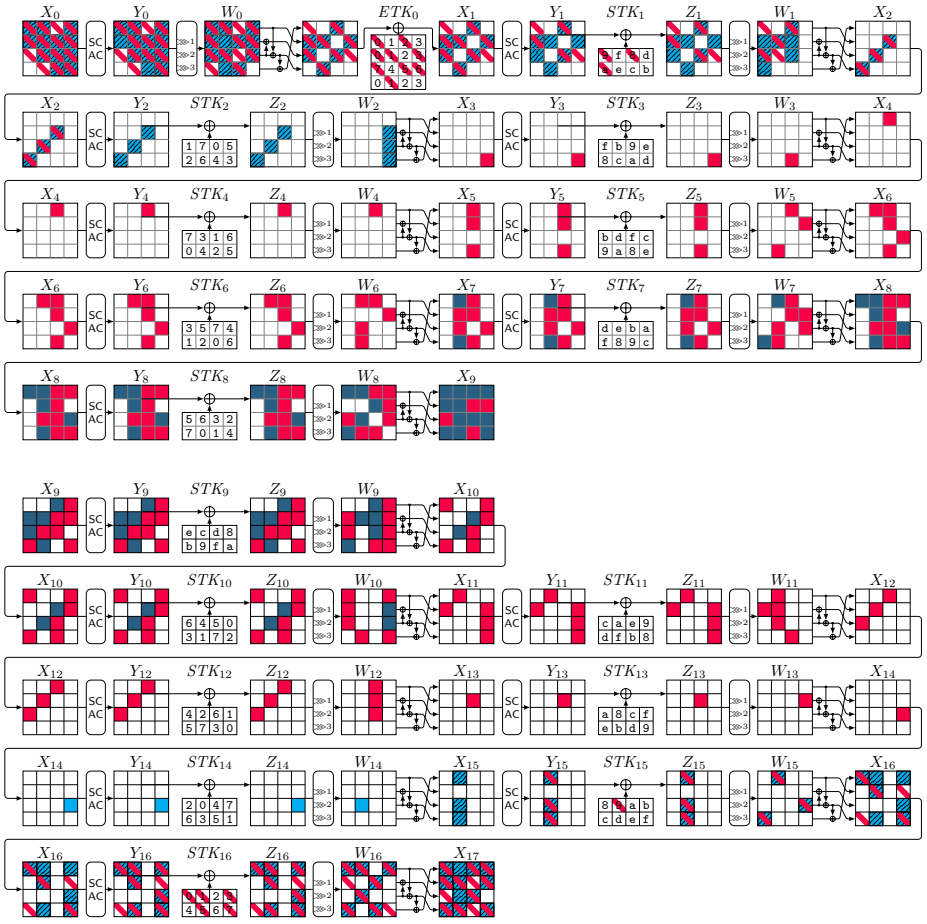

Fig. 6: ID attack on 17 rounds of SKINNY-n-n. $|k_B \cup k_F| = 10$, $c_B = 6$, $c_F = 6$, $\Delta_B = 7$, $\Delta_F = 7$.

c) *Satisfying round 17.* We guess $STK_{16}[0,7]$. Hence, we can compute Z_{15}, and ΔZ_{15} as shown in Figure 6. The time complexity of this step is $N.2^{-c}$, and the number of tests left for the next step is $N.2^{-c}$.

d) *Satisfying rounds 16 and 15.* We can calculate $\Delta X_{15}[9,13]$ using the value of $Z_{15}[9,13]$, and $\Delta Z_{15}[9,13]$. We have $\Delta X_{15}[1] = \Delta X_{15}[9] = \Delta X_{15}[13]$ because of the MC operation on the active cells in the second column of W_{14}. Checking if $\Delta X_{15}[9] = \Delta X_{15}[13]$ will lead to a $c-$bit filter. Given $\Delta X_{15}[1]$, we can determine $Y_{15}[1]$ and so $STK_{15}[1]$ by applying Lemma 1. Hence, we can compute ΔX_{14} as shown in Figure 6. The time complexity of this step is $N.2^{-c}$, and the number of tests left for the next step is $N.2^{-2c}$.

e) *Satisfying round 1.* From the knowledge of $STK_{16}[1,7]$ (from the previous steps), we can uniquely determine $ETK[5,8]$. Therefore, we determine

$\Delta Y_1[5,8]$. Due to the MC^{-1} operation on the active cells in the third column of X_2, we have $\Delta Y_1[2] = \Delta Y_1[5] = \Delta Y_1[8]$. The equality $\Delta Y_1[5] = \Delta Y_1[8]$ will lead to a $c-$bit filter. From the knowledge of $\Delta Y_1[2]$ and $\Delta X_1[2]$, we can determine $X_1[2]$ by applying Lemma 1. Thus, we can derive $ETK[2]$ (duo to $ETK[2] = W_0'[2] \oplus X_1[2]$). We guess $ETK[11]$, and compute Y_1 and ΔY_1 as shown in Figure 6. The time complexity of this step is $N.2^{-c}$, and the number of tests left for the next step is $N.2^{-2c}$.

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

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

19-round Impossible Differential Attack on SKINNY-n-$2n$ In this part, we present the details of our 19-round attack on SKINNY-n-$2n$. We extend the 11-round impossible differential trail illustrated in Figure 7 by 3 and 5 rounds in backward and forward directions, respectively.

Pair Generation. We should build 2^x structures at W_0' and evaluate all possible values in seven cells $W_0'[2,4,5,7,8,10,13]$ for each structure, while the other cells assume a fixed value. By using 2^{x+7c} plaintexts, we can have $2^{x+14c-1}$ pairs of plaintexts $(P, \overline{P})$. The expected number of the remaining pairs of ciphertexts $(C, \overline{C})$ is approximately $N = 2^{x+2|\Delta_B|-1-(n-|\Delta_F|)}$ pairs. In our 19-round attack $n = |\Delta_F|$, and so $N = 2^{x+2|\Delta_B|-1} = 2^{x+14c-1}$. This step needs a total of 2^{x+7c} encryption calls.

Guess-and-Filter. For each of the N pairs

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

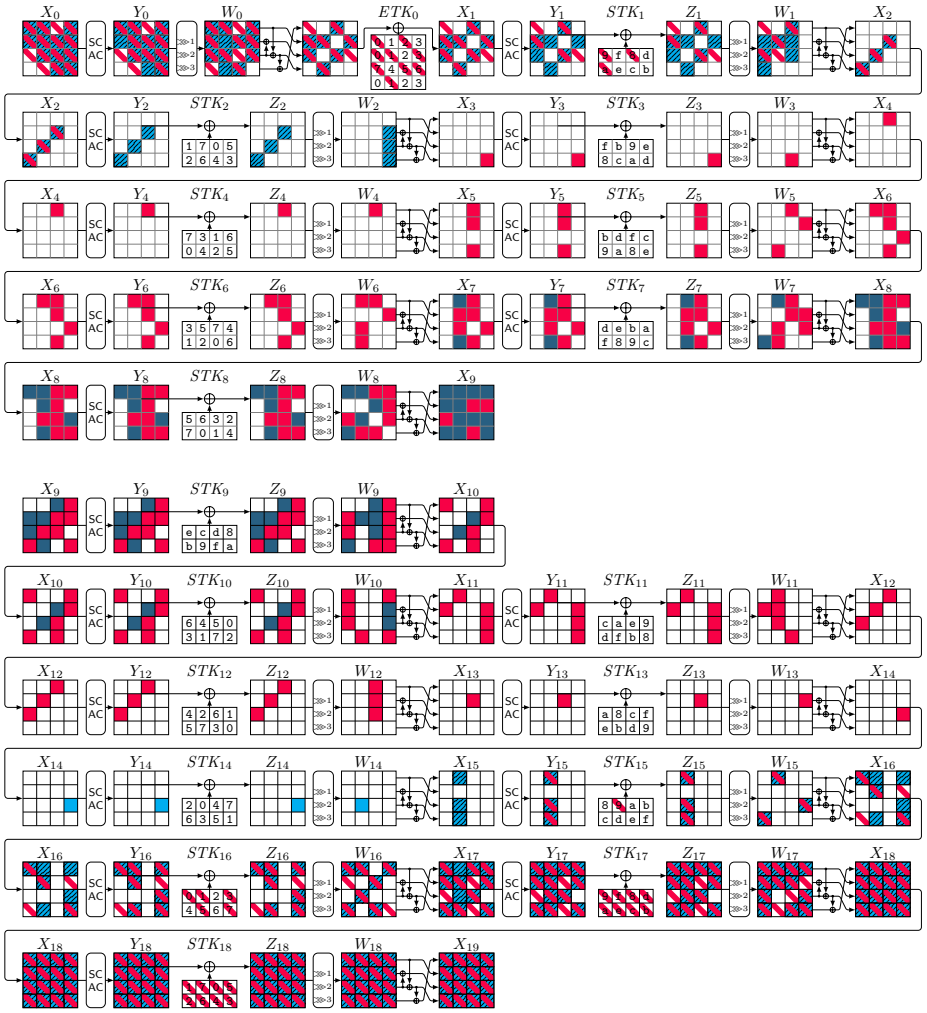

Fig. 7: ID attack on 19 rounds of SKINNY-n-$2n$, $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 26$, $c_{\mathrm{B}} = 6$, $c_{\mathrm{F}} = 15$, $\Delta_{\mathrm{B}} = 7$, $\Delta_{\mathrm{F}} = 16$

column of W_{17}, we can also derive tweakey cells $STK_{18}[6]$ and $STK_{18}[1,5]$, respectively. We guess $STK_{18}[0,2,3,4]$. Hence, we can calculate Z_{17} and ΔZ_{17} as shown in Figure 7. The time complexity of this step is $N.2^{4c}$, and the number of tests left for the next step is $N.2^{4c}$.

b) *Satisfying round 18.* We can calculate $\Delta X_{17}[15]$ using the value of $Z_{17}[15]$, and $\Delta Z_{17}[15]$. We have $\Delta X_{17}[15] = \Delta X_{17}[3] = \Delta X_{17}[7]$ because of the MC operation on the active cells in the fourth column of W_{16}. Given $\Delta X_{17}[3,7]$, we can determine $Y_{17}[3,7]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{17}[3,7]$. Now, we can determine $STK_{17}[3,7]$ (due to $Y_{17}[3] = Z_{17}[3] \oplus$

$STK_{17}[3]$, and $Y_{17}[7] = Z_{17}[7] \oplus STK_{17}[7]$). Similarly, due to the MC operation on the active cells in the second and the first column of W_{16}, we can also derive tweakey cells $STK_{17}[1,5]$, and $STK_{17}[4]$. We guess $STK_{17}[2,6]$. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{6c}$.

c) *Satisfying round 17.* Based on the previous steps, we can compute the cells $Z_{16}[11,15]$, and $\Delta Z_{16}[11,15]$, and thus, $\Delta X_{16}[11,15]$ (there is no requirement for any tweaked information of STK_{16} to compute these cells here). On the other hand, We have $\Delta X_{16}[11] = \Delta X_{16}[15]$ due to the MC operation on the active cells in the fourth column of W_{15}. Checking if $\Delta X_{16}[11] = \Delta X_{16}[15]$ will lead to a $c-$bit filter. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{5c}$.

d) *Satisfying round 18.* We guess $STK_{17}[0]$. Hence, we can calculate Z_{16} and ΔZ_{16} as shown in Figure 7. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{6c}$.

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

f) *Satisfying round 1.* Since the values of $STK_{18}[0,3,7]$, and $STK_{16}[1,3,5]$ are known from the previous steps, we can uniquely determine $ETK[7,10,13]$. Therefore, we can determine $\Delta Y_1[7,10,13]$. Due to the MC^{-1} operation on the active cells in the first column of X_2, we have $\Delta Y_1[7] = \Delta Y_1[10] = \Delta Y_1[13]$ and this will lead to two $c-$bit filters. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{4c}$.

g) *Satisfying round 17.* We guess $STK_{16}[0,7]$. Hence, we can compute Z_{15}, and ΔZ_{15} as shown in Figure 7. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{6c}$.

h) *Satisfying rounds 16 and 15.* We calculate $\Delta X_{15}[9,13]$ using the values of $Z_{15}[9,13]$, and $\Delta Z_{15}[9,13]$. We have $\Delta X_{15}[1] = \Delta X_{15}[9] = \Delta X_{15}[13]$, because of the MC operation on the active cells in the second column of W_{14}. Checking if $\Delta X_{15}[9] = \Delta X_{15}[13]$ will lead to a $c-$bit filter. Given $\Delta X_{15}[1]$, we can determine $Y_{15}[1]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{15}[1]$. Now, we can determine $STK_{15}[1]$ (due to $Y_{15}[1] = Z_{15}[1] \oplus STK_{15}[1]$). Hence, we can compute ΔX_{14} as shown in Figure 7. The time complexity of this step is $N.2^{6c}$, and the number of tests left for the next step is $N.2^{5c}$.

i) *Satisfying round 1.* From the knowledge of $STK_{18}[0,1]$, and $STK_{16}[1,7]$, we can uniquely determine $ETK[5,8]$. Therefore, we determine $\Delta Y_1[5,8]$. Due to the MC^{-1} operation on the active cells in the third column of X_2, we have $\Delta Y_1[2] = \Delta Y_1[5] = \Delta Y_1[8]$. The equality $\Delta Y_1[5] = \Delta Y_1[8]$ will lead to a $c-$bit filter. Since we know the values of $\Delta Y_1[2]$ and $\Delta X_1[2]$, we can

determine $X_1[2]$ by applying Lemma 1. Hence, we derive the value of $ETK[2]$. The time complexity of this step is $N.2^{5c}$, and the number of tests left for the next step is $N.2^{4c}$.

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

k) *Satisfying rounds 2 and 3.* We know the values of $STK_{17}[0]$, and $STK_{15}[1]$ from the previous steps, so we can uniquely determine $STK_1[0]$. We guess $STK_1[4]$. Therefore, we can determine $\Delta Y_2[9, 12]$. Due to the MC^{-1} operation on the active cells in the fourth column of X_3, we have $\Delta Y_2[6] = \Delta Y_2[9] = \Delta Y_2[12]$. The equality $\Delta Y_2[9] = \Delta Y_2[12]$ will lead to a $c-$bit filter. Since we know the values of $\Delta Y_2[6]$ and $\Delta X_2[6]$, we can determine $X_2[6]$ and so $Z_1[2]$ by applying Lemma 1. Therefore, we can determine the value of $STK_1[2]$ as $Y_1[2] \oplus Z_1[2]$. Hence, we can compute ΔX_3 as shown in Figure 7. The time complexity of this step is $N.2^{6c}$, and the number of tests to verify the impossible distinguisher is $N.2^{5c}$.

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

21-round Impossible Differential Attack on SKINNY-n-3n A 11-round distinguisher is placed between Round 6 to Round 16 to attack 21-round of SKINNY-n-3n (see Figure 8). In this attack $|k_B \cup k_F| = 42cc$, $|\Delta_B| = |\Delta_F| = 16c$, $c_B = c_F = 15c$.

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

Guess-and-Filter. For each of the N pairs

a) *Satisfying round 21.* We guess $STK_{20}[0-7]$ and compute W_{19} as shown in Figure 8. Here, we have four $c-$bit filters based on the W_{19} state. The time complexity of this step is $N.2^{8c}$, and the number of tests left for the next step is $N.2^{4c}$.

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

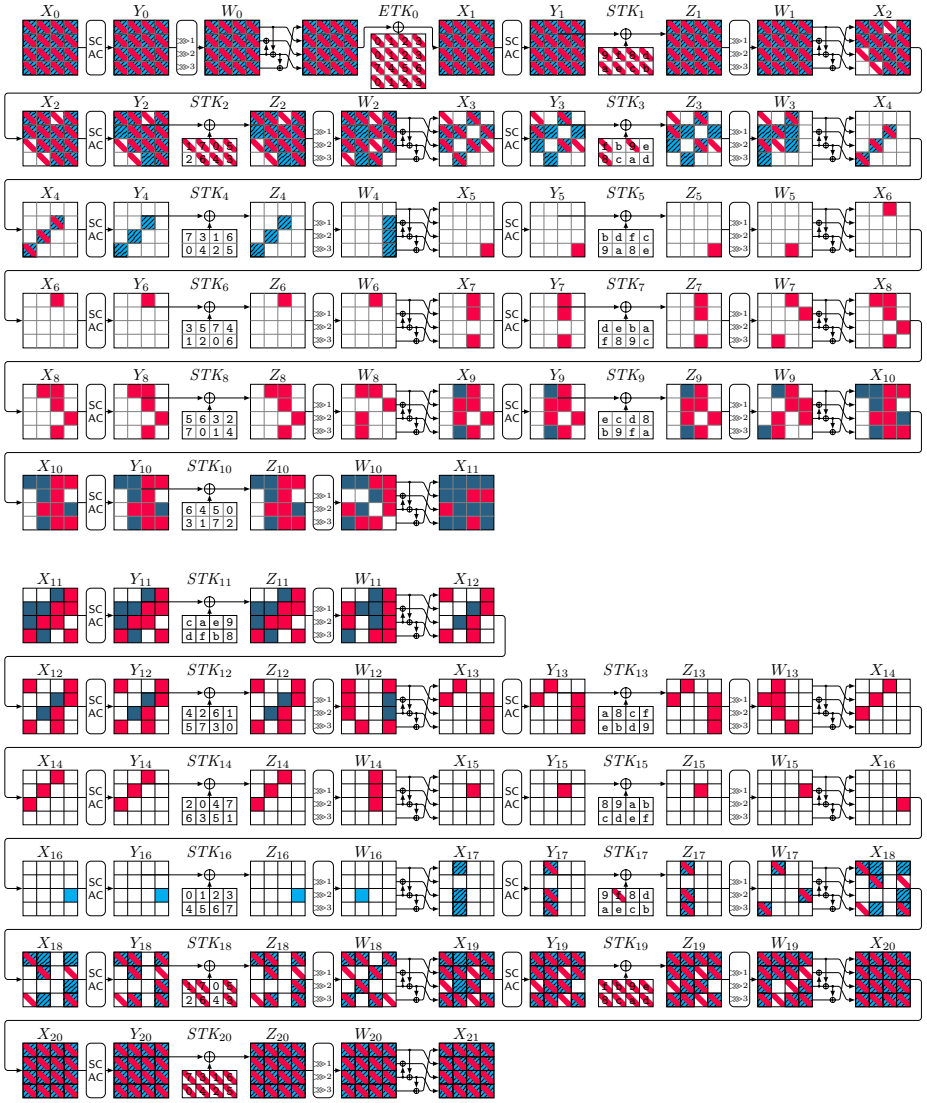

Fig. 8: ID attack on 21 rounds of SKINNY-n-3n. $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 42$, $c_{\mathrm{B}} = 15$, $c_{\mathrm{F}} = 15$, $\Delta_{\mathrm{B}} = 16$, $\Delta_{\mathrm{F}} = 16$.

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

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

e) *Satisfying round 2.* We guess $STK_1[7]$. We can compute Y_2, and ΔY_2 as shown in Figure 8. The time complexity of this step is $N.2^{11c}$, and the number of tests left for the next step is $N.2^{11c}$.

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

g) *Satisfying round 20.* We guess $STK_{19}[0,6]$. Thus, we can compute Z_{18} and ΔZ_{18} as shown in Figure 8. The time complexity of this step is $N.2^{13c}$, and the number of tests left for the next step is $N.2^{13c}$.

h) *Satisfying round 20.* We calculate $\Delta X_{18}[15]$ using the value of $Z_{18}[15]$. We have $\Delta X_{18}[3] = \Delta X_{18}[15]$ because of the MC operation on the active cells in the fourth column of W_{17}. Given $\Delta X_{18}[3]$, we can determine $Y_{18}[3]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{18}[3]$. Now, we can determine $STK_{18}[3]$ (due to $Y_{18}[3] = Z_{18}[3] \oplus STK_{18}[3]$). Similarly, due to the MC operation on the active cells in the second column of W_{17}, we can also derive tweakey cells $STK_{18}[1,5]$. The time complexity of this step is $N.2^{13c}$, and the number of tests left for the next step is $N.2^{13c}$.

i) *Satisfying round 3.* We know the values of $STK_{20}[0,3,7]$, $STK_{18}[1,3,5]$, and $STK_0[5,6,7]$ from the previous steps. Therefore, we will have $STK_2[1,3,5]$. These values help us to compute $W_2[1,3,6,9,10]$ and thus, $Y_3[7,10,13]$, and $\Delta Y_3[7,10,13]$. Due to MC^{-1} operation on the active cells in the first column of X_4, we have $\Delta Y_3[7] = \Delta Y_3[10] = \Delta Y_3[13]$ that will lead to two c-bit filters. The time complexity of this step is $N.2^{13c}$, and the number of tests left for the next step is $N.2^{11c}$.

j) *Satisfying round 19.* We guess $STK_{18}[0,7]$. We compute Z_{17} and ΔZ_{17} as shown in Figure 8. The time complexity of this step is $N.2^{13c}$, and the number of tests left for the next step is $N.2^{13c}$.

k) *Satisfying rounds 18 and 17.* We calculate $\Delta X_{17}[9, 13]$ using the values of $Z_{17}[9, 13]$. We have $\Delta X_{17}[9] = \Delta X_{17}[13] = \Delta X_{17}[1]$ because of the MC operation on the active cells in the second column of W_{16}. The equality $\Delta X_{17}[9] = \Delta X_{17}[13]$ will lead to a c-bit filter. Also, since we know $\Delta X_{17}[1]$, thus, we can determine $Y_{17}[1]$ by applying Lemma 1 and the knowledge of the $\Delta Y_{17}[1]$. Thus, we can determine $STK_{17}[1]$ (due to $Y_{17}[1] = Z_{17}[1] \oplus STK_{17}[1]$). Compute ΔX_{16} as shown in Figure 8. The time complexity of this step is $N.2^{13c}$, and the number of tests left for the next step is $N.2^{12c}$.

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

l) *Satisfying round 3.* We can determine the values of $STK_2[0, 1, 3, 5, 7]$, from the knowledge of $STK_{20}[0-3, 6, 7]$, $STK_{18}[0, 1, 3, 5, 7]$, and $STK_0[1-3, 5-7]$. We also know the value of $STK_2[2]$ from the previous step. We just guess $STK_2[6]$ and compute Y_3, and ΔY_3 as shown in Figure 8. The time complexity of this step is $N.2^{12c}$, and the number of tests left for the next step is $N.2^{12c}$.

m) *Satisfying round 4.* We guess $STK_3[4]$ to determine $X_4[9]$, $\Delta X_4[9]$ and thus, $\Delta Y_4[9]$. Due to MC^{-1} operation on the active cells in the fourth column of X_5, we have $\Delta Y_4[6] = \Delta Y_4[9]$. From the knowledge of $\Delta Y_4[6]$, we can determine $X_4[6]$ by applying Lemma 1 and the knowledge of the $\Delta X_4[6]$. Thus, we can determine the value of $STK_3[2]$ due to $X_4[6] = Y_3[2] \oplus STK_3[2]$. We also can determine the value of $STK_3[0]$, from the knowledge of $STK_{19}[0]$, $STK_{17}[1]$, and $STK_1[1]$. We compute Y_4, and ΔY_4 as shown in Figure 8. Due to MC^{-1} operation on the active cells in the fourth column of X_5, we have $\Delta X_4[9] = \Delta X_4[15]$ that will lead to a c-bit filter. We can compute X_5, and ΔX_5 as shown in Figure 8. The time complexity of this step is $N.2^{13c}$, and the number of tests to verify the impossible distinguisher is $N.2^{12c}$.

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

27-round Related-Tweakey ID Attack on SKINNY-n-3n In this section we provide a 27-round ID attack on SKINNY-n-3n in the related-tweakey setting. Figure 9 illustrates the attack discovered by our tool.

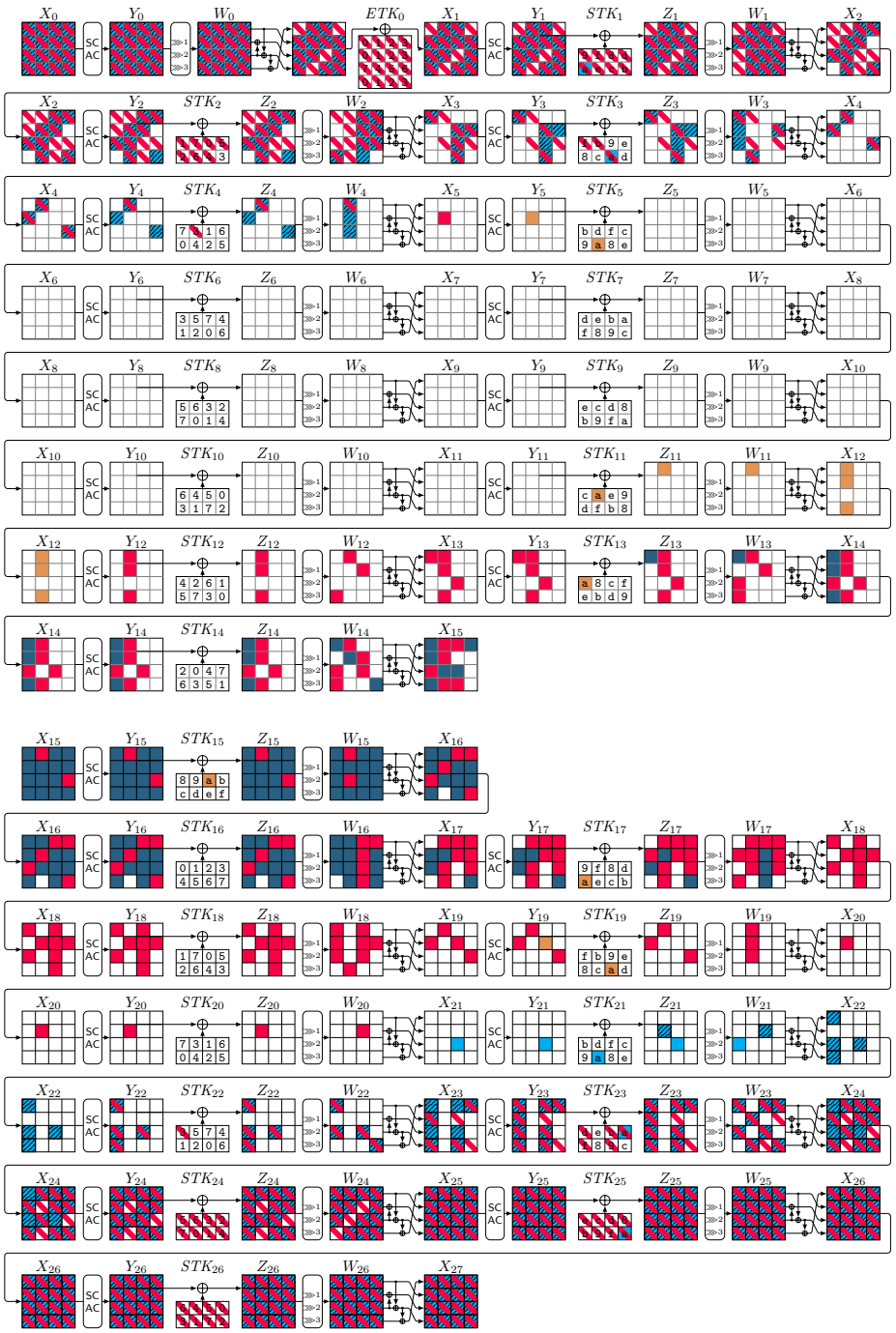

Fig. 9: ID attack on 27 rounds of SKINNY-n-$3n$ in the related-tweakey setting. $|k_{\mathrm{B}} \cup k_{\mathrm{F}}| = 45$, $c_{\mathrm{B}} = 12$, $c_{\mathrm{F}} = 15$, $\Delta_{\mathrm{B}} = 12$, $\Delta_{\mathrm{F}} = 16$.

C.2 Multidimensional Zero-Correlation Linear Cryptanalysis on SKINNY variant

Throughout this section, we apply the multidimensional zero-correlation linear attack to reduced-round versions of SKINNY variant.

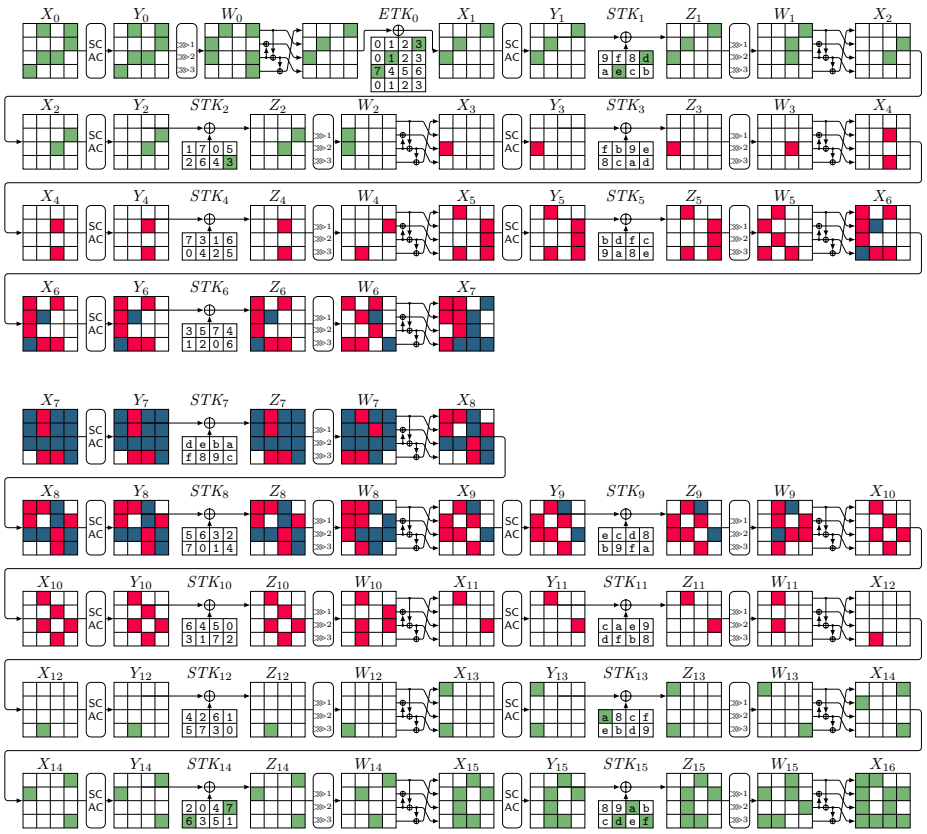

Fig. 10: ZC attack on 16 rounds of SKINNY-n-n. Number of actual involved key cells: 8

Key-Recovery Attack on SKINNY-n-n If the zero-correlation linear approximation over 9-round SKINNY in Figure 10 cover rounds 4 to 12, we can attack 16-round SKINNY-n-n by adding three round before and four rounds after the distinguisher, as shown in Figure 10.

Attack procedure

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

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

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

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

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

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

Key-Recovery Attack on SKINNY-n-2n If the zero-correlation linear approximations over 9-round SKINNY in Figure 11 cover rounds 5 to 13, we can attack 19-round SKINNY-n-2n by adding four rounds before and six rounds after the linear approximations, as shown in Figure 11.

Attack procedure

1. Collect N pairs of plaintexts and the corresponding ciphertexts. Guess 13 cells $STK_{18}[0-7]$, and $STK_{17}[0, 1, 4, 6, 7]$, do the partial decryption and calculate Z_{16} for each pair. Allocate a $4c$-bit counter $N_0[W_0', Z_{16}]$ for all 2^{12c} possible value of $[W_0', Z_{16}]$ and initialize it to zero. Next, compute the number of pairs of plaintext-ciphertext with given values W_0', and Z_{16} and store it in $N_0[W_0', Z_{16}]$. In this step, around 2^{16c} pairs are divided into 2^{12c} distinct values of $[W_0', Z_{16}]$, so $4c$-bit counter is sufficient. The time complexity of this step is equal to $N + N \times 2^{13c}$.
2. Guess 3 cells $STK_{16}[2, 5, 7]$. Next, allocate a counter $N_1[W_0', Z_{15}]$ for all 2^{9c} possible value of $[W_0', Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Z_{16}, do the partial decryption to obtain Z_{15} and update the value $N_1[W_0', Z_{15}] = N_1[W_0', Z_{15}] + N_0[W_0', Z_{16}]$ for all 2^{6c} values of W_0'. The time complexity of this step is equal to $2^{(13c+3c)} \times 2^{6c} \times 2^{6c} = 2^{28c}$ memory access.
3. Guess 2 cells $STK_{15}[3, 4]$. Next, allocate a counter $N_2[W_0', Z_{14}]$ for all 2^{8c} possible value of $[W_0', Z_{14}]$ and initialize it to zero. For all 2^{3c} possible values of Z_{15}, do the partial decryption to obtain Z_{14} and update the value $N_2[W_0', Z_{14}] = N_2[W_0', Z_{14}] + N_1[W_0', Z_{15}]$ for all 2^{6c} values of Y_0. The time complexity of this step is equal to $2^{(16c+2c)} \times 2^{3c} \times 2^{6c} = 2^{27c}$ memory access.
4. We know $STK_{18}[4]$ and $STK_{16}[2]$, thus, we can determine the value of $STK_{14}[0]$. Next, allocate a counter $N_3[W_0', Y_{13}]$ for all 2^{7c} possible value of $[W_0', Y_{13}]$

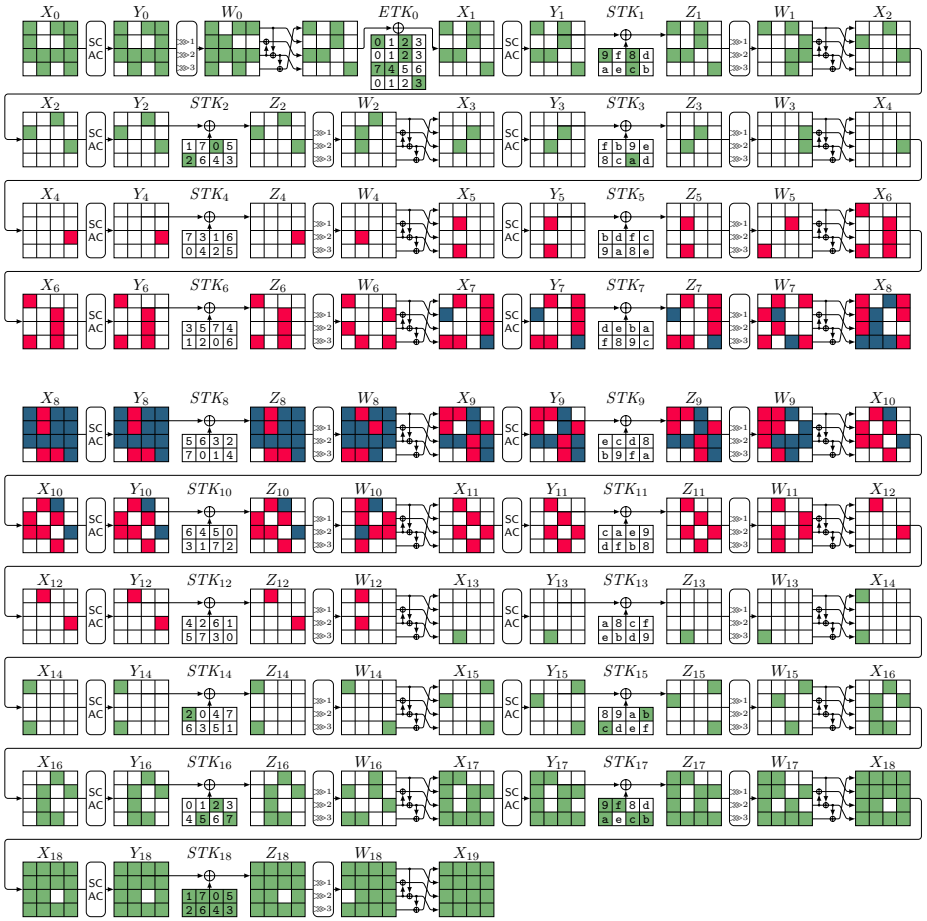

Fig. 11: ZC attack on 19 rounds of SKINNY-n-$2n$. Number of actual involved key cells: 24.

and initialize it to zero. For all 2^{2c} possible values of Z_{14}, do the partial decryption to obtain Y_{13} and update the value $N_3[W_0', Y_{13}] = N_3[W_0', Y_{13}] + N_2[W_0', Z_{14}]$ for all 2^{6c} values of Y_0. The time complexity of this step is equal to $2^{18c} \times 2^{7c} \times 2^{6c} = 2^{31c}$ memory access.

5. From the knowledge of $STK_{18}[4]$ and $STK_{16}[2]$, we can determine the value of $ETK[2, 6]$. Also, from $STK_{18}[1]$ and $STK_{16}[7]$, we determine the value of $ETK[8]$. Thus, we just guess 3 cells $ETK[0, 9, 15]$. Next, allocate a counter $N_4[Y_1, Y_{13}]$ for all 2^{4c} possible value of $[Y_1, Y_{13}]$ and initialize it to zero. For all 2^{6c} possible values of W_0', do the partial decryption to obtain Y_1 and update the value $N_4[Y_1, Y_{13}] = N_4[Y_1, Y_{13}] + N_3[W_0', Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{(18c+3c)} \times 2^{6c} \times 2^c = 2^{28c}$ memory access.

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

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

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

Key-Recovery Attack on SKINNY-n-3n If the zero-correlation linear approximation over 9-round SKINNY-n-3n in Figure 12 cover rounds 5 to 13, we can attack 21-round SKINNY-n-3n by adding four rounds before and eight rounds after the linear approximations, as shown in Figure 12.

Attack procedure

1. Collect N pairs of plaintexts and the corresponding ciphertexts. Guess 29 cells $STK_{20}[0-7], STK_{19}[0-7], STK_{18}[0-7]$, and $STK_{17}[0,1,4,6,7]$, do the

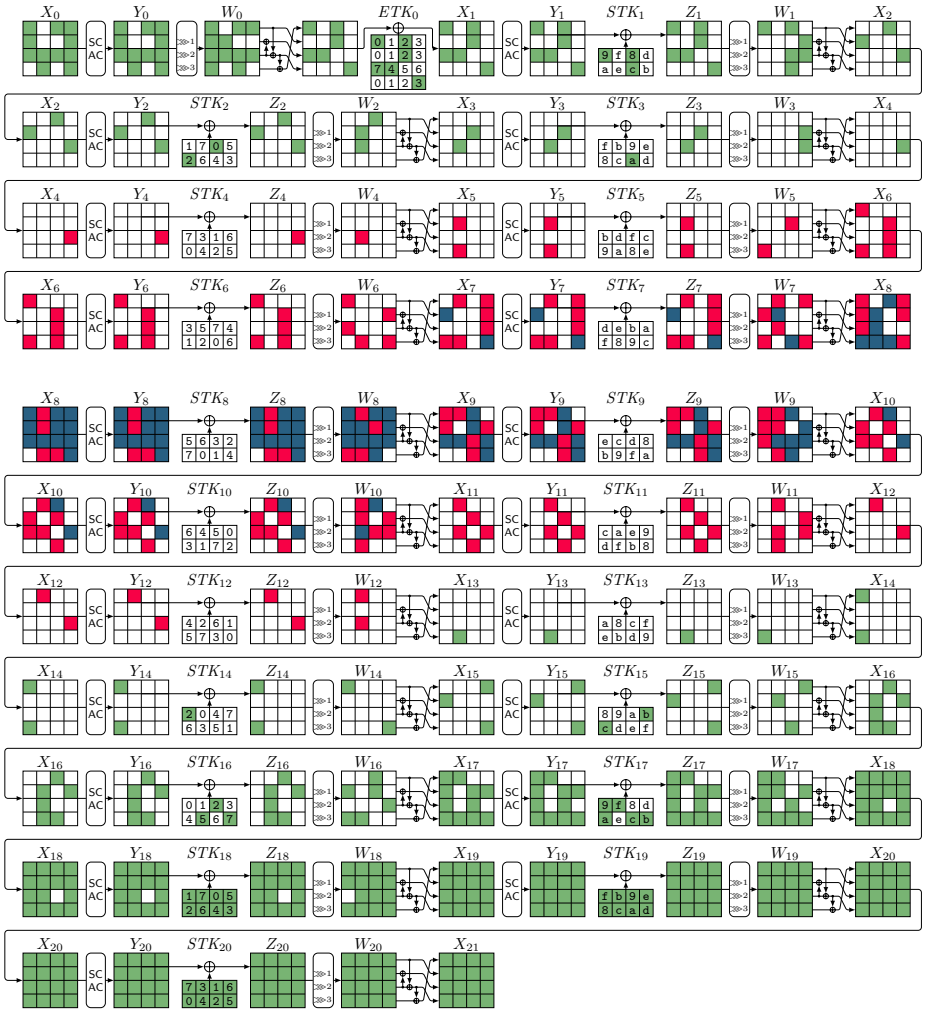

Fig. 12: ZC attack on 21 rounds of SKINNY-n-$3n$. Number of actual involved key cells: 40

partial decryption and calculate Z_{16} for each pair. Allocate a $4c$-bit counter $N_0[W'_0, Z_{16}]$ for all 2^{12c} possible value of $[W'_0, Z_{16}]$ and initialize it to zero. Next, compute the number of pairs of plaintext-ciphertext with given values W'_0, and Z_{16} and store it in $N_0[W'_0, Z_{16}]$. In this step, around 2^{16c} pairs are divided into 2^{12c} distinct values of $[W'_0, Z_{16}]$, so $4c$-bit counter is sufficient. The time complexity of this step is equal to $N + N \times 2^{29c}$.

2. Guess 5 cells $ETK[0, 2, 6, 8, 9, 15]$ ($ETK[6] = ETK[2]$). Next, allocate a counter $N_1[Y_1, Z_{16}]$ for all 2^{12c} possible value of $[Y_1, Z_{16}]$ and initialize it to zero. For all 2^{6c} possible values of Y_0, do the partial encryption to obtain Y_1 and up-

date the value $N_1[Y_1, Z_{16}] = N_1[Y_1, Z_{16}] + N_0[W_0', Z_{16}]$ for all 2^{6c} values of Z_{16}. The time complexity of this step is equal to $2^{(29c+5c)} \times 2^{6c} \times 2^{6c} = 2^{46c}$.

3. From the knowledge of $STK_{20}[0, 6]$, $STK_{18}[1, 4]$, and $ETK[2, 8]$, we can determine the values of $STK_{16}[2, 7]$. Thus, we guess just 1 cell $TK_{16}[5]$. Allocate a counter $N_2[Y_1, Z_{15}]$ for all 2^{9c} possible value of $[Y_1, Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Z_{16}, do the partial decryption to obtain Z_{15} and update the value $N_2[Y_1, Z_{15}] = N_2[Y_1, Z_{15}] + N_1[Y_1, Z_{16}]$ for all 2^{6c} values of Y_1. The time complexity of this step is equal to $2^{(34c+c)} \times 2^{6c} \times 2^{6c} = 2^{47c}$.
4. Guess 3 cells $STK_1[0, 2, 6]$. Allocate a counter $N_3[Y_2, Z_{15}]$ for all 2^{6c} possible value of $[Y_2, Z_{15}]$ and initialize it to zero. For all 2^{6c} possible values of Y_1, do the partial encryption to obtain Y_2 and update the value $N_3[Y_2, Z_{15}] = N_3[Y_2, Z_{15}] + N_2[Y_1, Z_{15}]$ for all 2^{3c} values of Z_{15}. The time complexity of this step is equal to $2^{(35c+3c)} \times 2^{6c} \times 2^{3c} = 2^{47c}$.
5. We can determine the values of $STK_{15}[4]$ from the knowledge of $STK_{19}[5]$, $STK_{17}[6]$,and $STK_1[6]$. Therefore, guess just 1 cell $STK_{15}[3]$. Allocate a counter $N_4[Y_2, Z_{14}]$ for all 2^{5c} possible value of $[Y_2, Z_{14}]$ and initialize it to zero. For all 2^{3c} possible values of Z_{15}, do the partial decryption to obtain Z_{14} and update the value $N_4[Y_2, Z_{14}] = N_4[Y_2, Z_{14}] + N_3[Y_2, Z_{15}]$ for all 2^{3c} values of Y_2. The time complexity of this step is equal to $2^{(38c+c)} \times 2^{3c} \times 2^{3c} = 2^{45c}$.
6. We determine the values of $TK_{14}[0]$ from the knowledge of $STK_{20}[6]$, $STK_{18}[4]$, and $STK_{16}[2]$. Then, allocate a counter $N_5[Y_2, Y_{13}]$ for all 2^{4c} possible values of $[Y_2, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Z_{14}, do the partial decryption to obtain Y_{13} and update the value $N_5[Y_2, Y_{13}] = N_5[Y_2, Y_{13}] + N_4[Y_2, Z_{15}]$ for all 2^{3c} values of Y_2. The time complexity of this step is equal to $2^{39c} \times 2^{2c} \times 2^{3c} = 2^{44c}$.
7. From the knowledge of $STK_{20}[4, 6]$, $STK_{18}[2, 4]$,and $ETK[0, 2]$, we determine $STK_2[2, 4]$. Then, allocate a counter $N_6[Y_3, Y_{13}]$ for all 2^{3c} possible values of $[Y_3, Y_{13}]$ and initialize it to zero. For all 2^{3c} possible values of Y_2, do the partial encryption to obtain Y_3 and update the value $N_6[Y_3, Y_{13}] = N_6[Y_3, Y_{13}] + N_5[Y_2, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{39c} \times 2^{3c} \times 2^c = 2^{43c}$.
8. Guess 1 cell $STK_3[6]$, and then, allocate a counter $N_7[X_4, Y_{13}]$ for all 2^{2c} possible values of $[X_4, Y_{13}]$ and initialize it to zero. For all 2^{2c} possible values of Y_3, do the partial encryption to obtain X_4 and update the value $N_7[X_4, Y_{13}] = N_7[X_4, Y_{13}] + N_6[Y_3, Y_{13}]$ for all 2^c values of Y_{13}. The time complexity of this step is equal to $2^{(39c+c)} \times 2^{2c} \times 2^c = 2^{43c}$.
9. To recover the secret key, allocate a counter $V[z]$ for $2c$-bit z. For 2^{2c} values of $[X_4, Y_{13}]$, evaluate all $2c$ basis zero-correlation masks on $[X_4, Y_{13}]$ and get z. Update the counter $V[z]$ by $V[z] = V[z] + N_7[X_4, Y_{13}]$. Calculate the statistical value T. If $T < \tau$, the guessed key values are possible right key candidates. The time complexity of this step is equal to $2^{40c} \times 4c \times 2^{2c}$ times of reading the $4c$-bit memory.
1. Do an exhaustive search for all the right candidates. The time complexity of this step is equal to $\beta \times 2^{48c}$.

Attack complexity For $c = 4$, we set $\alpha = 2^{-2.7}$ and $\beta = 2^{-7}$, then $Z_{1-\alpha} = 1.01$, and $Z_{1-\beta} = 2.41$. Thus, based on the Equation 5; $N = 2^{62.63}$. The

decision threshold is $\tau = \mu_0 + \sigma_0 Z_{1-\alpha}$. If we consider one memory accesses as a one round, then the time complexity of our attack on 21-round SKINNY-64-192 is about $2^{62.63} + 2^{178.63} \times \frac{4}{21} + (2^{184} + 2^{188} + \cdots + 2^{172} + 2^{172}) \times \frac{1}{21} + 2^{185} = 2^{185.83}$ 21-round encryptions. The required memory complexity is dominated by step 1, which needs about 2^{49} bytes.

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

C.3 Integral Attacks based on ZC Distinguishers for SKINNY

In this section, we transform zero-correlation linear hulls into integral distinguishers to obtain integral attacks for SKINNY in the related-tweakey setting.

ZC-Integral Key-Recovery Attack on 22-Round SKINNY-n-2n Our tool finds a zero-correlation distinguisher for 14 rounds (labelled as rounds 1 to 14 in Figure 13), combined with one free initial round (round 0) and a final key recovery phase over 7 rounds (15 to 21). In this distinguisher, the tweakey cell 8 is only active in at most $p = 2$ cells ('any' in STK_7, 'active' in STK_9). At the input to the distinguisher, 4 cells are active. Thus, we can convert it to an integral distinguisher [1] with data complexity $2^{4\cdot(16-4+2)} = 2^{56}$, where the values in the active input cells and the 2 tweakey cells with index 8 iterate over all values. The inactive input cells are constant, the other tweakey cells form the $4 \cdot 2 \cdot 15 = 120$-bit key. Then, the distinguisher's outputs in $W_{14}[14]$ sum to zero. We can trivially prepend 1 round because the addition of the equivalent tweakey does not change the input structure (key cell 8 is not involved), and all other operations in the first round are unkeyed.

Key recovery For the key recovery, we separately recover the sums in $X_{15}[2]$ and $X_{15}[14]$ using the partial-sum technique [15] and merge the results following the meet-in-the-middle approach [32]. The procedures for both sums are summarized in Table 2. For each sum, we start with Step 0 by storing the obtained ciphertexts (after unwrapping the last linear layer, i.e., Z_{21}) together with their corresponding chosen tweakey values. For the tweakey, we either store the required subtweakey values (i.e., $STK_{21}[6]$) or, if the index is involved more than $p = 2$ times, the input tweakey values from which all subtweakeys can be reconstructed. In each of the following steps in round r, we guess one column of involved subtweakey STK_r and replace the stored values of this column in Z_r by those (potentially fewer) in W_{r-1}. If a subtweakey index is involved more than $p = 2$ times, we only guess the first p times and derive the remaining values afterwards. In each round, we reorder the column order if necessary to minimize the complexity of this round; that is, we first handle columns with fewer key guesses and a stronger reduction in the MixColumns step.

Complexity The complexity of each step is determined by the number of guessed key cells so far and the number of new stored cells (for memory) or previously stored cells (for time). We use the number of S-box lookups as unit for the time complexity, as customary in previous attacks (although in reality, the memory accesses would likely be more expensive). Overall, we obtain a mapping from values of the sum in $X_{15}[2]$ to corresponding 2^{92} key candidates with complexity $2^{106.7}$, and for the sum in $X_{15}[14]$ for 2^{96} candidates with complexity $2^{102.8}$. These can be merged to obtain $2^{120-4} = 2^{116}$ key candidates that produce zero-sums. This remaining keyspace can either be brute-forced (complexity 2^{116}), or the attack can be repeated 3 times (complexity $3 \cdot 2^{106.7} = 2^{108.3}$ plus merging plus $2^{120-3\cdot 4} = 2^{108}$). As merging can be done efficiently, the total complexity is less than 2^{110} encryptions equivalents for 22-round SKINNY-64-128 with 120-bit keys. The same approach yields a complexity of $3 \cdot 2^{216-5.3} + 2^{240-3\cdot 8} = 2^{216}$ for 22-round SKINNY-128-256 with 240-bit keys.

ZC-Integral Key-Recovery Attack on 26-Round SKINNY-n-3n We follow the same approach as above with 4 active input cells, where tweakey cell index e is only active $p = 3$ times ('any' in STK_7, STK_9, 'active' in STK_{11}) in the 16 distinguisher rounds labelled 1 to 16 plus the free initial round 0. We append 9 rounds for key recovery for a total of 26 rounds. The distinguisher and key recovery are illustrated in Figure 14, with key-recovery details given in Table 3.

Complexity The combined complexity of recovering $X_{17}[1]$ and $X_{17}[13]$ from $2^{4\cdot(16-4+3)} = 2^{60}$ data is $2^{172-4.8} + 2^{172-4.9} = 2^{168.2}$. We repeat the attack 2 times and then brute-force the remaining keyspace of $2^{180-2\cdot 4} = 2^{172}$ candidates, which dominates the complexity for 26-round SKINNY-64-192 with 180-bit key. For 26-round SKINNY-128-384 with 360-bit key, the complexity is $2^{2\cdot 172-4.8} + 2^{2\cdot 172-4.9} = 2^{340.2}$ per repetition; with 2 repetitions, the brute-force complexity of $2^{360-2\cdot 8} = 2^{344}$ encryptions dominates.

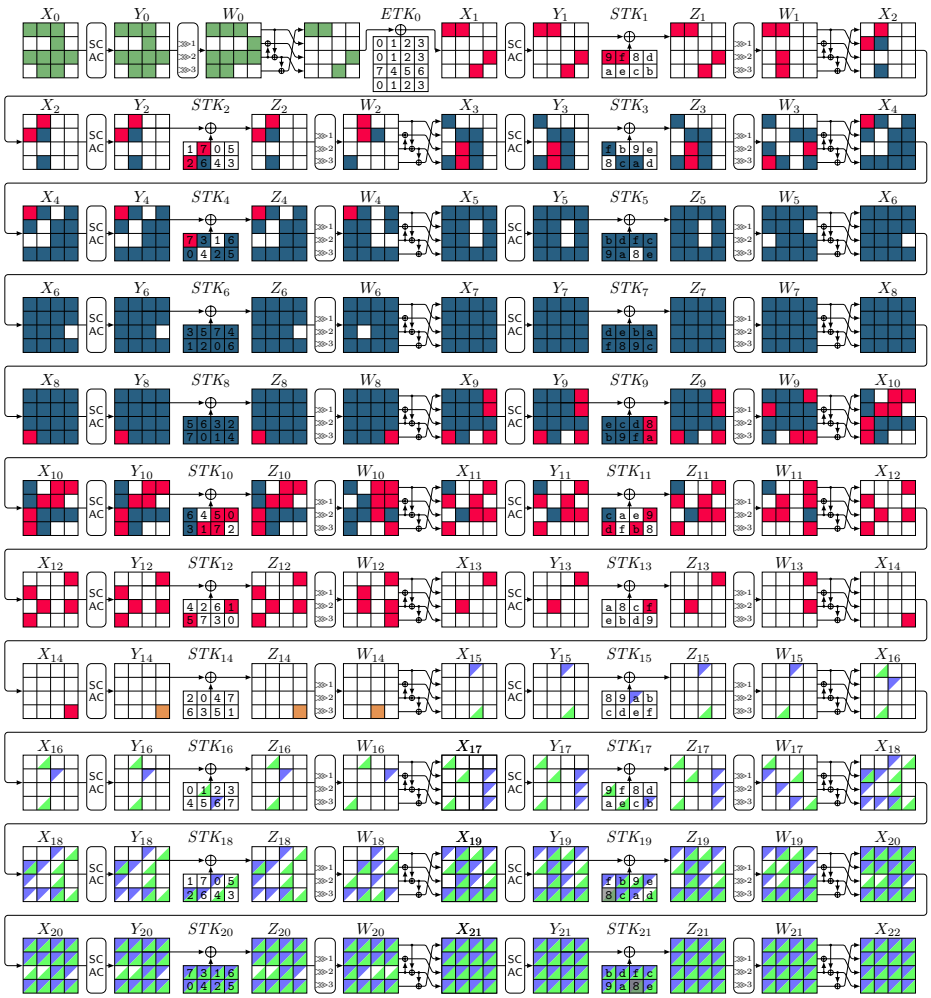

Fig. 13: ZC-based integral attack on 22 rounds of SKINNY-n-2n.

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

(a) Recovery of $X_{15}[2]$ (in Figure 13) with total complexity $2^{112-5.3} = 2^{106.7}$.

Step	Guessed	Keys × Data = Mem	Time · Unit	Stored Texts
0	–	$2^{0} \times 2^{56} = 2^{56}$	$2^{56} \cdot 2^{0.0}$	$Z_{21}[0\text{–}15]$; $STK_{21}[6]$
1–4	$STK_{21}[0\text{–}5, 7]$	$2^{28} \times 2^{56} = 2^{84}$	$2^{84} \cdot 2^{-6.5}$	$W_{20}[0\text{–}7, 9, 12\text{–}15]$
5	$STK_{20}[0, 4]$	$2^{36} \times 2^{56} = 2^{92}$	$2^{92} \cdot 2^{-6.9}$	$Z_{20}[1\text{–}3, 5\text{–}7, 10, 11, 13\text{–}15]$; $W_{19}[0, 8, 12]$
6	$STK_{20}[1, 5]$	$2^{44} \times 2^{52} = 2^{96}$	$2^{100} \cdot 2^{-6.9}$	$Z_{20}[2, 3, 6, 7, 10, 11, 14, 15]$; $W_{19}[0, 8, 12, 1, 13]$
7	$STK_{20}[2, 6]$	$2^{52} \times 2^{44} = 2^{96}$	$2^{104} \cdot 2^{-6.5}$	$Z_{20}[3, 7, 11, 15]$; $W_{19}[0, 8, 12, 1, 13, 6, 14]$
8	$STK_{20}[3, 7]$	$2^{60} \times 2^{44} = 2^{104}$	$2^{104} \cdot 2^{-6.5}$	$W_{19}[0, 8, 12, 1, 13, 6, 14, 3, 7, 11, 15]$
9	$STK_{19}[0]$	$2^{64} \times 2^{40} = 2^{104}$	$2^{108} \cdot 2^{-7.5}$	$Z_{19}[1, 3, 5, 6, 9, 10, 13, 14, 15]$; $W_{18}[12]$
10	$STK_{19}[1, 5]$	$2^{72} \times 2^{32} = 2^{104}$	$2^{112} \cdot 2^{-6.5}$	$Z_{19}[3, 6, 10, 14, 15]$; $W_{18}[12, 5, 13]$
11	$STK_{19}[6]$	$2^{76} \times 2^{32} = 2^{108}$	$2^{108} \cdot 2^{-6.9}$	$Z_{19}[3, 15]$; $W_{18}[12, 5, 13, 2, 6, 10]$
12	$STK_{19}[3]$	$2^{80} \times 2^{28} = 2^{108}$	$2^{112} \cdot 2^{-7.5}$	$W_{18}[12, 5, 13, 2, 6, 10, 15]$
13	$STK_{18}[4]$	$2^{84} \times 2^{20} = 2^{104}$	$2^{112} \cdot 2^{-6.9}$	$Z_{18}[2, 5, 13, 14]$; $W_{17}[4]$
14	$STK_{18}[5]$	$2^{88} \times 2^{16} = 2^{104}$	$2^{108} \cdot 2^{-7.5}$	$Z_{18}[2, 14]$; $W_{17}[4, 9]$
15	$STK_{18}[2]$	$2^{92} \times 2^{12} = 2^{104}$	$2^{108} \cdot 2^{-7.5}$	$W_{17}[4, 9, 14]$
16	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{104} \cdot 2^{-6.9}$	$W_{16}[7]$
17	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{96} \cdot 2^{-8.5}$	$W_{15}[2]$
18	–	$2^{92} \times 2^{4} = 2^{96}$	$2^{96} \cdot 2^{-8.5}$	$X_{14}[2]$

(b) Recovery of $X_{15}[14]$ (in Figure 13) with total complexity $2^{108-5.2} = 2^{-102.8}$.

Step	Guessed	Keys × Data = Mem	Time · Unit	Stored Texts
0	–	$2^{0} \times 2^{56} = 2^{56}$	$2^{56} \cdot 2^{0.0}$	$Z_{21}[0\text{–}15]$; $STK_{19}[4]$, $STK_{21}[6]$
1–4	$STK_{21}[0\text{–}5, 7]$	$2^{28} \times 2^{56} = 2^{84}$	$2^{84} \cdot 2^{-6.5}$	$STK_{19}[4]$; $W_{20}[0\text{–}8, 10\text{–}15]$
5	$STK_{20}[0, 4]$	$2^{36} \times 2^{56} = 2^{92}$	$2^{92} \cdot 2^{-6.5}$	$Z_{20}[1\text{–}3, 5\text{–}7, 9, 10, 13\text{–}15]$; $STK_{19}[4]$; $W_{19}[4, 12]$
6	$STK_{20}[2, 6]$	$2^{44} \times 2^{52} = 2^{96}$	$2^{100} \cdot 2^{-6.5}$	$Z_{20}[1, 3, 5, 7, 9, 13, 15]$; $STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14]$
7	$STK_{20}[3, 7]$	$2^{52} \times 2^{48} = 2^{100}$	$2^{104} \cdot 2^{-6.9}$	$Z_{20}[1, 5, 9, 13]$; $STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14, 11, 15]$
8	$STK_{20}[1, 5]$	$2^{60} \times 2^{48} = 2^{108}$	$2^{108} \cdot 2^{-6.5}$	$STK_{19}[4]$; $W_{19}[4, 12, 2, 6, 14, 11, 15, 1, 5, 9, 13]$
9	–	$2^{60} \times 2^{40} = 2^{100}$	$2^{108} \cdot 2^{-7.5}$	$Z_{19}[1, 2, 5, 7, 9, 11, 13, 14, 15]$; $W_{18}[8]$
10	$STK_{19}[1, 5]$	$2^{68} \times 2^{32} = 2^{100}$	$2^{108} \cdot 2^{-6.5}$	$Z_{19}[2, 7, 11, 14, 15]$; $W_{18}[8, 5, 13]$
11	$STK_{19}[2]$	$2^{72} \times 2^{28} = 2^{100}$	$2^{104} \cdot 2^{-7.5}$	$Z_{19}[7, 11, 15]$; $W_{18}[8, 5, 13, 14]$
12	$STK_{19}[7]$	$2^{76} \times 2^{24} = 2^{100}$	$2^{104} \cdot 2^{-6.9}$	$W_{18}[8, 5, 13, 14, 3, 7]$
13	$STK_{18}[6]$	$2^{80} \times 2^{16} = 2^{96}$	$2^{104} \cdot 2^{-6.9}$	$Z_{18}[3, 4, 15]$; $W_{17}[6]$
14	$STK_{18}[4]$	$2^{84} \times 2^{16} = 2^{100}$	$2^{100} \cdot 2^{-8.5}$	$Z_{18}[3, 15]$; $W_{17}[6, 0]$
15	$STK_{18}[3]$	$2^{88} \times 2^{12} = 2^{100}$	$2^{104} \cdot 2^{-7.5}$	$W_{17}[6, 0, 15]$
16	–	$2^{88} \times 2^{8} = 2^{96}$	$2^{100} \cdot 2^{-7.5}$	$Z_{17}[5]$; $W_{16}[12]$
17	$STK_{17}[5]$	$2^{92} \times 2^{8} = 2^{100}$	$2^{100} \cdot 2^{-8.5}$	$W_{16}[12, 1]$
18	$STK_{16}[1]$	$2^{96} \times 2^{4} = 2^{100}$	$2^{104} \cdot 2^{-7.5}$	$W_{15}[13]$
19	–	$2^{96} \times 2^{4} = 2^{100}$	$2^{100} \cdot 2^{-8.5}$	$X_{14}[14]$

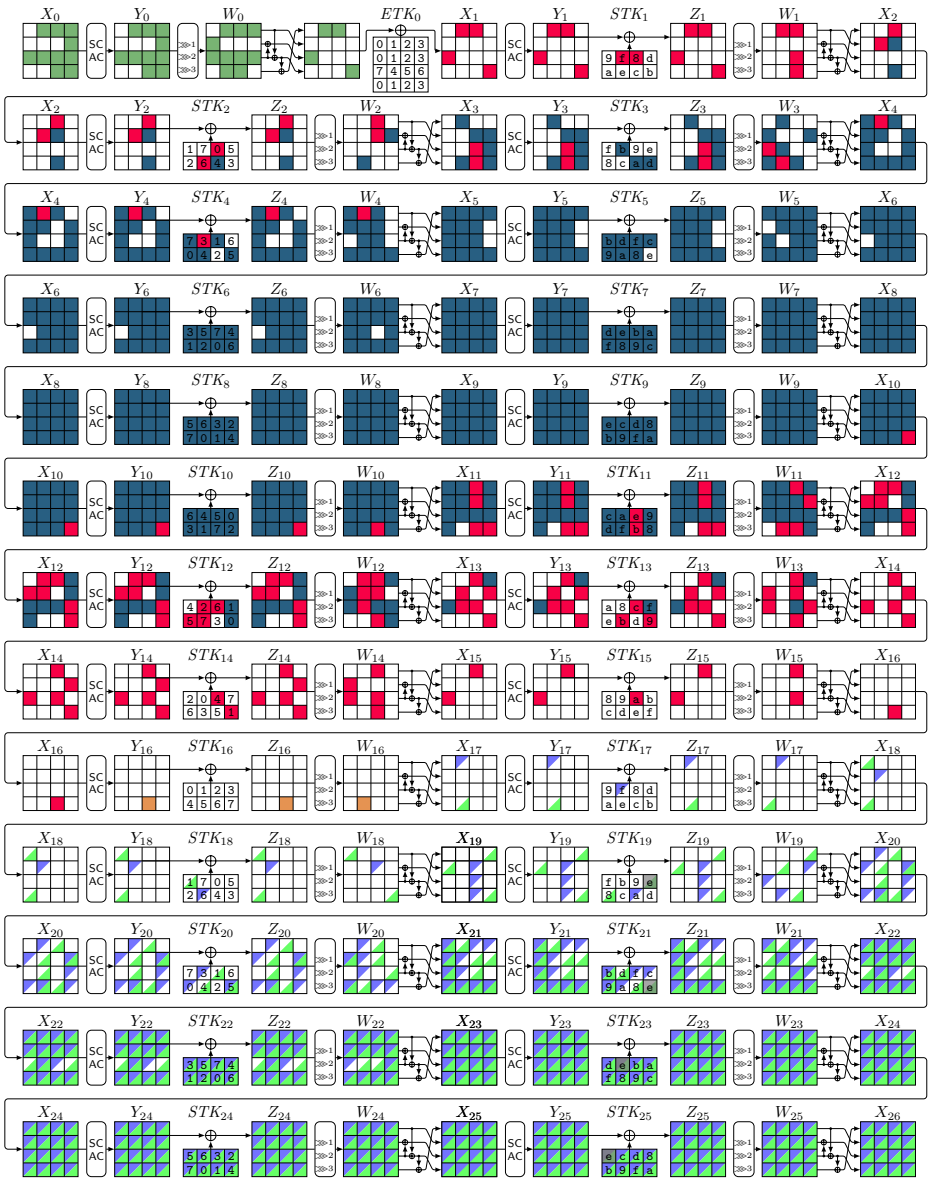

Fig. 14: ZC-based integral attack on 26 rounds of SKINNY-n-3n.

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

(a) Recovery of $X_{17}[1]$ (in Figure 14) with total complexity $2^{167.2}$.

Step	Guessed	Keys × Data = Mem	Time · Unit	Stored Texts
0	–	$2^{0} \times 2^{60} = 2^{60}$	$2^{60} \cdot 2^{0.0}$	$Z_{25}[0\text{–}15]$; $STK_{23}[1], STK_{25}[0]$
1–4	$STK_{25}[3,7]$	$2^{28} \times 2^{60} = 2^{88}$	$2^{88} \cdot 2^{-6.7}$	$STK_{23}[1]$; $W_{24}[0\text{–}15]$
5–8	$STK_{24}[3,7]$	$2^{60} \times 2^{60} = 2^{120}$	$2^{120} \cdot 2^{-6.7}$	$STK_{23}[1]$; $W_{23}[0\text{–}15]$
9	$STK_{23}[0,4]$	$2^{68} \times 2^{60} = 2^{128}$	$2^{128} \cdot 2^{-6.7}$	$Z_{23}[1\text{–}3,5\text{–}7,9\text{–}11,13\text{–}15]$; $STK_{23}[1]$; $W_{22}[0,4,8,12]$
10	$STK_{23}[5]$	$2^{72} \times 2^{60} = 2^{132}$	$2^{132} \cdot 2^{-6.7}$	$Z_{23}[2,3,6,7,10,11,14,15]$; $W_{22}[0,4,8,12,1,5,13]$
11	$STK_{23}[2,6]$	$2^{80} \times 2^{56} = 2^{136}$	$2^{140} \cdot 2^{-6.7}$	$Z_{23}[3,7,11,15]$; $W_{22}[0,4,8,12,1,5,13,2,6,14]$
12	$STK_{23}[3,7]$	$2^{88} \times 2^{56} = 2^{144}$	$2^{144} \cdot 2^{-6.7}$	$W_{22}[0,4,8,12,1,5,13,2,6,14,3,7,11,15]$
13	$STK_{22}[0,4]$	$2^{96} \times 2^{52} = 2^{148}$	$2^{152} \cdot 2^{-7.1}$	$Z_{22}[1,2,3,5,6,7,9,10,13,14,15]$; $W_{21}[0,12]$
14	$STK_{22}[1,5]$	$2^{104} \times 2^{44} = 2^{148}$	$2^{156} \cdot 2^{-6.7}$	$Z_{22}[2,3,6,7,10,14,15]$; $W_{21}[0,12,5,13]$
15	$STK_{22}[2,6]$	$2^{112} \times 2^{44} = 2^{156}$	$2^{156} \cdot 2^{-6.7}$	$Z_{22}[3,7,15]$; $W_{21}[0,12,5,13,2,6,10,14]$
16	$STK_{22}[3,7]$	$2^{120} \times 2^{44} = 2^{164}$	$2^{164} \cdot 2^{-7.1}$	$W_{21}[0,12,5,13,2,6,10,14,3,11,15]$
17	$STK_{21}[0,4]$	$2^{128} \times 2^{36} = 2^{164}$	$2^{172} \cdot 2^{-6.7}$	$Z_{21}[2,3,5,9,13,14,15]$; $W_{20}[4,12]$
18	$STK_{21}[5]$	$2^{132} \times 2^{36} = 2^{168}$	$2^{168} \cdot 2^{-7.1}$	$Z_{21}[2,3,14,15]$; $W_{20}[4,12,1,5,9]$
19	$STK_{21}[2]$	$2^{136} \times 2^{32} = 2^{168}$	$2^{172} \cdot 2^{-7.7}$	$Z_{21}[3,15]$; $W_{20}[4,12,1,5,9,14]$
20	$STK_{21}[3]$	$2^{140} \times 2^{28} = 2^{168}$	$2^{172} \cdot 2^{-7.7}$	$W_{20}[4,12,1,5,9,14,15]$
21	$STK_{20}[4]$	$2^{144} \times 2^{24} = 2^{168}$	$2^{172} \cdot 2^{-7.7}$	$Z_{20}[1,7,11,13,15]$; $W_{19}[8]$
22	$STK_{20}[1]$	$2^{148} \times 2^{20} = 2^{168}$	$2^{172} \cdot 2^{-7.7}$	$Z_{20}[7,11,15]$; $W_{19}[8,13]$
23	$STK_{20}[7]$	$2^{152} \times 2^{12} = 2^{164}$	$2^{172} \cdot 2^{-7.1}$	$W_{19}[8,13,7]$
24	–	$2^{152} \times 2^{4} = 2^{156}$	$2^{164} \cdot 2^{-7.1}$	$W_{18}[6]$
25	$STK_{18}[5]$	$2^{156} \times 2^{4} = 2^{160}$	$2^{160} \cdot 2^{-8.7}$	$W_{17}[1]$
26	–	$2^{156} \times 2^{4} = 2^{160}$	$2^{160} \cdot 2^{-8.7}$	$X_{16}[1]$

(b) Recovery of $X_{17}[13]$ (in Figure 14) with total complexity $2^{167.1}$.

Step	Guessed	Keys × Data = Mem	Time · Unit	Stored Texts
0	–	$2^{0} \times 2^{60} = 2^{60}$	$2^{60} \cdot 2^{0.0}$	$Z_{25}[0\text{–}15]$; $STK_{19}[3], STK_{21}[7], STK_{23}[1]$
1–4	$STK_{25}[3,7]$	$2^{28} \times 2^{60} = 2^{88}$	$2^{88} \cdot 2^{-6.7}$	$STK_{19}[3], STK_{21}[7], STK_{23}[1]$; $W_{24}[0\text{–}15]$
5–8	$STK_{24}[3,7]$	$2^{60} \times 2^{60} = 2^{120}$	$2^{120} \cdot 2^{-6.7}$	$STK_{19}[3], STK_{21}[7], STK_{23}[1]$; $W_{23}[0\text{–}15]$
9–12	$STK_{23}[0,2\text{–}7]$	$2^{88} \times 2^{60} = 2^{148}$	$2^{148} \cdot 2^{-6.7}$	$STK_{19}[3], STK_{21}[7]$; $W_{22}[0\text{–}7,9\text{–}15]$
13	$STK_{22}[1,5]$	$2^{96} \times 2^{60} = 2^{156}$	$2^{156} \cdot 2^{-6.7}$	$Z_{22}[0,2\text{–}4,6\text{–}8,11,12,14,15]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13]$
14	$STK_{22}[2,6]$	$2^{104} \times 2^{60} = 2^{164}$	$2^{164} \cdot 2^{-7.1}$	$Z_{22}[0,3,4,7,8,11,12,15]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14]$
15	$STK_{22}[3,7]$	$2^{112} \times 2^{52} = 2^{164}$	$2^{172} \cdot 2^{-6.7}$	$Z_{22}[0,4,8,12]$; $STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14,7,15]$
16	$STK_{22}[0,4]$	$2^{120} \times 2^{52} = 2^{172}$	$2^{172} \cdot 2^{-6.7}$	$STK_{19}[3], STK_{21}[7]$; $W_{21}[1,5,13,10,14,7,15,0,4,8,12]$
17	–	$2^{120} \times 2^{44} = 2^{164}$	$2^{172} \cdot 2^{-7.7}$	$Z_{21}[0,1,4,6,8,10,12,13,14]$; $STK_{19}[3]$; $W_{20}[11]$
18	$STK_{21}[0,4]$	$2^{128} \times 2^{36} = 2^{164}$	$2^{172} \cdot 2^{-6.7}$	$Z_{21}[1,6,10,13,14]$; $STK_{19}[3]$; $W_{20}[11,4,12]$
19	$STK_{21}[1]$	$2^{132} \times 2^{32} = 2^{164}$	$2^{168} \cdot 2^{-7.7}$	$Z_{21}[6,10,14]$; $STK_{19}[3]$; $W_{20}[11,4,12,13]$
20	$STK_{21}[6]$	$2^{136} \times 2^{28} = 2^{164}$	$2^{168} \cdot 2^{-7.1}$	$STK_{19}[3]$; $W_{20}[11,4,12,13,2,6]$
21	$STK_{20}[5]$	$2^{140} \times 2^{20} = 2^{160}$	$2^{168} \cdot 2^{-7.1}$	$Z_{20}[2,7,14]$; $STK_{19}[3]$; $W_{19}[5]$
22	$STK_{20}[2]$	$2^{144} \times 2^{16} = 2^{160}$	$2^{164} \cdot 2^{-7.7}$	$Z_{20}[7]$; $STK_{19}[3]$; $W_{19}[5,14]$
23	$STK_{20}[7]$	$2^{148} \times 2^{16} = 2^{164}$	$2^{164} \cdot 2^{-8.7}$	$STK_{19}[3]$; $W_{19}[5,14,3]$
24	–	$2^{148} \times 2^{16} = 2^{164}$	$2^{164} \cdot 2^{-8.7}$	$Z_{19}[3,15]$; $STK_{19}[3]$; $W_{18}[0]$
25	–	$2^{148} \times 2^{8} = 2^{156}$	$2^{164} \cdot 2^{-7.7}$	$W_{18}[0,15]$
26	–	$2^{148} \times 2^{4} = 2^{152}$	$2^{156} \cdot 2^{-7.7}$	$W_{17}[12]$
27	–	$2^{148} \times 2^{4} = 2^{152}$	$2^{152} \cdot 2^{-8.7}$	$X_{16}[13]$