Key Recovery Attacks on ZIP Ciphers: Application to ZIP-AES and ZIP-GIFT

Marcel Nageler[1], Debasmita Chakraborty[2], Simon Scherer[1] and Maria Eichlseder[1]

[1] Graz University of Technology, Graz, Austria
[2] NTT Social Informatics Laboratories, Tokyo, Japan

Abstract

The construction of building beyond-birthday-bound secure pseudorandom functions (PRFs) from the Xor-sum of 2 pseudorandom permutations (PRPs) has been known since EUROCRYPT 1998. However, the first concrete instance was only published recently at FSE 2022: the low-latency PRF Orthros. Subsequently, at ASIACRYPT 2024, Flórez-Gutiérrez et al. proposed the general framework of ZIP ciphers, where a block cipher $E_1 \circ E_0$ is used to construct the PRF $E_0 \oplus E_1^{-1}$. They propose the PRF ZIP-AES, as the Xor-sum of 5 AES encryption rounds and 5 decryption rounds. They discuss differential, linear, and integral distinguishers for this construction, but provide no concrete key recovery attacks. Furthermore, they propose ZIP-GIFT as a 64-bit PRF but leave cryptanalysis as future work. In this work, we provide the first third-party analysis of ZIP-AES and ZIP-GIFT. We investigate the challenges of applying classical cryptanalytic techniques to ZIP ciphers and discuss several adaptations needed for key recovery attacks in this setting. We show differential, linear, and integral key recovery attacks for both PRFs. In particular, we adapt integral key recovery techniques for ZIP ciphers, and show how to extend differential characteristics by some rounds for key recovery using truncated differential distinguishers. Our analysis deepens the understanding of the security of ZIP ciphers but does not threaten the security claims.

Keywords: PRFs, ZIP-AES, ZIP-GIFT, Cryptanalysis, Key recovery attacks

1 Introduction

In recent years, a variety of new symmetric primitives have been proposed with the goal of designing more efficient pseudorandom functions (PRFs). Building PRFs from pseudorandom permutations (PRPs) has been extensively explored, with the sum of PRPs construction being one of the central approaches. This idea was introduced by Bellare et al. at EUROCRYPT 1998 and later formally proven by Lucks at EUROCRYPT 2020 [BKR98, Luc00]. Subsequent works further strengthened these results [BI99, Pat10, Din24].

Although the construction of such PRFs is of high interest, there is little discussion on their cryptanalysis, largely because no concrete instances had been designed. The first notable proposal appears to be Orthros [BIL+21], a low-latency symmetric primitive that realizes the sum-of-permutations approach. A key challenge lies in providing a convincing security evaluation. While the designers of Orthros carefully studied its resistance against known cryptanalytic techniques, their specification did not present any key recovery

Emails: marcel.nageler@tugraz.at (Marcel Nageler), debasmita.chakraborty@ntt.com (Debasmita Chakraborty), simon.scherer@student.tugraz.at (Simon Scherer), maria.eichlseder@tugraz.at (Maria Eichlseder)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-06-12.

attack. Subsequently, differential and differential-linear weak-key attacks were proposed [LSW22, HWW+24] (the weak-key property of [LSW22] was noted in [FGGL+24]), the differential clustering effect due to the ciphertext Xor was analyzed [TISI23], and universal-key attacks based on weak-key distinguishers were proposed [HTR+25].

At ASIACRYPT 2024, Flórez-Gutiérrez et al. proposed *general practical cryptanalysis* [FGGL+24], to examine the relationship between the sum of permutations, $E_0 \oplus E_1$, and that of their composition $E_1^{-1} \circ E_0$. They demonstrate that many distinguishers on the sum construction can be translated to corresponding distinguishers on the composition. Considering differential and linear cryptanalysis, they observe that differential probabilities and linear correlations remain mostly unchanged between the constructions. However, the sum construction is only as strong as the strongest part against integral distinguishers.

To apply their framework, they introduce the ZIP construction. Let $E = E_1 \circ E_0$ denote a secure iterative block cipher; the corresponding ZIP cipher of E is then defined as $E_0 \oplus E_1^{-1}$. This design offers several appealing features. First, much cryptanalysis on E can be transferred. Second, a ZIP cipher achieves beyond-birthday-bound security in certain modes of operation. Finally, its performance is particularly attractive, as the latency is roughly half that of the underlying block cipher. As a concrete instance, the authors propose ZIP-AES, combining 5 AES encryption rounds with 5 decryption rounds.

Attacks on symmetric primitives typically consist of two components: some distinguisher rounds plus some key recovery rounds. Hence, beyond identifying effective distinguishers, it is equally important to understand how additional rounds for key recovery can be incorporated into attacks on ZIP ciphers. In [FGGL+24], the discussion primarily centered on distinguishing attacks against ZIP-AES, with only a brief mention of the challenges in constructing key recovery attacks on general ZIP ciphers. In fact, explicit key recovery attacks on ZIP-AES were left as an open problem. Furthermore, the same work introduced another variant, ZIP-GIFT, built from the lightweight 64-bit block cipher GIFT-64. However, its cryptanalysis was left as an interesting direction for future research.

In general, the trust in symmetric cryptographic primitives largely depends on the extent and depth of cryptanalysis they have undergone. A key indicator of their robustness is the gap between the maximum number of rounds successfully broken and the total number of rounds, i.e., the security margin. To get a good estimate for the security margin, it is essential to determine the strongest possible attacks for all established cryptanalytic techniques. In this context, we are interested in key recovery attacks on ZIP ciphers.

The unique feature of ZIP ciphers, combining the outputs of two separate branches, complicates key recovery attacks. Typically, an attacker appends some rounds to a distinguisher, guesses some key bits and partially decrypts the ciphertext to verify the distinguisher. For ZIP ciphers this strategy is problematic as reversing the ciphertext requires recovering the outputs of both branches; thus an attacker would have to guess the branch outputs and key bits, which prevents most techniques. Attacking on the input side is more promising: the attacker chooses the input to both branches. However, the distinguisher needs to hold for both branches, and extending two-branch distinguishers back to the plaintext is particularly difficult for chosen-plaintext attacks. The situation is further complicated as the branches use different round functions and round keys. Given these interesting challenges, we show key recovery attacks for ZIP-AES and ZIP-GIFT.

Our Contributions. We present differential, linear, and integral key recovery attacks on two concrete ZIP ciphers, ZIP-AES and ZIP-GIFT, as summarized in Table 1. These attacks provide the first concrete analysis of key recovery of ZIP ciphers, highlighting unique challenges in this setting and proposing non-trivial ideas to overcome them:

- *Differential key recovery.* Motivated by the criterion in [FGGL+24, Prop. 2] that a $(r^{\mathrm{f}}, r^{\mathrm{b}})$-round differential key recovery requires a differential characteristic on $r^{\mathrm{f}} + r^{\mathrm{b}}$ rounds with probability $p > 2^{-n}$, we analyze the differential key recovery and find

that the strongest attacks arise from truncated differential characteristics. We find that differential characteristics can be extended by some rounds of deterministic truncated propagation, while key guessing is still performed on the input side. This leads to attacks covering more rounds than a single characteristic can. We combine this with improved estimates for the clustering effect, detailed analysis of the key recovery procedure, and methods to find suitable differential distinguishers. To attack ZIP-AES-(3,2), we extend a 1-round differential distinguisher by 4 rounds of deterministic propagation. To attack ZIP-GIFT-(8,8), we extend a 12-round differential characteristic by 4 rounds of deterministic propagation.

- *Linear key recovery.* We present the first linear key recovery attacks on ZIP-AES-(3,2) and ZIP-GIFT-(10,9). As discussed in [FGGL+24], key recovery in linear cryptanalysis can be largely used unmodified. Hence, we can use the existing optimizations such as the partial-sum technique and the Fast Walsh-Hadamard Transform (FWHT).
- *Integral key recovery.* Although an integral distinguisher for ZIP-AES was presented in [FGGL+24], we provide the first integral key recovery attacks for both ZIP-AES and ZIP-GIFT. Our analysis shows that integral key recovery on ZIP ciphers requires distinguishers on both branches that are mutually compatible and permit some key dependency. We therefore develop techniques to inject key dependency into one branch's integral distinguisher while leaving the other distinguisher intact. Using this approach, we show integral key recovery attacks on ZIP-AES-(3,3) and ZIP-GIFT-(8,8).

Overall, we successfully adapt existing cryptanalytic attacks to the specifics of ZIP ciphers and present the first key recovery attacks for ZIP-AES and ZIP-GIFT. By extending the designers' security analysis with a particular focus on key recovery, we deepen the understanding of the security of ZIP ciphers but stay within the security claims. We do not claim our attacks are optimal; further analysis is possible and desirable.

We choose differential, linear, and integral cryptanalysis as they are often the most successful techniques. We discuss our choice of technique and other techniques in Section 6.

Table 1: Summary of our distinguishers and key recovery attacks. Dist.: length of the longest distinguisher. #R: number of rounds for key recovery attacks.

Cipher	Attack	Dist.	Ref.	#R	Time	Data	Mem.	Ref.
ZIP-AES	Diff.	–	–	$(3,2)$	2^{97}	2^{42}	2^{24}	Sect. 3.1
		–	–	$(2,2)$	2^{71}	2^{42}	2^{24}	Sect. 3.1
	Linear	–	–	$(3,2)$	$2^{123.7}$	$2^{123.7}$	2^{112}	Sect. 4.1
	Integral	$(4,4)$	[FGGL+24]	$(3,3)$	2^{48}	$\approx 2^{41.3}$	≈ 1	Sect. 5.1
		$(4,4)$	[FGGL+24]	$(4,2)$	2^{96}	2^{40}	2^{32}	Sect. A
ZIP-GIFT	Diff.	$(6,7)$	Sect. 3.2	$(8,8)$	$2^{119.3}$	2^{51}	$2^{23.29}$	Sect. 3.2
		$(6,7)$	Sect. 3.2	$(7,8)$	$2^{109.9}$	2^{51}	$2^{7.93}$	Sect. 3.2
	Linear	$(6,6)$	Sect. 4.2	$(10,9)$	2^{126}	2^{63}	$2^{60.18}$	Sect. 4.2
		$(6,6)$	Sect. 4.2	$(9,9)$	$2^{122.5}$	2^{62}	$2^{57.09}$	Sect. C
	Integral	$(9,9)$	Sect. 5.2	$(8,8)$	2^{126}	2^{63}	≈ 1	Sect. 5.2

Outline. In Section 2, we recall the necessary background. In Section 3, we show differential key recovery attacks on ZIP ciphers. In Section 4, we show linear key recovery attacks on ZIP ciphers. In Section 5, we show integral key recovery attacks on ZIP ciphers. We discuss our findings and conclude in Section 6 and 7. Our code is available on GitHub[1].

[1] https://github.com/isec-tugraz/zip-cipher-analysis

2 Background

2.1 Brief Specification of ZIP-AES

ZIP-AES is a pseudorandom function (PRF) proposed at ASIACRYPT 2024 by Flórez-Gutiérrez et al. [FGGL+24]. It uses the AES round function $R = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$ and its inverse R^{-1} [DR02]. Concretely, based on the 10-round AES-128, they propose

$$\mathsf{ZIP\text{-}AES}_{r^{\mathrm{f}},r^{\mathrm{b}}}(x) = \mathsf{AES}_{r^{\mathrm{f}}}(x) \oplus \mathsf{AES}^{-1}_{r^{\mathrm{b}}}(x) \quad \text{with } (r^{\mathrm{f}}, r^{\mathrm{b}}) = (5, 5).$$

They reuse the key schedule from AES, but use the round keys in a different order: $\mathsf{AES}_5(x)$ uses the round keys $K_0, \ldots, K_5$ while $\mathsf{AES}^{-1}_5(x)$ uses $K_6, \ldots, K_{10}, 0^{128}$. We depict a round-reduced version, $\mathsf{ZIP\text{-}AES}_{2,2}$, in Figure 1.

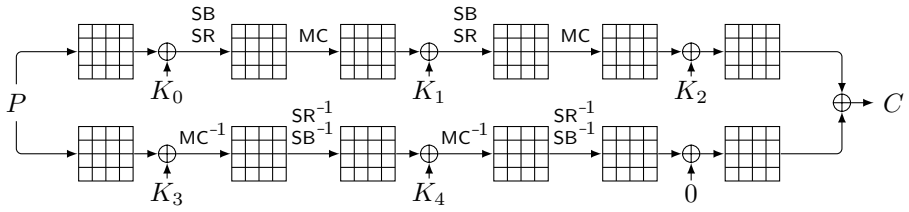

Figure 1: Specification of ZIP-AES, illustrated for $(r^{\mathrm{f}}, r^{\mathrm{b}}) = (2, 2)$.

2.2 Brief Specification of ZIP-GIFT

ZIP-GIFT is a 64-bit PRF, derived from the GIFT-64 [BPP+17] block cipher:

$$\mathsf{ZIP\text{-}GIFT}_{r^{\mathrm{f}},r^{\mathrm{b}}}(x) = \mathsf{GIFT}_{r^{\mathrm{f}}}(x) \oplus \mathsf{GIFT}^{-1}_{r^{\mathrm{b}}}(x), \quad \text{with } (r^{\mathrm{f}}, r^{\mathrm{b}}) = (14, 14).$$

$\mathsf{GIFT}_{r^{\mathrm{f}}}$ uses the GIFT-64 round function: Subcells, PermBits, and AddRoundKey, where the GIFT S-box $\mathcal{S}$ is applied to every nibble of the state, the bit permutation P_{64} is applied to the whole state, and finally the round key is XORed to the two leftmost bits of every nibble. Similarly, $\mathsf{GIFT}^{-1}_{r^{\mathrm{b}}}$ uses GIFT-64's inverse round function, with the inverse S-box denoted $\mathcal{S}'$ and inverse bit permutation P^{-1}_{64}.

For the key schedule, the round key K_i for the forward branch is first extracted before updating the key state with the key update function L as in GIFT:

$$k_7 \parallel k_6 \parallel k_5 \parallel \cdots \parallel k_1 \parallel k_0 \leftarrow k_1 \ggg 2 \parallel k_0 \ggg 12 \parallel \cdots \parallel k_3 \parallel k_2,$$

where $K_0 = K$. The round keys of the backward branch K_{-i} are created using the inverse function L^{-1}. We depict reduced-round ZIP-GIFT in Figure 2. Note that, consistent with GIFT-64, the forward branch of ZIP-GIFT does not include a pre-whitening key.

We also introduce an equivalent-key representation of ZIP-GIFT. In the forward branch, we denote $EY_{-i} = P_{64}(Y_i)$ for $0 \le i \le r^{\mathrm{f}} - 1$. Similarly, in the backward branch, we define $EX_{-j+1} = P^{-1}_{64}(X_{-j+1})$ and $EK_{-j} = P^{-1}_{64}(K_{-j})$ for $1 \le j \le r^{\mathrm{b}}$. For our differential and linear attacks on ZIP-GIFT, we adopt this equivalent-key view for simplicity.

2.3 Overview of Cryptanalytic Techniques on Symmetric Primitives

In this work, we focus on differential, linear, and integral cryptanalysis. This section provides a concise overview of these techniques.

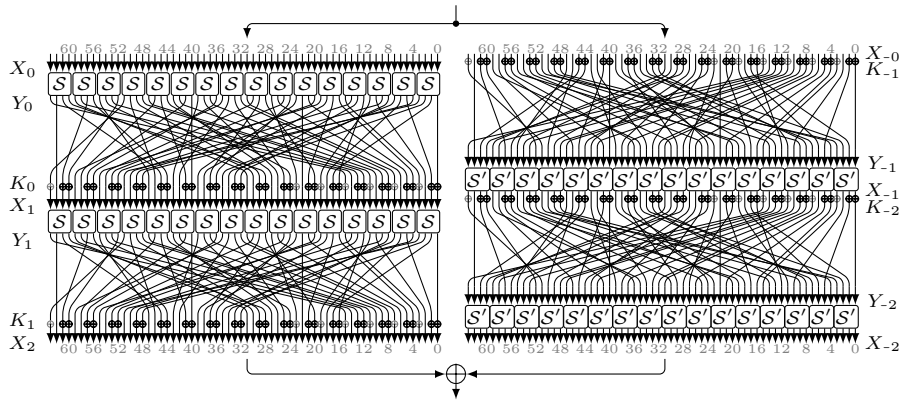

Figure 2: Specification of ZIP-GIFT, illustrated for $(r^{\mathrm{f}}, r^{\mathrm{b}}) = (2, 2)$.

Differential Cryptanalysis [BS91]. In differential cryptanalysis, a statistical chosen-plaintext attack, the distinguisher is given by the differences $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ such that Δ_{in} propagates to Δ_{out} through the cipher's rounds with probability above the random baseline 2^{-n}. Given a function f, the probability can be calculated as:

$$\Pr(\Delta_{\mathrm{in}} \xrightarrow{f} \Delta_{\mathrm{out}}) = \frac{|\{x \in \mathbb{F}_2^n \mid f(x) \oplus f(x \oplus \Delta_{\mathrm{in}}) = \Delta_{\mathrm{out}}\}|}{2^n}$$

To recover keys, the distinguisher is typically extended by appending rounds in one or both directions; these extra rounds allow the attacker to identify partial subkeys for which many plaintext-ciphertext pairs follow the differential. Over the years, the techniques for the key recovery have been improved [KMNP10, QHS16, BDD+24].

Linear Cryptanalysis [Mat93]. Linear cryptanalysis is a statistical known-plaintext technique that exploits linear approximations between bits of the plaintext, key, and ciphertext. A *linear approximation* is an Xor-sum of input bits selected by Γ_{in} that correlates with the Xor-sum of output bits selected by Γ_{out} after application of the cipher. Its quality is measured by the correlation, for a function f:

$$\mathrm{cor}(\Gamma_{\mathrm{in}}, \Gamma_{\mathrm{out}}) = \Pr_x\big[\Gamma_{\mathrm{in}} \cdot x = \Gamma_{\mathrm{out}} \cdot f(x)\big] - \Pr_x\big[\Gamma_{\mathrm{in}} \cdot x \neq \Gamma_{\mathrm{out}} \cdot f(x)\big].$$

Matsui's Algorithm 2 provides a practical last-round partial-key recovery method for such approximations [Mat93]. Subsequent improvements, most notably FFT-based optimization by Collard et al. [CSQ07] and the partial-sums technique by Ferguson et al. [FKL+00], substantially lowered the time complexity of the partial key recovery.

Integral Cryptanalysis [Lai94, DKR97, Tod15]. Integral cryptanalysis exploits input structures that produce key-independent behavior in certain output bits when summed over all inputs. Modeling a symmetric primitive as a vectorial Boolean function over key and data, one chooses d active bits forming a d-dimensional subspace; the Xor-sum of outputs then represents the d-th derivative. Output bits whose Algebraic Normal Form (ANF) lacks monomials combining active inputs and key bits are constant, yielding balanced properties. To find specific monomials in the ANF, monomial prediction is commonly used [Tod15, HSWW20]. Moreover, key recovery rounds can be added by evaluating the zero-sum for all guesses of the κ key bits needed to reach the targets, with various optimizations proposed [FKL+00, TA14].

3 Differential Cryptanalysis on ZIP Ciphers

In this section, we explore differential cryptanalysis of ZIP ciphers. We begin by examining how differential characteristics of the Xor of two keyed permutations relate to those of their composition. Next, we outline the necessary conditions for differential key recovery attacks on ZIP ciphers, as initially studied in [FGGL+24]. Then, we show differential key recovery attacks on ZIP-AES and ZIP-GIFT.

In [FGGL+24], the authors show that differential characteristics of the sum construction are related to those of the composition:

Proposition 1 ([FGGL+24]). *Let P, Q be two keyed permutations over $\mathbb{F}_2^n$, and let $\mathsf{F} := P \oplus Q$ and $\mathsf{S} := Q \circ P^{-1}$. For each differential characteristic with probability p traversing F, there is a characteristic with the same probability p traversing S.*

Proposition 1 shows that every differential characteristic of probability p for a ZIP cipher corresponds to one with the same probability p for the composition. Since any 4-round differential characteristic for AES has at most probability 2^{-150}, ZIP-AES cannot admit any distinguishable $(2,2)$-round characteristics. Likewise, the longest differential characteristic for GIFT-64 spans 13 rounds with probability 2^{-62} [LWZZ19]. Hence, no distinguishable differential characteristic for $(7,7)$-round ZIP-GIFT can exist.

In [FGGL+24], the authors highlight the inherent difficulty of mounting differential key recovery attacks on ZIP ciphers, as one must always prepend rounds above the differential distinguisher in both branches, leading to Proposition 2:

Proposition 2 ([FGGL+24]). *Let $E^{\oplus} = (E^f_{dist} \circ E^f_{key}) \oplus (E^b_{dist} \circ E^b_{key})$. We consider a differential key recovery attack where the input differences of E^f_{dist} and E^b_{dist} are fixed to Δ^f_{dist} and Δ^b_{dist}, respectively, and the output difference is $\Delta^{\oplus}$. The necessary key material from E^f_{key}, and E^b_{key} is guessed. This attack works only if*

$$\Pr((\Delta^f_{dist}, \Delta^b_{dist}) \xrightarrow{E^f_{dist} \oplus E^b_{dist}} \Delta^{\oplus}) \cdot Prob(\Delta^f_{dist} \xrightarrow{E^b_{key} \circ (E^f_{key})^{-1}} \Delta^b_{dist}) > 2^{-n}.$$

From Proposition 2, we see that the key recovery rounds also count towards the probability of the distinguisher. To simplify our descriptions, we will phrase our attacks in terms of a differential distinguisher over the whole cipher with the same input difference for both branches $\Delta \xrightarrow{E^{\oplus}} \Delta^{\oplus}$ with probability $p > 2^{-n}$. This is similar to what Biham and Shamir call a 0R attack [BS91].

Furthermore, Proposition 2 [FGGL+24, Prop. 2] seems to imply that the differential characteristic needs to span all attacked rounds. However, we find that extending the characteristic to a truncated differential distinguisher is possible. While the authors of ZIP-AES discuss truncated differential attacks in general, they do not perform concrete analysis for ZIP-AES and do not consider extending a differential characteristic with a few rounds of truncated (probability 1) propagation.

Extending a differential characteristic with truncated propagation presents some unique challenges. First, the number of possible ciphertext differences grows quickly, leading to pairs which have an expected ciphertext difference but do not actually follow the differential characteristic, i.e., decreasing the signal-to-noise ratio. Second, we can only perform key recovery at the output side if we can derive information about the internal values simultaneously for both branches. That is, only when an active S-box (with $p < 1$) in the forward branch and an active S-box (with $p < 1$) in the backward branch are used to calculate some common output bits, can we learn something about the involved key. In the case where the output of an active S-box is Xored with the output of an inactive S-box, we can restrict the solution set for the active S-box, but since we do not know anything about the output of the inactive S-box, we cannot learn anything about the key.

Since we perform deterministic truncated extension to attack more rounds, recovering key bits at the output is hardly possible. Hence, we have to recover the key at the input side.

The problem of recovering the key at the input side is that we cannot efficiently distinguish between signal (pairs that actually follow the distinguisher) and noise (pairs that do not follow the distinguisher but have an expected ciphertext difference). While we can guess some partial round key and verify whether some part of differential characteristic is followed, this usually only allows us to restrict some value of key bits for this pair but not to disregard the whole pair as noise. Therefore, we are mostly stuck with analyzing all pairs that could follow the distinguisher based on the ciphertext difference. This leads to the effect that the number of surviving pairs is a multiplicative factor in the *total complexity* of the attack, as opposed to only influencing the recovery of some partial key material and not influencing the brute force step.

To address this difficulty we propose two strategies. First, we perform careful analysis of the set of ciphertext differences to make it as small as possible. For ZIP-AES, we include the count of ciphertext differences in the optimization goal when searching for the distinguisher. For ZIP-GIFT, we also include the count of ciphertext bits with fixed difference, while also considering undisturbed bits of the GIFT S-box. Then, we perform a more precise analysis of the concrete set of 64-bit ciphertext differences which shows that the set is smaller by a factor of $2^{1.5}$ than what one would expect based on the count of bits with fixed difference. Second, we perform precise analysis of every step of the key recovery procedure. Because the usual bottleneck of the final brute force is multiplicative instead of additive, every step can possibly slow down the entire attack.

3.1 Differential Key Recovery Attacks on ZIP-AES

We now show a key recovery attack on (3,2)-round ZIP-AES leveraging truncated differential cryptanalysis using 2^{42} data, 2^{97} time with 86% success probability. As shown in Figure 3, we recover the key in the first round of the forward branch. We note that the designers' analysis of ZIP-AES against differential attacks is limited to a discussion on bounds for probabilities of differential characteristics.

Characteristic. To find our differential distinguisher, we use MiniZinc to model truncated cellwise differential characteristics for AES, where an S-box transition can either be probabilistic or deterministic. We constrain at least one probabilistic S-box in the beginning to provide a key dependency and optimize for as few active cells at the ciphertext as possible (counting before the last MixColumns for the forward branch) and as few probabilistically active S-boxes as possible.

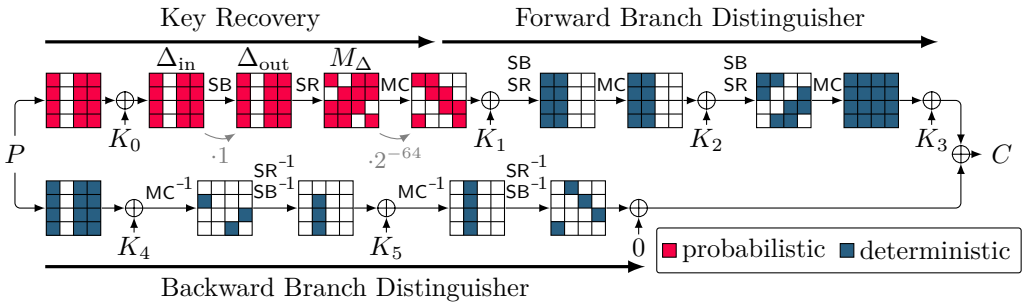

Figure 3: Truncated differential distinguisher for (3,2)-rounds of ZIP-AES, where key recovery is performed in the first round of the forward branch.

Data Complexity. For a successful attack, we need to identify a plaintext pair that follows the *forward* branch (probabilistically) and the *backward* branch (deterministically). First, consider the MixColumns operation of the forward branch. Here, every input column has one inactive input cell and two inactive output cells. Consequently, there are $2^8 - 1$ valid non-zero transitions per column, resulting in $|M_\Delta| \approx 2^{32}$ transitions for the whole state. Since the plaintext has 12 active cells, the probability for valid MixColumns transition is $p = \frac{2^{32}}{2^{96}} = 2^{-64}$. We aim for $n = 2^{65}$ pairs, so we expect at least one valid pair with probability $1 - e^{-2} \approx 0.86$. Second, consider the first MixColumns in the backward branch, which is satisfied by only 2^{24} transitions. Since only the key addition precedes MixColumns, we describe the plaintext differences by a linear space $\mathcal{L}$ of dimension 24. For an AES state P_o, the affine space $\mathcal{L} \oplus P_o$ gives valid plaintexts. As each pair follows the input difference and MixColumns pattern, we get $\binom{2^{24}}{2} \approx 2^{47}$ pairs. By using 2^{18} different P_o we get $2^{47} \cdot 2^{18} = 2^{65}$ pairs yielding a total data complexity of $2^{24} \cdot 2^{18} = 2^{42}$ plaintexts.

Filtering. We filter the obtained ciphertext pairs by using the output difference. Consider that output difference in the forward branch can be described by a linear space of dimension 64. Similarly, the output difference of the backward branch can be described by a linear space of dimension 32. Combined, we obtain a linear space of dimension 96, leading to a ciphertext filter probability of 2^{-32} yielding about 2^{33} surviving pairs.

Attack Procedure. For each of the 2^{33} surviving (P, P', C, C'), do the following:

1. Let $\Delta_{\text{in}} = P \oplus P'$ and let M_Δ denote the set of all input differences for the first MC in the forward branch that, after applying MC, yield the prescribed output difference pattern. The size of M_Δ is about 2^{32}.

2. For every $\Delta_{\text{R}} \in M_\Delta$, let $\Delta_{\text{out}} = \mathsf{SR}^{-1}(\Delta_{\text{R}})$.

   (a) Let $\Delta_{\text{in}}^{(i)}, \Delta_{\text{out}}^{(i)}, P^{(i)}$ denote cell i of $\Delta_{\text{in}}, \Delta_{\text{out}}, P$, respectively.

   (b) For every i with $\Delta_{\text{in}}^{(i)} \neq 0$ calculate

   $$X_i = \{x : \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \Delta_{\text{in}}^{(i)}) = \Delta_{\text{out}}^{(i)}\}, \quad K_i = \{P^{(i)} \oplus x : x \in X_i\}.$$

   (c) Combine all sets K_i with the Cartesian product, producing one 96-bit candidate key on average. Recover the remaining 32 bits of the key using brute force by testing each candidate key through encryption of the plaintext.

Runtime Evaluation. On average, we expect that $|X_i| = 1$ in step 2b because about half of the entries in each AES DDT row equal 2. Therefore, for each of the 2^{33} surviving pairs, we expect one key candidate for every element in M_Δ, and hence every surviving pair suggests $|M_\Delta| \approx 2^{32}$ 96-bit key candidates. To recover the remaining 32 bits, we need about $2^{33} \cdot 2^{32} \cdot 2^{32} = 2^{97}$ brute force attempts.

Attack on the (2,2)-round Balanced Scenario Based on the above attack, we can construct an attack on (2,2)-round ZIP-AES by removing the last round in the forward branch. Hence, we receive a stronger ciphertext filter of 2^{-48}. Since the data complexity remains the same, the number of surviving pairs reduces to $\frac{2^{65}}{2^{48}} = 2^{17}$, leading to a total runtime of $2^{17} \cdot 2^{32} \cdot 2^{32} = 2^{71}$ brute force attempts.

3.2 Differential Key Recovery Attacks on ZIP-GIFT

For GIFT-64, the longest known differential characteristics cover 13 rounds. Due to Proposition 1, this directly translates into differential characteristics of at most $(6, 7)$ or $(7, 6)$ rounds for ZIP-GIFT. Consequently, following Proposition 2, differential key recovery attacks on ZIP-GIFT are seemingly limited to at most $(7, 6)$ or $(6, 7)$ rounds.

This reasoning, however, is based on the assumption that differential key recovery must rely on a single differential characteristic satisfying the probability condition of Proposition 2. Instead of relying solely on a single differential characteristic, we combine a single differential characteristic of ZIP-GIFT with a truncated extension at the output. This approach enables key recovery beyond $(7, 6)$ or $(6, 7)$ rounds for ZIP-GIFT.

Specifically, we mount a differential key recovery attack on $(8, 8)$ rounds of ZIP-GIFT with 2^{51} data and $2^{119.3}$ time. For comparison, the best known differential attack on GIFT-64 extends a 13-round distinguisher to 20 rounds requiring $2^{62.58}$ data and $2^{125.50}$ time [SWW21b]. The remainder of this section details the underlying differential characteristic, the truncated extension strategy, and the resulting key recovery procedure for ZIP-GIFT.

Characteristic. The longest differential trail we find is a $(6, 7)$-round trail for ZIP-GIFT with probability 2^{-62}, consistent with [LWZZ19]. Our key recovery attack is based on an (8,8)-round truncated differential characteristic for ZIP-GIFT with probability p=2^{-60} and $2^{37.3}$ possible ciphertext differences. To find this characteristic, we use MiniZinc to model differential characteristics (without truncation) for (6,6)-round ZIP-GIFT and (2,2) rounds of deterministic truncated differential propagation, where we consider the undisturbed bits of the GIFT S-box and inverse S-box. We then optimize for a minimum number of unknown difference bits at the ciphertext while requiring a fixed near-optimal differential characteristic probability. As GIFT is known to exhibit key-dependent behavior for differential characteristics [PT22], we generate multiple characteristics and analyze them for key dependencies with AutoDiVer [NGJE25], to select one without key dependencies.

Our $(8, 8)$-round truncated differential characteristic $\Delta \xrightarrow{E^{\oplus}} \Delta^{\oplus}$ with probability $p = 2^{-60}$, can be described by a single input difference Δ and a set of output differences $\Delta^{\oplus}$ with known zero difference at 15 bits:

$$\Delta = \texttt{0a00 0000 0a00 0000} \quad \rightarrow \quad \Delta^{\oplus} \,\&\, \texttt{0000 2aaa 0000 aaaa} = 0,$$

where & denotes bitwise And. This yields an initial upper bound on the count of ciphertext differences of $|\Delta^{\oplus}| \leq 2^{49}$. To improve this bound, we start by listing all possible 64-bit differences in the forward and backward branches just before the Xor. Concretely, we start with the differences after (6,6) rounds of differential propagation: (0000 0800 0000 0800, 0000 00a0 0000 00a0). For both differences, we generate all possible 64-bit differences after two rounds, resulting in two sets Δ_O^{f} and Δ_O^{b}, with $|\Delta_O^{\mathrm{f}}| = 2^{20.98}$ and $|\Delta_O^{\mathrm{b}}| = 2^{17.93}$. This gives an improved upper bound of $|\Delta^{\oplus}| \leq |\Delta_O^{\mathrm{f}}| \cdot |\Delta_O^{\mathrm{b}}| = 2^{38.82}$. Counting the distinct values of $\Delta^{\oplus}$ via a hash table is very expensive due to memory requirements of at least 1.22 TiB, hence we approximate the cardinality with the HyperLogLog algorithm [FFGM07, HNH13]. The algorithm provides an estimate of $|\Delta^{\oplus}| \approx 2^{37.29}$, which is about $2^{1.5}$ better than the previous upper bound. As this algorithm provides a tradeoff between memory usage and accuracy, we set the parameter $b = 18$ to get a precise estimate, resulting in a standard error of $\frac{1.04}{\sqrt{2^b}} \approx 2^{-8.9}$ corresponding to a standard deviation of $|\Delta^{\oplus}| \cdot 2^{-8.9} \approx 2^{28.4}$.

Notation. Let $\mathcal{S}_r^{\mathtt{i}}$ denote S-box $\mathtt{i}$ in round r. Let $\mathrm{XDDT}^{\mathrm{f}}_{\Delta_{\mathrm{i}} \to \Delta_{\mathrm{o}}} = \{\, x \mid \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \Delta_{\mathrm{i}}) = \Delta_{\mathrm{o}} \}$ and $\mathrm{XDDT}^{\mathrm{b}}_{\Delta_{\mathrm{i}} \to \Delta_{\mathrm{o}}} = \{\, x \mid \mathcal{S}'(x) \oplus \mathcal{S}'(x \oplus \Delta_{\mathrm{i}}) = \Delta_{\mathrm{o}} \}$ denote the solution sets for the GIFT S-box $\mathcal{S}$ and its inverse $\mathcal{S}'$. We define $\mathcal{X}_{\mathtt{0***}} = \{x | \mathcal{S}(x) = \mathtt{0***}\}$, $\mathcal{X}_{\mathtt{*1**}} = \{x | \mathcal{S}(x) = \mathtt{*1**}\}$ and $\mathcal{Y}_{\mathtt{*0**}} = \{x | \mathcal{S}'(x) = \mathtt{*0**}\}$, $\mathcal{Y}_{\mathtt{*1**}} = \{x | \mathcal{S}'(x) = \mathtt{*1**}\}$.

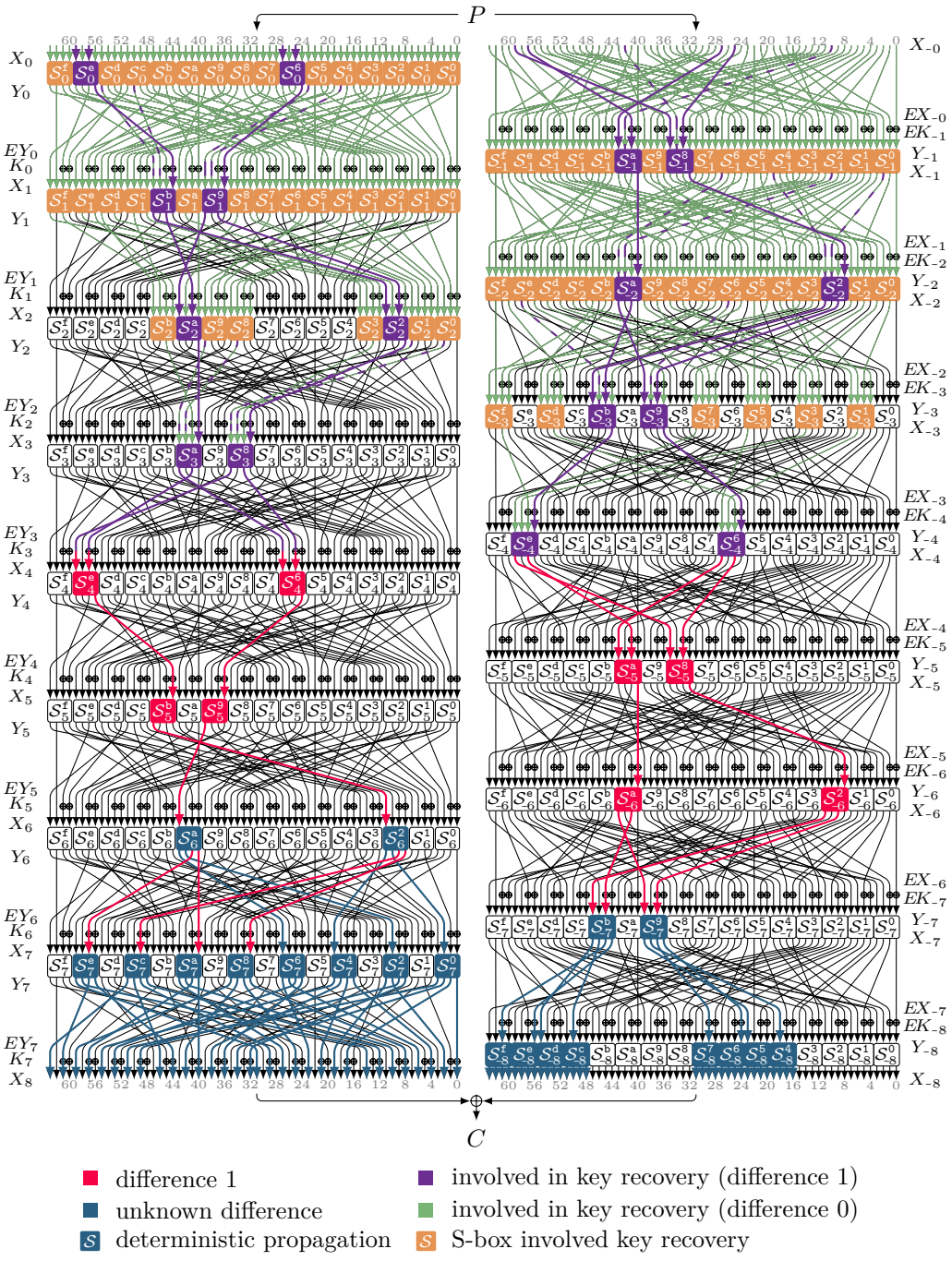

Figure 4: Differential attack on ZIP-GIFT-(8,8) where the key recovery is performed on the first 4 rounds of the forward and the backward branch.

Data Complexity. To mount a valid differential key recovery attack, we need sufficiently many plaintext pairs (P, P') with $P \oplus P' = \Delta$ such that we get at least one pair that actually follows the distinguisher. Based on the partial key addition and missing initial whitening key in the forward branch, we will now reduce the data complexity by a factor of 2^{10}. Since the S-boxes $\mathcal{S}_0^6$ and $\mathcal{S}_0^e$ are not keyed, we set $P[24..27] \in \mathrm{XDDT}^{\mathrm{f}}_{\mathtt{a}\to\mathtt{1}}$ and $P[56..59] \in \mathrm{XDDT}^{\mathrm{f}}_{\mathtt{a}\to\mathtt{1}}$ improving the data complexity by 2^4. For the $\mathtt{1} \to \mathtt{a}$ transition in S-boxes $\mathcal{S}_1^0$ and $\mathcal{S}_1^{\mathtt{b}}$ with $\mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}$, we need $X_1[47, 46, 39, 38,] = (1, 0, 1, 0)$. By setting $P[56..59] \in \mathcal{X}_{\mathtt{0***}}$, $P[48..51] \in \mathcal{X}_{\mathtt{*1**}}$, $P[20..23] \in \mathcal{X}_{\mathtt{0***}}$, $P[16..19] \in \mathcal{X}_{\mathtt{*1**}}$, we gain an improvement of another 2^4. Finally, for the $\mathtt{a} \to \mathtt{1}$ transition in S-boxes $\mathcal{S}_{\text{-}1}^{\mathtt{a}}$ and $\mathcal{S}_{\text{-}1}^{\mathtt{a}}$ with $\mathrm{XDDT}^{\mathrm{b}}_{\mathtt{a}\to\mathtt{1}} = \{\mathtt{3}, \mathtt{9}\}$, we set $P[10] = 1$ and $P[42] = 0$, leading to a final improvement of 2^2. Overall, we need 2^{50} pairs or 2^{51} encryptions for a success probability of about 63 %.

Key Recovery Steps. For each of the 2^{50} ciphertext pairs, verify whether it is part of $\Delta^{\oplus}$, by iterating over Δ_O^{f}, XORING with ΔC, and performing a lookup in Δ_O^{b}. We expect $N \cdot \frac{|\Delta^{\oplus}|}{2^{64}} \approx 2^{23.3}$ surviving pairs. For each surviving pair, we repeat the following steps:

1. For the transition $\mathtt{1} \to \mathtt{a}$ in $\mathcal{S}_1^9$ and $\mathcal{S}_1^{\mathtt{b}}$ with $\mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}$, we find $X_1[24..27] \in \{\mathtt{6}, \mathtt{7}\}$ and $X_1[44..47] \in \{\mathtt{6}, \mathtt{7}\}$. Hence, $X_1[25] = X_1[45] = 1$, giving us two key bits: $K_0[19] = EY_0[25] \oplus X_1[25]$ and $K_0[23] = EY_0[45] \oplus X_1[45]$. After recovering these bits, we guess the remaining 30 bits of K_0 and compute EY_1 for each guess.

2. For the transition $\mathtt{a} \to \mathtt{1}$ in $\mathcal{S}_2^2$ and $\mathcal{S}_2^{\mathtt{a}}$ with $\mathrm{XDDT}^{\mathrm{f}}_{\mathtt{a}\to\mathtt{1}} = \{\mathtt{3}, \mathtt{5}, \mathtt{9}, \mathtt{f}\}$, we find $X_2[8..11] \in \{\mathtt{3}, \mathtt{5}, \mathtt{9}, \mathtt{f}\}$ and $X_2[40..43] \in \{\mathtt{3}, \mathtt{5}, \mathtt{9}, \mathtt{f}\}$. As we know the upper two bits of each nibble and as they uniquely identify the lower two bits in the solution set, we can uniquely determine the key bits $K_1[4, 5, 20, 21]$.

   For the transition $\mathtt{1} \to \mathtt{a}$ in $\mathcal{S}_3^8$ and $\mathcal{S}_3^{\mathtt{a}}$ with $\mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}$, we find 6 fixed bits: $X_3[35, 34, 33, 43, 42, 41] = 0, 1, 1, 0, 1, 1$. The bits $X_3[35, 34, 43, 32]$ restrict one output bit of $\mathcal{S}_2^0$, $\mathcal{S}_2^1$, $\mathcal{S}_2^8$, and $\mathcal{S}_2^9$, restricting the input to some set of 8 values each. In turn, this reduces the options for $K_1[0..3, 16..19]$ from 2^8 to 2^4. With these 2 reductions, we complete the key guess for $K_1[0..7, 16..23]$, enumerating 2^8 candidates, allowing us to calculate $EY_2[32..35, 40..43]$.

3. The knowledge of $X_3[33, 41]$ from the previous step, combined with $EY_2[33, 41]$ allows us to calculate the 2 key bits $K_2[17, 21]$. At this point, we are considering 2^{38} candidates for 50 bits of key material for each surviving pair, such that the differential characteristic in the first 4 rounds of the forward branch is followed. Next, we analyze the backward branch.

4. For the transition $\mathtt{a} \to \mathtt{1}$ in $\mathcal{S}_{\text{-}1}^8$ and $\mathcal{S}_{\text{-}1}^{\mathtt{a}}$ with $\mathrm{XDDT}^{\mathrm{b}}_{\mathtt{a}\to\mathtt{1}} = \{\mathtt{3}, \mathtt{9}\}$, we find $Y_{\text{-}1}[32] = Y_{\text{-}1}[40] = 1$. Additionally, we derive $Y_{\text{-}1}[33, 41]$ from the known values $Y_{\text{-}1}[35, 43]$ via the solution set. This allows us to calculate $EK_{\text{-}1}[16, 17, 20, 21]$.

   For the transition $\mathtt{1} \to \mathtt{a}$ in $\mathcal{S}_{\text{-}2}^2$ and $\mathcal{S}_{\text{-}2}^{\mathtt{a}}$ with $\mathrm{XDDT}^{\mathrm{b}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$, we find $Y_{\text{-}2}[10, 11, 42, 43] = (1, 1, 1, 1)$. This restricts one output bit of the S-boxes $\mathcal{S}_{\text{-}1}^0$, $\mathcal{S}_{\text{-}1}^2$, $\mathcal{S}_{\text{-}1}^4$, $\mathcal{S}_{\text{-}1}^6$, $\mathcal{S}_{\text{-}1}^8$ each, leading to $Y_{\text{-}1}[0..3] \in \mathcal{Y}_{\mathtt{*1**}}$, $Y_{\text{-}1}[8..11] \in \mathcal{Y}_{\mathtt{*1**}}$, $Y_{\text{-}1}[16..19] \in \mathcal{Y}_{\mathtt{1***}}$, and $Y_{\text{-}1}[24..27] \in \mathcal{Y}_{\mathtt{1***}}$. This restricts the options for $EK_{\text{-}1}[0, 1, 4, 5, 8, 9, 12, 13]$ from 2^8 to 2^4. We iterate over the 2^{24} candidates for $EK_{\text{-}1}$ and calculate $EX_{\text{-}1}$

5. For the transition $\mathtt{a} \to \mathtt{1}$ in $\mathcal{S}_{\text{-}3}^9$ and $\mathcal{S}_{\text{-}3}^{\mathtt{b}}$ with $\mathrm{XDDT}^{\mathrm{b}}_{\mathtt{a}\to\mathtt{1}} = \{\mathtt{3}, \mathtt{9}\}$, we find $Y_{\text{-}3}[37, 46] = (0, 0)$. This restricts one output bit of $\mathcal{S}_{\text{-}1}^0$, $\mathcal{S}_{\text{-}1}^2$ each, leading to $Y_{\text{-}2}[24..27] \in \mathcal{Y}_{\mathtt{*0**}}$ and $Y_{\text{-}2}[56..59] \in \mathcal{Y}_{\mathtt{*0**}}$. This restricts the options for $EK_{\text{-}2}[12, 13, 28, 29]$ from 2^4 to 2^2. Additionally, due to the key schedule, $EK_{\text{-}2}[5, 6, 22, 23]$ is known from Step 3. Overall, we iterate over 2^{24} candidates for $EK_{\text{-}2}$ and calculate $EX_{\text{-}2}$.

6. The transition $\mathtt{a} \to \mathtt{1}$ in $\mathcal{S}^{9}_{\text{-}3}$ and $\mathcal{S}^{\mathsf{b}}_{\text{-}3}$, which we analyzed in the previous step, also helps for recovering $EK_{\text{-}3}$. Due to the solution set, we know $Y_{\text{-}3}[36, 44] = (1, 1)$, which allows us to calculate $EK_{\text{-}3}[18, 22] = Y_{\text{-}3}[36, 44] \oplus EX_{\text{-}2}[36, 44]$.

   For the transition $\mathtt{1} \to \mathtt{a}$ in $\mathcal{S}^{6}_{\text{-}4}$ and $\mathcal{S}^{\mathsf{e}}_{\text{-}4}$ with $\text{XDDT}^{\mathsf{b}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$, we find $Y_{\text{-}4}[26, 27, 58, 59] = (1, 1, 1, 1)$. This restricts one output bit of $\mathcal{S}^{1}_{\text{-}3}$, $\mathcal{S}^{3}_{\text{-}3}$, $\mathcal{S}^{5}_{\text{-}3}$, $\mathcal{S}^{7}_{\text{-}3}$ each, leading to $Y_{\text{-}3}[4..7] \in \mathcal{Y}_{\mathtt{*1**}}$, $Y_{\text{-}3}[12..15] \in \mathcal{Y}_{\mathtt{*1**}}$, $Y_{\text{-}3}[20..23] \in \mathcal{Y}_{\mathtt{1***}}$, $Y_{\text{-}3}[28..31] \in \mathcal{Y}_{\mathtt{1***}}$, This restricts the options for $EK_{\text{-}3}[2, 3, 6, 7, 10, 11, 14, 15]$ from 2^8 to 2^4.

   As we already know $EK_{\text{-}3}[2, 6, 10, 14, 26, 27, 30, 31]$ from Step 2 via the key schedule, we only need $EK_{\text{-}3}[3, 7, 11, 15]$, which we learn due to the restrictions above.

7. For the backward branch, we have guessed an additional 2^{48} key candidates of size 68 bit. Considering both branches, we are iterating over 2^{86} key candidates of size 118 bit. Now, we perform brute-force search over the remaining 2^{10} keys.

Complexity analysis. The main loop of the attack needs 2^{96} operations and is repeated for about $2^{23.3}$ surviving pairs, leading to an overall time complexity of $2^{96+23.3} \approx 2^{119.3}$. We can also shorten our attack to ZIP-GIFT-$(7, 8)$, achieving a time complexity of $\approx 2^{103.9}$.

4 Linear Cryptanalysis on ZIP Ciphers

This section analyzes the security of the ZIP ciphers, ZIP-AES and ZIP-GIFT, against one of the most established attack families—linear cryptanalysis.

Linear Characteristics. In [FGGL+24], the authors show that the linear trails of the sum construction are related to those of the composition construction:

Proposition 3 ([FGGL+24])**.** *Let P, Q be two keyed permutations over $\mathbb{F}_2^n$, and let $\mathsf{F} := P \oplus Q$ and $\mathsf{S}^* := Q^{-1} \circ P$. For each linear trail with correlation c traversing F, there is a linear trail with the same correlation c traversing S^*.*

It is well known that linear trails for 4-round AES involve at least 25 active S-boxes, leading to a squared correlation of 2^{-150}. Consequently, for ZIP-AES, no $(2, 2)$-round linear trails exist. For GIFT-64, Sun et al. show that the best squared correlation for 13-round trails is 2^{-68} [SWW21a]. Hence, the longest linear trails for ZIP-GIFT cover $(6, 6)$ rounds.

On Key Recovery in Linear Cryptanalysis. As noted in [FGGL+24], linear key recovery attacks on ZIP ciphers can be achieved by prepending rounds before the linear distinguisher in both branches. Concretely, given a linear characteristic of $E^{\mathsf{f}}_{\text{dist}} \oplus E^{\mathsf{b}}_{\text{dist}}$, one can prepend $E^{\mathsf{f}}_{\text{key}}$ and $E^{\mathsf{b}}_{\text{key}}$ and attempt key recovery on these parts, provided the total key material needed to evaluate the input mask parities to $E^{\mathsf{f}}_{\text{dist}}$ and $E^{\mathsf{b}}_{\text{dist}}$ remains manageable. Since linear cryptanalysis relies only on known plaintexts, the adversary does not need to control internal states in either branch. Thus, after prepending rounds, linear key recovery attacks on ZIP ciphers can be carried out similarly to attacks on standard iterative block ciphers. In the following sections, we identify suitable linear characteristics for ZIP-AES and ZIP-GIFT and demonstrate corresponding key recovery attacks.

4.1 Linear Key Recovery Attacks on ZIP-AES

In this section, we show a linear key recovery attack on (3,2)-rounds of ZIP-AES. We utilize Matsui's Algorithm 2 combined with the Fast Walsh-Hadamard Transform (FWHT) [CSQ07] for key recovery in the first round of both branches. For the backward branch, we recover the equivalent key K'_4, which is the inverse of the MixColumns operation applied

to the actual first round key of the backward branch K_4. Our attack requires $2^{123.7}$ known plaintexts to recover the whole master key in about $2^{123.7}$ encryptions and $2^{120.8}$ additions with a success probability of 80%. We note that the designers' analysis of ZIP-AES against linear attacks is limited to a discussion on bounds for linear characteristics.

Characteristic. To find our linear characteristic, we first use MiniZinc to find a cellwise linear pattern with a fixed number of active S-boxes (19) and a minimal number of guessed key cells when considering the key recovery using the equivalent key $K_4' = \mathsf{MC}^{-1}(K_4)$. Then, to find a concrete linear characteristic, we adapt the code from [ENP19] and heuristically found a solution that mostly uses the optimal S-box transitions with $c^2 = 2^{-6}$. We observe that there are no solutions using only optimal S-box transitions for this cellwise pattern.

Attack procedure. We base our attack on the observations given in [FGNP20] and extend its specifics to ZIP ciphers. We want to guess and recover the $|k| = 14 \cdot 8 = 112$ key bits from K_0 and K_4' marked in Figure 5. Since the corresponding cells are disjoint, we introduce 3 masks $\mathcal{K}_F, \mathcal{K}_B$ and $\mathcal{K}$ to indicate the relevant key recovery positions of K_0, K_4' and the union of both, respectively. Furthermore, let $R(x)$ and $R^{-1}(x)$ denote one round of forward AES without key addition and its inverse. Let $x|_{\mathcal{K}}$ and $x \uparrow_{\mathcal{K}}$ denote the gathering and scattering of bits according to the mask $\mathcal{K}$ satisfying $(x \uparrow_{\mathcal{K}})|_{\mathcal{K}} = x$.

1. Gather a set $\mathcal{D}$ of approximately $N = 2^{123.7}$ plaintext-ciphertext pairs and compute the distillation vector $\mathbf{w} \in \mathbb{Z}^{2^{|k|}}$ using $w_j = \sum_{(x,y)\in\mathcal{D}, x|_{\mathcal{K}}=j} (-1)^{\Gamma^{\oplus}\cdot y}$.
2. Split the integer j into two parts $j = j_F || j_B$ for the key recovery cells of K_0 and K_4', respectively. Compute the vector $\mathbf{v} \in \mathbb{Z}^{2^{|k|}}$ using $v_j = (-1)^{\Gamma^{\mathrm{f}}\cdot R(j_F \uparrow_{\mathcal{K}_F}) \,\oplus\, \Gamma^{\mathrm{b}}\cdot R^{-1}(j_B \uparrow_{\mathcal{K}_B})}$.
3. Let $\mathbf{H}$ denote the $2^{|k|}$-dimensional Walsh matrix and compute $\mathbf{q}$ using the FWHT: $\mathbf{q} = 2^{-|k|} \cdot \mathbf{H} \cdot \mathrm{diag}(\mathbf{H} \cdot \mathbf{v}) \cdot \mathbf{H} \cdot \mathbf{w}$. Recover the partial key k by selecting the biggest absolute entry $|q_k|$ in $\mathbf{q}$.

Data Complexity. Our attack uses a linear characteristic of correlation $c \approx 2^{-58.15}$, which is close to optimal. Based on the formulas by Selçuk [Sel08], we set our advantage $a = 112$ and success probability to 80% and find $N = \left(\Phi^{-1}(\mathbb{P}_S) + \Phi^{-1}(1 - 2^{-a-1})\right)^2 \cdot c^{-2} \approx 2^{123.7}$.

Runtime Evaluation. Our attack requires evaluating the output mask for every ciphertext (i.e., N times) in the distillation phase. The cost of computing the vector $\mathbf{q}$ is approximately given by $2^{|k|}$ multiplication of k-bit integers, which we can regard as $|k|2^{|k|}$ additions and 3 matrix-vector products using the FWHT, each requiring $|k|2^{|k|}$ additions. Thus,

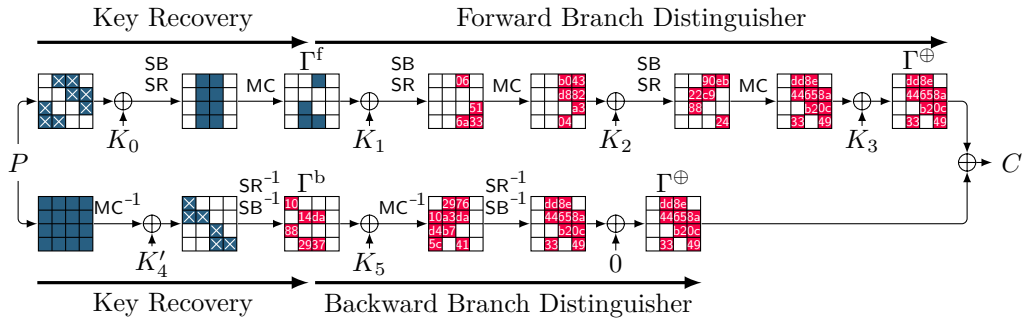

Figure 5: Linear attack on ZIP-AES-(3,2) with a (2,1)-round linear characteristic with 2 total key recovery rounds. ■ key recovery, ⊠ key guess, ■ linear trail with $c^2 = 2^{-116.3}$.

this step requires approximately $4 \cdot |k| 2^{|k|} \approx 2^{120.8}$ additions. Finally, brute-forcing the remaining 2^{64} key bits of K_0 requires 2^{64} trial encryptions. In total, our attack requires approximately $2^{123.7}$ encryptions and $2^{120.8}$ additions with a success probability of 80%.

4.2 Linear Key Recovery Attack on ZIP-GIFT

In this section, we show a $(10, 9)$-round linear key recovery attack on ZIP-GIFT. For this, we find a $(6, 6)$-round linear characteristic $((\Gamma^{\mathrm{f}}, \Gamma^{\mathrm{b}}) \to \Gamma^{\oplus})$ of correlation $c = 2^{-31}$ with

$$\Gamma^{\mathrm{f}} = \texttt{0000 0c0c 0000 0000}, \quad \Gamma^{\mathrm{b}} = \texttt{0040 0020 0000 8000}, \quad \Gamma^{\oplus} = \texttt{a0a0 0000 a0a0 0000}.$$

Notably, the best linear key recovery on GIFT-64 extends a 12-round linear trail to 19 rounds [SWW21b]. This indicates that our linear attack on ZIP-GIFT-(10,9) matches the number of rounds of the best known linear attack on GIFT-64.

Characteristic. To find the above characteristic, we use MiniZinc to model linear characteristic with fixed correlation and a minimal number of guessed key bits for key recovery.

We exploit the $(6, 6)$-round linear characteristic in a key recovery attack on ZIP-GIFT-$(10, 9)$ by prepending 4 and 3 rounds to the distinguisher as depicted in Figure 6.

We use the partial-sum technique to perform a $(10, 9)$-round key recovery attack on ZIP-GIFT. Given a collection of N plaintext-ciphertext pairs, we analyze for how often our linear approximation $\Gamma^{\mathrm{f}} \cdot X_4 \oplus \Gamma^{\mathrm{b}} \cdot X_{-3} \oplus \Gamma^{\oplus} \cdot C = 0$ holds for each key guess. Table 2 on page 28 shows the time and memory complexity calculation for each step of the partial-sum technique. The complete attack follows these steps:

0. Initialize a list $\mathcal{L}_0$ of size 2^{60} with all zeroes. For each of the N plaintext-ciphertext pairs, and for each possible 62-bit subkey value

$$K_0[0..31] \,\|\, K_1[0..15] \,\|\, K_2[0..3, 16..19] \,\|\, EK_{-1}[10, 11, 18, 19, 26, 27]$$

we compute the value of

$$z_0 = EX_0[0..19, 24..35, 40..51, 56..63] \,\|\, EX_{-1}[17..19, 24, 25, 27, 28] \,\|\, S_0,$$

where $S_0 = \Gamma^{\mathrm{f}} \cdot X_4 \oplus \Gamma^{\oplus} \cdot C \oplus Y_{-2}[29] \oplus Y_{-2}[30]$ and update $\mathcal{L}_0[z_0]$ with $\mathcal{L}_0[z_0] + 1$. From the algebraic normal form (ANF) of the inverse S-box, we find that $Y_{-2}[29]$ and $Y_{-2}[30]$ linearly influence $Y_{-3}[54]$, which then linearly influences $X_{-3}[54]$. Then, $Y_{-2}[29]$ and $Y_{-2}[30]$ can be merged with S and do not need to be stored separately. In this paper, we use the common simplified assumption that we can scale one partial sum step according to the number of S-boxes it performs. While this is not completely representative as it neglects the cost of the memory lookup, it is commonly used to allow better comparison. This step performs 47 S-box operations compared to the 304 of a $(10, 9)$-round encryption. As every repetition of this step costs $47/304 \approx 2^{-2.7}$ encryptions, the total complexity is about $N \cdot 2^{62} \cdot 2^{-2.7}$.

1. Initialize a list $\mathcal{L}_1$ for each of 2^{57} values of

$$z_1 = EX_0[0..3, 8..19, 24..35, 40..51, 56..63] \,\|\, EX_{-1}[16..19, 24..27] \,\|\, S_1,$$

where $S_1 = S_0 \oplus Y_{-3}[54]$. For each z_1 and each possible 2-bit subkey $EK[2, 3]$, compute z_2 to update $\mathcal{L}_1[z_1]$ as $\mathcal{L}_1[z_1] + \mathcal{L}_0[z_0]$. Similarly, the time complexity of this step is $2^{62} \cdot 2^2 \cdot 2^{60} \cdot 2^{-7.2}$ to access to a table with 2^{57} elements.

2–16. In each of these steps, we continue to guess a subset of the relevant subkeys in $E^{\mathrm{b}}_{\mathrm{key}}$, compute the corresponding intermediate steps, and then the counter corresponding to the relevant intermediate state bits will be stored in a list. See Table 2 on page 28.

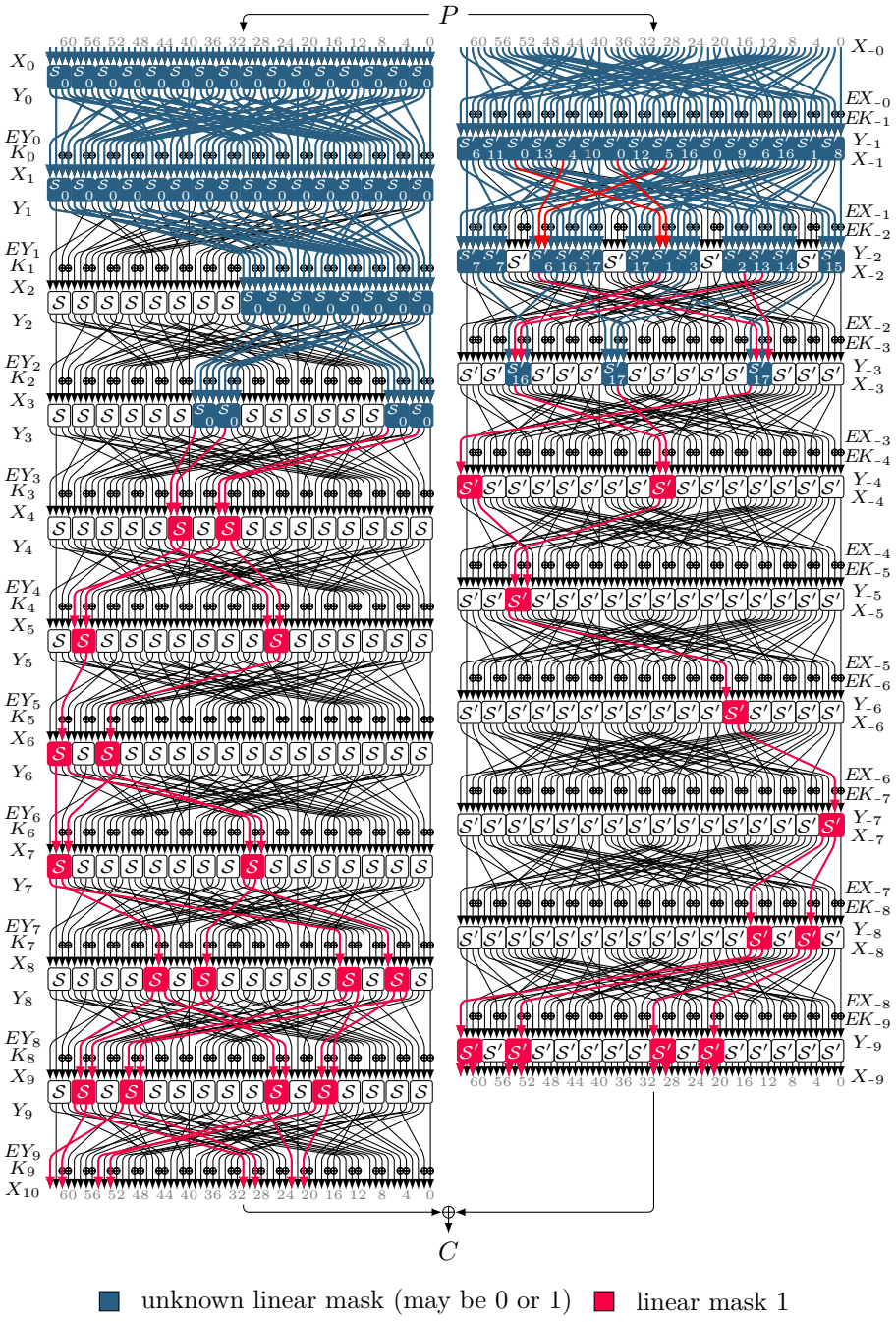

Figure 6: Linear attack on $(10, 9)$ rounds of ZIP-GIFT where the key recovery is performed on the first 4 rounds of the forward and the first 3 rounds of the backward branch

17. Initialize $\Sigma = 0$. For each 4-bit key $EK_{-2}[16, 17, 20, 21]$, compute $S' = S \oplus Y_{-2}[29] \oplus Y_{-2}[30] \oplus Y_{-2}[49] \oplus Y_{-2}[50] \oplus Y_{-3}[12] \oplus Y_{-3}[14] \oplus Y_{-3}[53] \oplus Y_{-3}[54]$. If $S' = 0$, update Σ as $\Sigma + \mathcal{L}_{16}[z_{16}]$, where $\mathcal{L}_{16}$ denotes the list constructed in Step 16 for each z_{16}. The complexity of this step is about $2^{106} \times 2^{13} \times 2^{-7.2}$.

18. We set the threshold for key recovery as $\theta = \sqrt{1/N} \cdot \Phi^{-1}(1 - 2^{-a-1})$ where a is the advantage of the attack, and $\Phi(\cdot)$ signifies the cumulative distribution function of the standard normal distribution. The key guess will be considered a candidate if the counter Σ satisfies $|\Sigma/N - 1/2| > \theta$. All keys compatible with the 106 guessed key bits are tested exhaustively.

Complexity Analysis. Let $\mathbb{P}_S$ be the success probability that a linear attack, with a linear characteristic of correlation c, with N known plaintext blocks, delivers an a-bit advantage (the proportion of keys survives after the subkey enumeration phase equals 2^{-a}). Using the method discussed in [Sel08, BN17], if we set $\theta = \sqrt{1/N} \cdot \Phi^{-1}(1-2^{-a-1})$, $a = 2$, and $N = 2^{63}$ then, the success probability is $\mathbb{P}_S \approx \Phi\left(c \cdot \sqrt{N} - \Phi^{-1}(1 - 2^{-a-1}) \cdot \sqrt{1 + N \cdot 2^{-n}}\right) \approx 60\%$, where $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. The time complexity of Step 0 to Step 17 is about $2^{122.48}$ (see Table 2) and Step 18 needs about $2^{128} \cdot 2^{-a} \cdot (1 + 2^{-64})$ $(10, 9)$-round of encryptions. Overall, our $(10, 9)$-round linear key recovery attack needs 2^{126} time and $2^{60.18}$ memory for a $\approx 60\%$ success probability.

5 Integral Cryptanalysis on ZIP Ciphers

In this section, we focus on integral cryptanalysis of two ZIP ciphers, namely ZIP-AES and ZIP-GIFT. It was observed in [FGGL+24] that the resistance of a sum of two keyed permutations against integral attacks is essentially determined by the stronger of the two components. Recall that if a cipher has algebraic degree d, it becomes susceptible to an integral attack over any linear subspace of dimension $d + 1$. Thus, for a ZIP cipher $E^{\oplus} = E^{\mathrm{f}} \oplus E^{\mathrm{b}}$, where E^{f} and E^{b} have degrees d^{f} and d^{b} with $d^{\mathrm{f}} > d^{\mathrm{b}}$, one can mount an integral attack on $E^{\oplus}$ by considering a subspace of dimension $d^{\mathrm{f}} + 1$.

ZIP ciphers complicate integral key recovery as rounds cannot be appended after the distinguisher and must be prepended instead. As observed in [FGGL+24], the two integral distinguishers seldom permit plaintext sets that fulfil the conditions after some key recovery rounds. We give distinguishers and key recovery attacks for ZIP-AES and ZIP-GIFT using new key recovery ideas and discuss why further round extensions are hard.

5.1 Integral Key Recovery Attacks on ZIP-AES

In this section, we present an integral key recovery attack on (3,3)-round ZIP-AES. See also Appendix A for an attack on (4,2)-round ZIP-AES, which might be more transferable to other applications. We adopt the standard terminology for integral attacks and use some notation from [KR07], i.e., a Λ-set consists of 2^n elements, where a cell is *active* if it assumes all possible values 2^n times, *constant* if it takes the same value across all states, and *balanced* if the Xor-sum over all states equals zero. Additionally, we use A_j for an active group: in an active group of c 8-bit cells, the concatenation of the c member cells takes all 2^{8c} values. We also write c_{ij} for the cell in row i and column j of an AES state.

We utilize the well-known 3-round forward branch zero-sum distinguisher and combine it with a 3-round backward branch zero-sum distinguisher. Figure 7 illustrates both zero-sum distinguishers separately. The first round of the backward branch performs key recovery by exploiting two zero differences between two cells, recovering the equivalent key K'_4. The following attack recovers 16 bits of key material in 2^{40} oracle encryptions.

Attack Procedure.

1. Let Λ^{f} be the set of 2^8 AES states for which the cell c_{01} is active and all others are constant. Let Λ_I denote the set of 2^{16} AES states where cells c_{02} and c_{12} are active and equal, cells c_{20} and c_{30} are active and equal, and all remaining cells are constant.

2. For $K'_4 = \mathsf{MC}^{-1}(K_4)$, we guess the differences $\Delta_x = c_{02} \oplus c_{12}$ and $\Delta_y = c_{20} \oplus c_{30}$. For all 2^{16} differences Δ_x and Δ_y do the following: Let K_g denote the 16-byte representation of the guess for K'_4 where the cell c_{20} equals Δ_x and cell c_{02} equals Δ_y. Let the other cells be 0. Build the set Λ^{b} as follows: $\Lambda^{\mathrm{b}} = \{\mathsf{MC}(x \oplus K_g) : x \in \Lambda_I\}$ For every pairwise combination of Λ^{f} and Λ^{b}, query the ciphertext sum:

$$S = \bigoplus_{P_1 \in \Lambda^{\mathrm{f}}} \bigoplus_{P_2 \in \Lambda^{\mathrm{b}}} \text{ZIP-AES}_{3,3}(P_1 \oplus P_2).$$

   If $S = 0$, Δ_x and Δ_y correspond to the relevant differences of K'_4.

Key-Dependent Behavior. Let us consider the forward and backward branches separately: For the forward branch, we see that $\bigoplus_{x \in \Lambda^{\mathrm{f}}} \mathrm{AES}_3(x) = 0$ as shown in Figure 7. For the backward branch, suppose we guess the correct differences Δ_x and Δ_y, then the encryption of the set Λ^{f} under the encryption oracle will produce in the first round the set Λ_I for which $c_{02} = c_{12}$ and $c_{20} = c_{30}$. After the application of inverse ShiftRows and inverse SubBytes, we get states with $c_{02} = c_{13}$ and $c_{22} = c_{33}$. Furthermore, for each column, the active bytes form an active group, where each of the 2^{16} values appears exactly once. After MixColumns and ShiftRows, we find that each column forms an active group with 2 active cells again. This can be verified by writing down the linear mapping for $\mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}$ under the assumption that $c_{02} = c_{12}$ and $c_{20} = c_{30}$ and observing that the submatrix for each active input/output group is invertible. Vice versa, for a wrong guess of Δ_x and Δ_y, we will not receive the set Λ_I in the first round. Thus, the S-Boxes in the first round will produce key-dependent non-zero differences $c_{02} \oplus c_{13}$ and $c_{22} \oplus c_{33}$ which differ across the 2^{16} states. This destroys the assumption for the analysis of $\mathsf{SR}^{-1} \circ \mathsf{MC}^{-1}$, thus destroying the active groups after MC^{-1}. Hence, we will not receive a zero-sum for a wrong key guess. We verified the behavior of both branches experimentally.

When combining both branches it is straightforward to see that we will get a zero-sum for the correct key guess; however, we must be careful to not accidentally receive zero-sums for wrong key guesses. Therefore, we ensure that the active cells of Λ^{f} and Λ^{b}

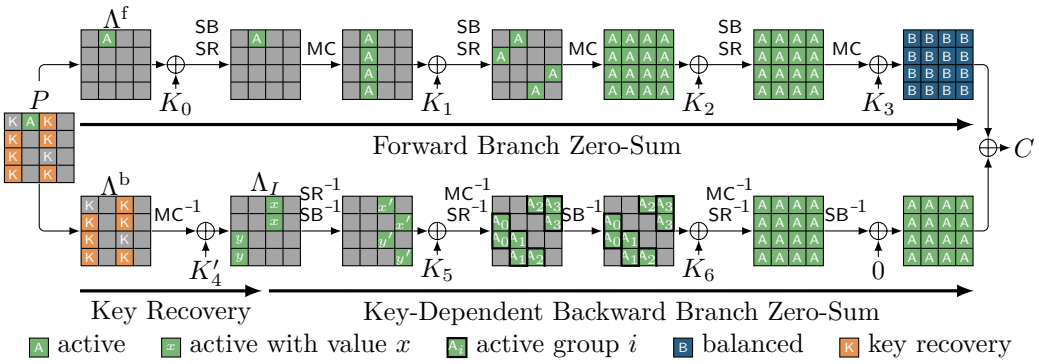

Figure 7: Integral attack on ZIP-AES-(3,3) with 1 key recovery round. Active cells take all possible values across all states, active groups take all possible values for the whole group, other cells are constant, and the XOR-sum for balanced cells is 0. Active/key recovery cells in Λ^{f} and Λ^{b} must not overlap. P is formed by the union of Λ^{f} and Λ^{b}.

do not overlap, otherwise, the active cells of Λ^{f} might destroy the key dependency of Λ^{b}. Furthermore, no cells between $\mathsf{MC}(K_g)$ must overlap with the active cells of Λ^{f}. If this were the case, we would observe a zero-sum for wrong key guesses as well, preventing a key recovery attack. To exclude other issues, we experimentally verify the attack and observe a zero-sum only for the correct key guess.

Complexity Evaluation. Our attack guesses 2^{16} key differences and, for every guess, encrypts the set $\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ of size 2^{24}. Hence, we need $2^{16} \cdot 2^{24} = 2^{40}$ oracle encryptions to recover 16 key bits, leaving a brute force step of 2^{112}. Note that we can repeat the attack up to 6 times by shifting our zero-sum to recover up to 96 key bits. We propose a trade-off of repeating the attack 5 times using $2^{42.3}$ data and 2^{48} key guesses. On average, we only need $2^{41.3}$ data, as we can stop gathering data once we observe a zero-sum for a key guess.

Difficulty of Higher Round Attacks. Flórez-Gutiérrez et al. showed a distinguishing attack on ZIP-AES-(4,4)-rounds [FGGL+24]. However, based on our analysis, we believe that integral key recovery attacks for (4,3)-round or (4,4)-round are unlikely. Due to the ZIP structure, we need to ensure that no active or key-dependent cells overlap between Λ^{f} and Λ^{b}. Suppose two such cells would overlap, then querying the combined set $\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ would cancel our key guess. For 4+ round distinguishers in the forward branch, at least one diagonal is active or key-dependent, and for 3+ round distinguishers in the backward branch, at least one column is active or key-dependent. Combining such distinguishers results in at least one overlapping cell, leading to a zero-sum for incorrect key guesses. Note that we do not claim a key recovery attack on ZIP-AES-(4,4)-rounds is impossible, but we are not aware of any attack strategy to overcome the above issue.

5.2 Integral Key Recovery Attacks on ZIP-GIFT

In this section, we present an integral key recovery attack on ZIP-GIFT. For this, we search for integral distinguishers using the monomial-prediction framework in an automated MILP search. In our search, we fix a single plaintext bit to a constant and mark all other plaintext bits as active. This way we find integral distinguishers for up to $(9, 9)$-rounds of ZIP-GIFT. Let $\mathcal{P}$ denote the set of 2^{63} 64-bit vectors with bit 60 fixed, then

$$\bigoplus_{x \in \mathcal{P}} \text{ZIP-GIFT}_{9,9}(x)[i] = 0, \ \forall i \in \{8, 16, 40, 48, 60\}.$$

5.2.1 Difficulty of Higher-Round Integral Key Recovery Attacks on ZIP-GIFT

For normal block ciphers, an integral distinguisher is usually turned into a key recovery by appending rounds and guessing partial keys to decrypt the ciphertext. This strategy fails for ZIP-GIFT as its output is the Xor of two branches. Instead, one might prepend rounds, which is impractical for 2 reasons: the branches use different round functions, so it is hard to pick a common plaintext set whose states—after the respective prepended rounds and key guesses—simultaneously meet the distinguisher's input sets in both branches; and the branches are keyed differently which makes it more difficult from extending the distinguisher by adding rounds on either side.

Suppose we find an integral distinguisher for ZIP-GIFT-$(r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}})$ with $\mathcal{I}^{\mathrm{f}}, \mathcal{I}^{\mathrm{b}}$ denoting the active inputs bits and Λ^{f}, Λ^{b} denoting the corresponding value sets. To prepend key recovery rounds, we first determine the common active bits in the plaintext, where we consider all input bits of an (inverse) S-box active if any output bits are active. Let $\mathcal{I}$ denote the set of common active plaintext bits, and let $\mathcal{P}$ be the corresponding plaintext set. If there are too many active input bits, then the plaintext set $\mathcal{P}$ will include all plaintexts, preventing a valid attack. Assume such a suitable plaintext set $\mathcal{P}$ exists, then

querying $\mathcal{P}$ would lead to a zero-sum, however, partitioning the space for key recovery would work: if we guess all involved key bits in one branch, and partition the plaintext set accordingly, we get randomized inputs for the other branch's distinguisher. Fundamentally, performing 2 chosen plaintext attacks simultaneously is only possible in special cases. To overcome this difficulty, we show a dedicated attack exploiting the GIFT S-box.

The $(9, 9)$-round integral distinguisher for ZIP-GIFT has data complexity 2^{63}, with $|\mathcal{I}^{\mathrm{f}}| = |\mathcal{I}^{\mathrm{b}}| = 63$. If we prepend a round to any branch, the common plaintext set becomes the full 2^{64} input space, which always leads to a zero-sum, eliminating any key-dependency.

5.2.2 Integral Key Recovery Attack on ZIP-GIFT-(8, 8)

In this section, we present an integral attack on ZIP-GIFT-(8,8) based on a 7-round and an 8-round distinguisher as depicted in Figure 8. For comparison, the designers of GIFT [BPP+17] use a 9-round integral distinguisher for a 14-round integral key recovery attack. We use a 7-round backward branch zero-sum distinguisher, prepend one key recovery round, and combine it with an 8-round forward branch zero-sum distinguisher. We utilize

$$\bigoplus_{x \in \Lambda^{\mathrm{f}}_{\mathrm{dist}}} \mathrm{GIFT}_8(x)[i] = \bigoplus_{x \in \Lambda^{\mathrm{b}}_{\mathrm{dist}}} \mathrm{GIFT}_7^{-1}(x)[i] = 0 \qquad \forall i \in \mathcal{I}^{\oplus},$$

where $\Lambda^{\mathrm{f}}_{\mathrm{dist}}$ and $\Lambda^{\mathrm{b}}_{\mathrm{dist}}$ denote the sets of 2^{61} 64-bit vectors with bits 0, 17, 34 and bits 1, 2, 3 fixed, respectively. The set of balanced bits at the output is $\mathcal{I}^{\oplus} = \{0, 4, 8, 12, 16, 20, 24, 28, 40, 44, 48, 56\}$. More precisely, we first use MiniZinc to identify the set of balanced bits after 8 rounds of GIFT for the plaintext structure $\Lambda^{\mathrm{f}}_{\mathrm{dist}}$, and then compute the set of balanced bits after 7 rounds of GIFT^{-1} for the plaintext structure $\Lambda^{\mathrm{b}}_{\mathrm{dist}}$. The intersection of these two sets yields the index set $\mathcal{I}^{\oplus}$. Our $(8, 8)$-round key recovery attack proceeds as follows:

1. Let $\Lambda^{\mathrm{b}} = \{e \in \mathbb{F}_2^{64} \mid e[3..0] = \mathtt{001*}\}$ with $|\Lambda^{\mathrm{b}}| = 2^{61}$. Moreover, we note a property of inverse GIFT S-box $\mathcal{S}'$:

   $$\mathcal{S}'(x) = \{2, 3\} \text{ if } x \in \{4, \mathtt{c}\} \quad \text{ or in binary } \quad \mathcal{S}'(\mathtt{*100}) = \mathtt{001*}.$$

   Combining these properties, we find a well-structured cube of dimension 61 before the first application of the inverse S-box layer $\mathsf{SubCells}^{-1}$:

   $$(\mathsf{SubCells}^{-1})^{-1}(\Lambda^{\mathrm{b}}) = \{e \in \mathbb{F}_2^{64} \mid e[3..0] = \mathtt{*100}\}.$$

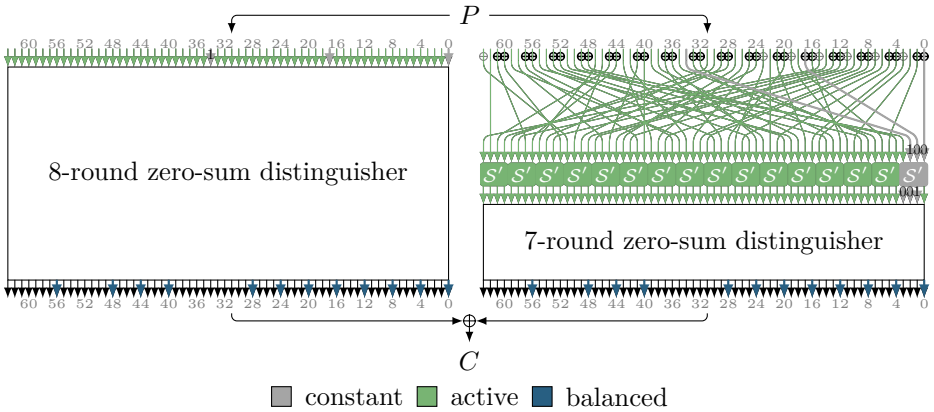

Figure 8: Integral key recovery attack on ZIP-GIFT-(8, 8) with one key recovery round. For the backwards branch, the activity pattern for the correct key guess is shown.

2. Guess bits 0 and 17 of K_{-1} and store in K_g. For each of the 4 guesses, decrypt each element of Λ^{b} by one round to construct the plaintext set and query the output sum:

$$\mathcal{P} = \left\{P_{64}(\mathsf{SubCells}(x)) \oplus K_g | x \in \Lambda^{\mathrm{b}}\right\}, \qquad S = \bigoplus_{x\in\mathcal{P}} \text{ZIP-GIFT}_{8,8}(x).$$

3. If $S[i] = 0$ for the 12 bits $i \in \mathcal{I}^{\oplus}$, brute force the remaining 126 key bits.

Key-Dependent Behavior. From the attack procedure, we see that in the backward branch, for the correct key guess, $\{R^{-1}(x) \,|\, x \in \mathcal{P}\} = \Lambda^{\mathrm{b}}$ should hold. Hence, we observe the zero-sum distinguisher in the backward branch. For the forward branch, we also observe the zero-sum distinguisher as the active input bits nicely overlap (see Figure 8). Conversely, for wrong key guesses, the set $\{R^{-1}(x) \mid x \in \mathcal{P}\} \neq \Lambda^{\mathrm{b}}$ behaves randomly for the rightmost S-box and does not exhibit any cube structure. However, it will still form an affine space which could lead to a zero-sum. To analyze this possibility, we use monomial prediction. Since we cannot directly formulate an affine space, we analyze the function $R^{-7} \circ L$, where L denotes an invertible linear function that operates on the 4 rightmost bits of the state. We choose one value for L for each of the three wrong key guesses such that L^{-1} transforms the affine space into a cube of the form `CCC*`. By analyzing the input cube for the different functions $R^{-7} \circ L$, we did not observe the exact same set of balanced bit positions as in $\mathcal{I}^{\oplus}$. In particular, for the other three invertible linear mappings L, and for each wrong key guess such that L^{-1} transforms the affine space into a cube of the form `*CCC`, `C*CC`, or `CC*C` at the input of $R^{-7} \circ L$, we can expect that the balanced bit positions will not coincide exactly with those in $\mathcal{I}^{\oplus}$. While this does not prove that there are no zero-sum properties, since an even number of monomial trails could exist, we still consider it evidence for the wrong-key randomization hypothesis. Consequently, we assume the sum of ciphertexts is uniformly distributed. Since we filter 12 balanced bits at the ciphertext and only 2 key bits are guessed, we expect only a negligible effect on success probability.

Complexity Analysis. For each of the 2^2 key guesses, we query 2^{61} plaintexts leading to a total data complexity of at most 2^{63}. The time complexity is dominated by the brute force search, which needs at most 2^{126} encryptions.

6 Discussion of Key Recovery Attacks

In this section, we summarize our main findings for key recovery attacks on ZIP ciphers.

Differential Attacks. For differential key recovery, we find that extending a differential characteristic or distinguisher by some rounds of truncated, i.e., deterministic, propagation works well. This way, we can attack more rounds than by just using the differential characteristic as suggested by [FGGL+24, Prop. 2]. To perform key recovery, we find it best to do a 0-round attack, recovering the key using the first few rounds of the distinguisher. For such a key recovery attack, it is crucial that we carefully analyze the set of ciphertext differences to get a strong ciphertext filter and that we carefully analyze the key recovery procedure, as we have found both them to be important factors for the total time complexity. For ZIP-GIFT, we found that there is only one input difference that works for both branches at the same time, while for ZIP-AES, we used a set of 2^{24} plaintext differences to amplify the data gathering phase. In both cases, we perform key recovery by assuming that a pair that survives the ciphertext filter follows our differential distinguisher, and then we enumerate all keys that make the plaintext pair compatible with the distinguisher. Since each surviving pair leads to many key candidates, we found it crucial to have a differential distinguisher with strong ciphertext filtering, i.e., we want

to discard as many pairs as possible. Hence, for ZIP-AES, we include the ciphertext filter in our optimization goal for the MiniZinc program, while for ZIP-GIFT, we heuristically optimize for a strong ciphertext filter using the undisturbed bits of the S-box and then perform a detailed analysis afterwards using the HyperLogLog algorithm.

Linear Attacks. We find linear key recovery to be the most straightforward one. This is because linear cryptanalysis is a known-plaintext attack, and we do not need to worry about performing two or more chosen-plaintext attacks at the same time. To perform key recovery, we find that the existing optimizations of using the Fast-Walsh-Hadamard-Transform (FWHT) and the partial-sum technique work well, as we apply them to ZIP-AES and ZIP-GIFT, respectively. For ZIP-GIFT, this leads to our best attack of $(10, 9)$ rounds.

Integral Attacks. While integral cryptanalysis seems to give the best distinguishers, key recovery is particularly difficult. This is highly interesting for ZIP ciphers in low-latency scenarios. Concretely, it appears that an integral distinguishing attack can often cover more rounds than an integral key recovery attack for ZIP-ciphers. We find that for a key recovery attack, we need to find distinguishers for both branches that combine well and permit a key dependency. For ZIP-GIFT, we used a key-dependent integral transition of the S-box, while for ZIP-AES, we found a distinguisher based on a conditional affine space.

Differential-Linear Attacks. For differential-linear attacks, the strategy outlined by the designers of ZIP-AES seems most promising [FGGL+24]. To attack a ZIP cipher, they propose to use a differential-linear distinguisher in one of the two branches and to only use a linear distinguisher in the other branch. This way, only one branch needs chosen plaintexts, and some key recovery rounds can be prepended in the other branch.

Impossible-Differential Attacks While one can find impossible-differential distinguishers for ZIP ciphers, in our case they are very short (e.g., ZIP-GIFT-3,2). As we did not see a good way to add rounds for key recovery, we deem this attack class to be not competitive with normal differential attacks.

Rectangle Attacks. While boomerang attacks are not possible as there is no decryption for ZIP ciphers, Rectangle attacks are possible in principle. In a rectangle attack on an n-bit block cipher, one probabilistically expects the correct difference between two pairs with probability 2^{-n}. For a ZIP cipher, the state size is doubled so this would only work with probability 2^{-2n}, likely leading to not competitive attacks.

Applicability to Orthros and Other ZIP Ciphers. We believe our techniques for key recovery can also be applied to other ZIP ciphers or sums-of-permutations, like Orthros. In particular, the similarity between ZIP-GIFT and Orthros suggests that the techniques are transferable. Transferring the integral and linear key recovery attacks would be of high interest as this seems to be a gap in the literature. Furthermore, applying truncated differential cryptanalysis to Orthros could lead to improved results. Similarly, our key recovery attacks could be applied to other wide-trail-SPN-based ZIP cipher, but only ZIP-AES has been proposed so far in this category.

7 Conclusion

We have presented the first differential, linear, and integral key recovery attacks on ZIP-AES and ZIP-GIFT. We discuss challenges that arise when applying existing key recovery techniques and how to overcome them. For differential attacks, we discuss how to add

rounds after a differential characteristic and have shown how to optimize the key recovery such that the attack is possible despite suboptimal signal-to-noise ratio. For linear attacks, we show how the existing optimization techniques for improving time complexity can be applied. For integral attacks, we show new techniques to perform a key recovery, as existing techniques are not applicable. This deepens our understanding of the cryptanalysis of ZIP ciphers and allows us to better estimate their security margin. We believe the security of ZIP ciphers could be further evaluated in future work. In particular, key recovery based on cube attacks with superpoly recovery seems promising.

Acknowledgments

We would like to thank Simon Gerhalter for insightful discussions. Debasmita Chakraborty was a postdoctoral researcher at Graz University of Technology while carrying out this work. This paper was co-funded by the European Union (ERC Starting Grant KEYLESS, #101165216). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[BDD+24] Christina Boura, Nicolas David, Patrick Derbez, Rachelle Heim Boissier, and María Naya-Plasencia. A generic algorithm for efficient key recovery in differential attacks - and its associated tool. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024*, volume 14651 of *LNCS*, pages 217–248. Springer, 2024. doi:10.1007/978-3-031-58716-0_8.

[BI99] Mihir Bellare and Russell Impagliazzo. A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to PRP to PRF conversion. IACR Cryptology ePrint Archive, Paper 1999/024, 1999. URL: http://eprint.iacr.org/1999/024.

[BIL+21] Subhadeep Banik, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, and Kosei Sakamoto. Orthros: A low-latency PRF. *IACR Transactions on Symmetric Cryptology*, 2021(1):37–77, 2021. doi:10.46586/tosc.v2021.i1.37-77.

[BKR98] Mihir Bellare, Ted Krovetz, and Phillip Rogaway. Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. In Kaisa Nyberg, editor, *EUROCRYPT '98*, volume 1403 of *LNCS*, pages 266–280. Springer, 1998. doi:10.1007/bfb0054132.

[BN17] Céline Blondeau and Kaisa Nyberg. Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. *Designs, Codes and Cryptography*, 82(1-2):319–349, 2017. doi:10.1007/s10623-016-0268-6.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small Present – towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, 2017. doi:10.1007/978-3-319-66787-4_16.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991. doi:10.1007/bf00630563.

[CSQ07] Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the time complexity of Matsui's linear cryptanalysis. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC 2007*, volume 4817 of *LNCS*, pages 77–88. Springer, 2007. doi:10.1007/978-3-540-76788-6_7.

[Din24] Itai Dinur. Tight indistinguishability bounds for the XOR of independent random permutations by Fourier analysis. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024*, volume 14651 of *LNCS*, pages 33–62. Springer, 2024. doi:10.1007/978-3-031-58716-0_2.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997. doi:10.1007/bfb0052343.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. doi:10.1007/978-3-662-04722-4.

[ENP19] Maria Eichlseder, Marcel Nageler, and Robert Primas. Analyzing the linear keystream biases in AEGIS. *IACR Transactions on Symmetric Cryptology*, 2019(4):348–368, 2019. doi:10.13154/tosc.v2019.i4.348-368.

[FFGM07] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics and Theoretical Computer Science*, AOFA:137–156, 2007. doi:10.46298/dmtcs.3545.

[FGGL+24] Antonio Flórez-Gutiérrez, Lorenzo Grassi, Gregor Leander, Ferdinand Sibleyras, and Yosuke Todo. General practical cryptanalysis of the sum of round-reduced block ciphers and ZIP-AES. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024*, volume 15492 of *LNCS*, pages 280–311. Springer, 2024. doi:10.1007/978-981-96-0947-5_10.

[FGNP20] Antonio Flórez-Gutiérrez and María Naya-Plasencia. Improving key-recovery in linear attacks: Application to 28-round PRESENT. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 221–249. Springer, 2020. doi:10.1007/978-3-030-45721-1_9.

[FKL+00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David A. Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 213–230. Springer, 2000. doi:10.1007/3-540-44706-7_15.

[HNH13] Stefan Heule, Marc Nunkesser, and Alexander Hall. HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In Giovanna Guerrini and Norman W. Paton, editors, *EDBT '13*, pages 683–692. ACM, 2013. doi:10.1145/2452376.2452456.

[HSWW20] Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 446–476. Springer, 2020. doi:10.1007/978-3-030-64837-4_15.

[HTR+25] Hosein Hadipour, Yosuke Todo, Mostafizar Rahman, Maria Eichlseder, Ravi Anand, and Takanori Isobe. Turning multiple key-dependent attacks into universal attacks. IACR Cryptology ePrint Archive, Paper 2025/1996, 2025. URL: https://eprint.iacr.org/2025/1996.

[HWW+24] Shiqi Hou, Baofeng Wu, Shichang Wang, Hao Guo, and Dongdai Lin. Truncated differential attacks on symmetric primitives with linear key schedule: WARP and Orthros. *The Computer Journal*, 67(4):1483–1500, 2024. doi:10.1093/comjnl/bxad075.

[KMNP10] Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 130–145. Springer, 2010. doi:10.1007/978-3-642-17373-8_8.

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007. doi:10.1007/978-3-540-76900-2_19.

[Lai94] Xuejia Lai. *Communications and Cryptography: Two Sides of One Tapestry*, chapter Higher Order Derivatives and Differential Cryptanalysis, pages 227–233. Springer, 1994. doi:10.1007/978-1-4615-2694-0_23.

[LSW22] Muzhou Li, Ling Sun, and Meiqin Wang. Automated key recovery attacks on round-reduced Orthros. In Lejla Batina and Joan Daemen, editors, *AFRICACRYPT 2022*, volume 13503 of *LNCS*, pages 189–213. Springer Nature Switzerland, 2022. doi:10.1007/978-3-031-17433-9_9.

[Luc00] Stefan Lucks. The sum of PRPs is a secure PRF. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 470–484. Springer, 2000. doi:10.1007/3-540-45539-6_34.

[LWZZ19] Lingchen Li, Wenling Wu, Yafei Zheng, and Lei Zhang. The relationship between the construction and solution of the MILP models and applications. IACR Cryptology ePrint Archive, Paper 2019/049, 2019. URL: https://eprint.iacr.org/2019/049.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT '93*, volume 765 of *LNCS*, pages 386–397. Springer, 1993. doi:10.1007/3-540-48285-7_33.

[NGJE25] Marcel Nageler, Shibam Ghosh, Marlene Jüttler, and Maria Eichlseder. AutoDiVer: Automatically verifying differential characteristics and learning key conditions. *IACR Transactions on Symmetric Cryptology*, 2025(1):471–514, 2025. doi:10.46586/tosc.v2025.i1.471-514.

[Pat10] Jacques Patarin. Introduction to mirror theory: Analysis of systems of linear equalities and linear non equalities for cryptography. IACR Cryptology ePrint Archive, Paper 2010/287, 2010. URL: http://eprint.iacr.org/2010/287.

[PT22] Thomas Peyrin and Quan Quan Tan. Mind your path: On (key) dependencies in differential characteristics. *IACR Transactions on Symmetric Cryptology*, 2022(4):179–207, 2022. doi:10.46586/tosc.v2022.i4.179-207.

[QHS16] Kexin Qiao, Lei Hu, and Siwei Sun. Differential security evaluation of Simeck with dynamic key-guessing techniques. In Olivier Camp, Steven Furnell, and Paolo Mori, editors, *ICISSP 2016*, pages 74–84. SciTePress, 2016. doi:10.5220/0005684400740084.

[Sel08] Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008. doi:10.1007/s00145-007-9013-7.

[SWW21a] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Transactions on Symmetric Cryptology*, 2021(1):269–315, 2021. doi:10.46586/tosc.v2021.i1.269-315.

[SWW21b] Ling Sun, Wei Wang, and Meiqin Wang. Improved attacks on GIFT-64. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021*, volume 13203 of *LNCS*, pages 246–265. Springer, 2021. doi:10.1007/978-3-030-99277-4_12.

[TA14] Yosuke Todo and Kazumaro Aoki. FFT key recovery for integral attack. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 2014*, volume 8813 of *LNCS*, pages 64–81. Springer, 2014. doi:10.1007/978-3-319-12280-9_5.

[TISI23] Kazuma Taka, Tatsuya Ishikawa, Kosei Sakamoto, and Takanori Isobe. An efficient strategy to construct a better differential on multiple-branch-based designs: Application to Orthros. In Mike Rosulek, editor, *CT-RSA 2023*, volume 13871 of *LNCS*, pages 277–304. Springer, 2023. doi:10.1007/978-3-031-30872-7_11.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015. doi:10.1007/978-3-662-46800-5_12.

A Integral Attack on (4,2)-round ZIP-AES

In this section, we present an integral key recovery attack on (4,2)-rounds of ZIP-AES. We adopt the standard terminology for integral attacks, i.e., a Λ-set consists of 256 elements, where a cell is *active* if it assumes all possible values, *constant* if it takes the same value across all states, and *balanced* if the Xor-sum over all states equals zero. We utilize the well-known 3-round forward branch zero-sum distinguisher, prepend one round of key recovery, and combine it with a 2-round backward branch zero-sum distinguisher. Figure 9 shows both zero-sum distinguishers and the prepended key recovery round. The attack,

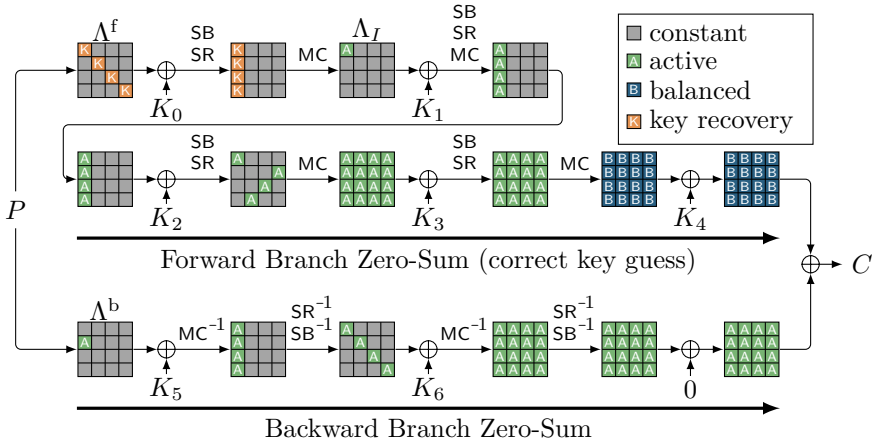

Figure 9: Integral attack on (4,2)-rounds of ZIP-AES with one key recovery round in the forward branch.

described in the following, recovers 32 bits of key material in 2^{48} oracle encryptions. With additional memory overhead, the attack can be extended to require only 2^{40} encryptions.

Attack Procedure.

1. Let Λ_I be a set of 256 AES states for which the first cell is active and all others are constant, and let Λ^{b} be a set of 256 AES states for which the first cell in the second row is active and all others are constant.
2. Let $\mathcal{K} = \{\mathrm{diag}(k_1, k_2, k_3, k_4) : k_i \in \{0, \ldots, 255\}\}$, i.e., the set of possible key states which take all possible values on the diagonal while all other cells are zero.
3. For every $K \in \mathcal{K}$ do as follows:
   (a) Decrypt each element of Λ_I backward by one round under the key K:
   $$\Lambda^{\mathrm{f}} = \left\{\mathrm{SB}^{-1}\left(\mathrm{SR}^{-1}\left(\mathrm{MC}^{-1}\left(x\right)\right)\right) \oplus K : x \in \Lambda_I\right\}.$$
   (b) For every $x \in \Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ (elementwise Xor of both sets), query the encryption oracle and build the cumulative Xor-sum
   $$S := \bigoplus_{x \in (\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}})} \text{ZIP-AES}_{4,2}(x).$$
   (c) If $S = 0$, then the diagonal key bytes of K match the master key K_0.

The above procedure recovers four key bytes. To recover the full key, we can repeat the procedure four times, each time selecting a different active cell in Λ_I, once for every cell in the first row. At the same time, we adjust $\mathcal{K}$ so that the cells that can take all possible values match the active cells in Λ^{f}. For Λ^{b}, we choose one active cell but need to ensure that this cell is not also active in Λ^{f}.

Key-Dependent Behavior. Let us consider the forward and backward branches separately: For the forward branch, suppose our key guess K matches the bytes in the master key K_0, then the encryption of the set Λ^{f} under the encryption oracle will produce after one round the set Λ_I and thus $\bigoplus_{x \in \Lambda^{\mathrm{f}}} \mathrm{AES}_4(x) = 0$. Vice versa, we experimentally verified for a substantial amount of wrong key guesses K that the set Λ^{f} produced by K leads to a non-zero Xor-sum $\bigoplus_{x \in \Lambda^{\mathrm{f}}} \mathrm{AES}_4(x) \neq 0$. In particular, for the wrong key guess already the whole first column of Λ_I is balanced and this property will be subsequently destroyed by the S-Boxes. For the backward branch, it is easy to see that $\bigoplus_{x \in \Lambda^{\mathrm{b}}} \mathrm{AES}_2^{-1}(x) = 0$, which can also be observed in Figure 9.

Furthermore, we can derive that

$$\begin{aligned} S :&= \bigoplus_{x \in (\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}})} \text{ZIP-AES}_{4,2}(x) \\ &= \bigoplus_{x \in (\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}})} \mathrm{AES}_4(x) \quad \oplus \quad \bigoplus_{x \in (\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}})} \mathrm{AES}_2^{-1}(x) \\ &= \bigoplus_{y \in \Lambda^{\mathrm{b}}} \underbrace{\bigoplus_{x \in \Lambda^{\mathrm{f}}} \mathrm{AES}_4(x \oplus y)}_{:=F} \quad \oplus \quad \bigoplus_{x \in \Lambda^{\mathrm{f}}} \underbrace{\bigoplus_{y \in \Lambda^{\mathrm{b}}} \mathrm{AES}_2^{-1}(y \oplus x)}_{:=B}. \end{aligned}$$

Since no active cells overlap for Λ^{f} and Λ^{b}, the addition of y in F and the addition of x in B only change the constant cells, which means that $F = 0$ for the correct key guess and $F \neq 0$ for the wrong key guess while $B = 0$ in both cases. Hence, if $S = 0$, the guessed key bytes of K match the master key K_0.

Runtime Evaluation. Our attack procedure recovers four key bytes by guessing all possible 2^{32} values and, for every such key guess, encrypts the combined set $\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ of size 2^{16}. Therefore, we require $2^{32} \cdot 2^{16} = 2^{48}$ oracle encryptions to recover 4 key bytes. Repeating the attack four times with different $\Lambda_I, \Lambda^{\mathrm{b}}, \mathcal{K}$ (as discussed above) then recovers the full key in $4 \cdot 2^{48} = 2^{50}$ attempts. We can also repeat the attack only 3 times and guess the remaining four key bytes by brute force, requiring only $3 \cdot 2^{48} + 2^{32} \approx 2^{49.58}$ attempts.

We can further improve the attack by using more memory. For all AES states S in which the cells along the main diagonal vary, we precompute the cumulative Xor-sum of the encryption of $S \oplus \Lambda^{\mathrm{b}}$ and store the result in a 2^{32} sized table indexed by S requiring $2^{32} \cdot 2^8 = 2^{40}$ time. Afterwards, we try out all keys along the main diagonal and compute Λ^{f} from Λ_I. Instead of querying the encryption oracle, we look up every element of Λ^{f} in our table requiring $2^{32} \cdot 2^8 = 2^{40}$ time. If their Xor-sum equals zero, we have found the correct key. This reduces the time complexity to 2^{41} but requires 64 GiB of memory.

B Detailed Steps for Linear Attack on ZIP-GIFT-(10,9)

In Table 2, we list the detailed steps of the linear key recovery on ZIP-GIFT-(10,9).

C Linear Key Recovery Attack on $(9, 9)$-round ZIP-GIFT

In this section, we propose a linear attack on (9,9)-round of ZIP-GIFT by prepending 4 rounds to the forward branch, and 3 rounds to the backward branch for key recovery to the plaintext side of our $(5, 6)$-round linear distinguisher with correlation 2^{-29}. As illustrated in Figure 10, we want to compute the value of

$$S = \bigoplus_{i \in \mathcal{I}^{\mathrm{f}}} X_4[i] \oplus \bigoplus_{i \in \mathcal{I}^{\mathrm{b}}} X_{-3}[i] \oplus \bigoplus_{i \in \mathcal{I}^{\oplus}} C[i]$$

where $\mathcal{I}^{\mathrm{f}} = \{34, 35, 42, 43\}$, $\mathcal{I}^{\mathrm{b}} = \{14, 44\}$, and $\mathcal{I}^{\oplus} = \{0, 2, 8, 10, 32, 34, 40, 42\}$. To perform this attack, we start with 2^{62} known plaintext-ciphertext pairs, and we set the advantage of this attack as $a = 8$. So, the data complexity of this attack is 2^{62}, and the success probability is 80%. Table 3 summarizes the partial-sum procedures to compute the sum of S. Hence, the total time complexity of our attack is $2^{122.48} + 2^{128} \cdot 2^{-8} \approx 2^{122.48}$ $(9, 9)$-round of encryptions, as in this attack, we consider one memory access (`MA`) as one $(9, 9)$-round of encryptions.

Table 2: (10, 9)-round linear key recovery attack on ZIP-GIFT using partial-sum technique. gray: redundant key bits, yellow: already guessed in an earlier step due to key scheduling algorithm.

#	Key	Time	Stored Texts	Mem.	
0	$K_0[0..31], K_1[0..15], K_2[0..3, 16..19], EK_{-1}[10, 11, 18, 19, 26, 27], EK_{-2}[15]$	$2^{62+63-2.7}$	$EX_0[0..19, 24..35, 40..51, 56..63], EX_{-1}[17..19, 24, 25, 27, 28], S_0$	2^{60}	
1	$EK_{-1}[2, 3], EK_{-2}[14]$	$2^{64+60-7.2}$	$EX_0[0..3, 8..19, 24..35, 40..51, 56..63], EX_{-1}[16..19, 24..27], S_1$	2^{57}	
2	$EK_{-2}[8, 9], EK_{-3}[6]$	$2^{66+57-8.2}$	$EX_0[0..3, 8..19, 24..35, 40..51, 56..63], EX_{-1}[24..27], S_2$	2^{53}	
3	$EK_{-2}[12, 13]$	$2^{68+53-8.2}$	$EX_0[0..3, 8..19, 24..35, 40..51, 56-63], Y_{-3}[38], S_2$	2^{50}	
4	$EK_{-1}[22, 23]$	$2^{70+50-8.2}$	$EX_0[0..3, 8..19, 24..35, 40..43, 48..51, 56..63], EX_{-1}[56, 61], Y_{-3}[38], S_3$	2^{48}	
5	$EK_{-1}[14, 15], EK_{-2}[25]$	$2^{72+48-8.2}$	$EX_0[0..3, 8..19, 24..27, 32..35, 40..43, 48..51, 56..63], EX_{-1}[56, 59..61], Y_{-3}[38], S_4$	2^{46}	
6	$EK_{-1}[6, 7, 30, 31], EK_{-2}[24]$	$2^{77+46-6.7}$	$EX_0[0..3, 8..11, 16..19, 24..27, 32..35, 40..43, 48..51, 56..59], EX_{-1}[56..63], Y_{-3}[38], S_5$	2^{42}	
7	$EK_{-2}[28..30], EK_{-2}[31], EK_{-3}[18, 26]$	$2^{80+42-7.2}$	$EX_0[0..3, 8..11, 16..19, 24..27, 32..35, 40..43, 48..51, 56..59], Y_{-3}[36, 38, 52], S_5$	2^{36}	
8	$EK_{-1}[0, 1]$	$2^{82+36-8.2}$	$EX_0[8..11, 16..19, 24..27, 32..35, 40..43, 48..51, 56..59], EX_{-1}[0, 10, 15], Y_{-3}[36, 38, 52], S_5$	2^{35}	
9	$EK_{-1}[8, 9]$	$2^{84+35-8.2}$	$EX_0[8..11, 24..27, 32..35, 40..43, 48..51, 56..59], EX_{-1}[0, 1, 10..12, 15], Y_{-3}[36, 38, 52], S_5$	2^{34}	Intermediates:
10	$EK_{-1}[20, 21]$	$2^{86+34-8.2}$	$EX_0[8..11, 24..27, 32..35, 48..51, 56..59], EX_{-1}[0, 1, 10..12, 15, 34, 40, 45], Y_{-3}[36, 38, 52], S_5$	2^{33}	
11	$EK_{-1}[28, 29]$	$2^{88+33-8.2}$	$EX_0[8..11, 24..27, 32..35, 48..51], EX_{-1}[0, 1, 10..12, 15, 34, 35, 40, 41, 45, 46], Y_{-3}[36, 38, 52], S_5$	2^{32}	
12	$EK_{-1}[16, 17]$	$2^{90+32-8.2}$	$EX_0[8..11, 24..27, 48..51], EX_{-1}[0..2, 8, 10..13, 34, 35, 40, 41, 45, 46], Y_{-3}[36, 38, 52], S_5$	2^{30}	
13	$EK_{-1}[24, 25], EK_{-2}[7], EK_{-2}[6], EK_{-3}[27]$	$2^{93+30-7.2}$	$EX_0[8..11, 24..27], EX_{-1}[0..3, 8..11, 34, 35, 40, 41, 45, 46], Y_{-3}[36, 38, 52], S_6$	2^{26}	
14	$EK_{-2}[4, 5], EK_{-3}[19]$	$2^{95+26-8.2}$	$EX_0[8..11, 24..27], EX_{-1}[0..3, 34, 35, 40, 41, 45, 46], Y_{-3}[36..38, 52], S_6$	2^{23}	
15	$EK_{-2}[0, 1]$	$2^{97+23-8.2}$	$EX_0[8..11, 24..27], EX_{-1}[34, 35, 40, 41, 45, 46], Y_{-3}[15, 36..38, 52], S_6$	2^{20}	
16	$EK_{-1}[4, 5, 12, 13], EK_{-2}[22], EK_{-2}[23]$	$2^{102+20-6.2}$	$EX_{-1}[32..35, 40..43], Y_{-3}[15, 36..38], S_7$	2^{13}	
17	$EK_{-2}[16, 17, 20, 21], EK_{-3}[7]$	$2^{106+13-6.2}$	S'	1	
Σ		$2^{122.4}$		$2^{60.18}$	

$S_0 = \bigoplus_{i\in\mathcal{I}^f} X_4[i] \oplus \bigoplus_{i\in\mathcal{I}^\oplus} C[i] \oplus Y_{-2}[29] \oplus Y_{-2}[30]$, where $\mathcal{I}^f = \{34, 35, 42, 43\}$, and $\mathcal{I}^\oplus = \{21, 23, 29, 31, 53, 55, 61, 63\}$, $S_1 = S_0 \oplus Y_{-3}[54]$, $S_2 = S_1 \oplus Y_{-3}[12]$, $S_3 = S_2 \oplus Y_{-2}[50]$, $S_4 = S_3 \oplus Y_{-2}[49]$, $S_5 = S_4 \oplus Y_{-3}[14]$, $S_6 = S_5 \oplus Y_{-3}[53]$, $S_7 = S_6 \oplus X_{-3}[54]$, $S' = S_7 \oplus X_{-3}[15] \oplus X_{-3}[37]$.

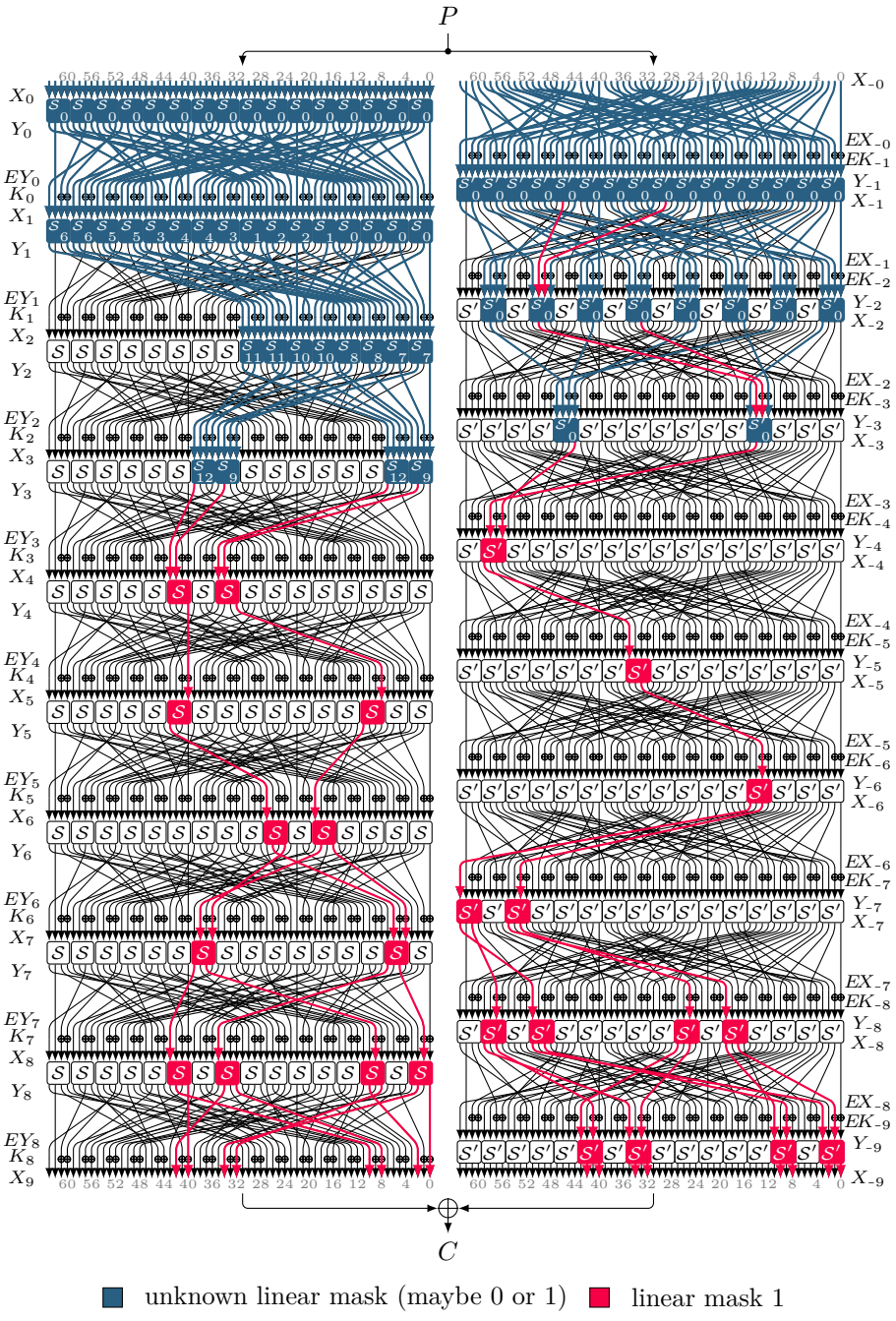

Figure 10: Linear attack on $(9, 9)$ rounds of ZIP-GIFT where the key recovery is performed on the first four rounds of the forward and the first three rounds of the backward branch

Table 3: Partial-sum technique for (9, 9)-round linear key recovery attack on ZIP-GIFT. Here the target sum is $S = \bigoplus_{i\in\mathcal{I}^{\mathrm{f}}} X_4[i] \oplus \bigoplus_{i\in\mathcal{I}^{\mathrm{b}}} X_{-3}[i] \oplus \bigoplus_{i\in\mathcal{I}^{\oplus}} C[i]$, where $\mathcal{I}^{\mathrm{f}} = \{34, 35, 42, 43\}$, $\mathcal{I}^{\mathrm{b}} = \{14, 44\}$, and $\mathcal{I}^{\oplus} = \{0, 2, 8, 10, 32, 34, 40, 42\}$. $S_1 : \bigoplus_{i\in\mathcal{I}^{\mathrm{b}}} X_{-3}[i] \oplus \bigoplus_{i\in\mathcal{I}^{\oplus}} C[i]$. $\mathcal{K}_{-2} : \{0, 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 28, 29\}$. $EY_i : P_{64}(Y_i)$, where $(0 \le i \le 2)$. $EK_j : P_{64}^{-1}(K_j)$, where $(-3 \le j \le -1)$. gray: redundant key bits

Step	Key	Time Complexity	Stored Texts	Memory
0	$EK_{-1}[0 - 31], EK_{-2}[\mathcal{K}_{-2}]$, $EK_{-3}[6, 22, 23], K_0[0 - 7]$, $EK_{-3}[5], EK_{-2}[25]$	$2^{58} \times 2^{62}$	$EY_0[16 - 63], EY_1[0 - 3, 16 - 19], \mathcal{S}_1$	2^{57}
1	$K_0[8, 9, 14, 15]$	$2^{62} \times 2^{57}$	$EY_0[20 - 27, 32 - 63], EY_1[0 - 4, 7, 16 - 21], \mathcal{S}_1$	2^{53}
2	$K_0[10 - 13]$	$2^{66} \times 2^{53}$	$EY_0[32 - 63], EY_1[0 - 7, 16 - 23], \mathcal{S}_1$	2^{49}
3	$K_0[16, 17, 22, 23]$	$2^{70} \times 2^{49}$	$EY_0[36 - 43, 48 - 63], EY_1[0 - 8, 11, 16 - 25], \mathcal{S}_1$	2^{45}
4	$K_0[18 - 21]$	$2^{74} \times 2^{45}$	$EY_0[48 - 63], EY_1[0 - 11, 16 - 27], \mathcal{S}_1$	2^{41}
5	$K_0[24 - 27]$	$2^{78} \times 2^{41}$	$EY_0[56 - 63], EY_1[0 - 13, 16 - 27, 29, 30], \mathcal{S}_1$	2^{37}
6	$K_0[29 - 31]$	$2^{82} \times 2^{37}$	$EY_1[0 - 31], \mathcal{S}_1$	2^{33}
7	$K_1[0 - 3]$	$2^{86} \times 2^{33}$	$EY_1[8 - 31], EY_2[0, 1, 34, 35], \mathcal{S}_1$	2^{29}
8	$K_1[4 - 7]$	$2^{90} \times 2^{29}$	$EY_1[16 - 31], EY_2[0 - 3, 32 - 35], \mathcal{S}_1$	2^{25}
9	$K_2[0, 1, 16, 17]$	$2^{94} \times 2^{25}$	$EY_1[16 - 31], \mathcal{S}_1 \oplus X_4[34] \oplus X_4[42]$	2^{17}
10	$K_1[8 - 11]$	$2^{98} \times 2^{17}$	$EY_1[24 - 31], EY_2[4, 5, 38, 39], \mathcal{S}_1 \oplus X_4[34] \oplus X_4[42]$	2^{13}
11	$K_1[12 - 15]$	$2^{102} \times 2^{13}$	$EY_2[4 - 7, 36 - 39], \mathcal{S}_1 \oplus X_4[34] \oplus X_4[42]$	2^{9}
12	$K_2[2, 3, 18, 19]$	$2^{106} \times 2^{9}$	S	1
$\sum$		$2^{122.48}$ memory access (MA)		$2^{57.09}$