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
marcel.nageler@tugraz.at,
[2] NTT Social Informatics Laboratories, Tokyo, Japan
debasmita.chakraborty@ntt.com

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 focus our efforts on the unique challenges of performing key recovery attacks for ZIP ciphers and propose new techniques to overcome these challenges. We show differential, linear, and integral key recovery attacks for both PRFs. We develop new techniques for integral key recovery attacks and show how to extend differential characteristics by some rounds for key recovery.

Keywords: PRFs · ZIP-AES · ZIP-GIFT · Differential cryptanalysis · Linear cryptanalysis · Integral 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). The problem of 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 first introduced by Bellare et al. at EUROCRYPT 1998 [4]. A formal security proof was later provided by Lucks at EUROCRYPT 2000 [26], and subsequent works have refined and strengthened these results [3,29,11].

Although the construction of such PRFs is of high interest, there is little discussion on their cryptanalysis, largely because no concrete instances have been designed. The first notable proposal appears to be Orthros [1], 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 conducted a careful study of its resistance against known cryptanalytic techniques, their specification did not present any key-recovery attack. Subsequently, differential and differential-linear weak-key attacks were proposed [25,19] (the weak-key property of [25] was noted in [15]), the differential clustering effect due to the ciphertext XOR was analyzed [35], and universal-key attacks based on weak-key distinguishers where proposed [17].

At ASIACRYPT 2024, Flórez-Gutiérrez et al. proposed a framework termed *general practical cryptanalysis* [15], in which they examined the relationship between the security of the sum of two keyed permutations, $E_0 \oplus E_1$, and that of their composition $E_1^{-1} \circ E_0$. Their results demonstrated that many distinguishers on the sum construction can be translated to corresponding distinguishers on the composition. Considering differential, linear, and integral cryptanalysis, they observed 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 attacks.

To apply their framework, they introduced the ZIP construction [15]. Let $E = E_1 \circ E_0$ denote a secure iterative block cipher; the corresponding ZIP version of E is then defined as $E_0 \oplus E_1^{-1}$. This design offers several appealing features. First, due to the close relationship between the two constructions, much cryptanalysis on E can be transferred. Second, the ZIP construction 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 instantiation, the authors presented ZIP-AES, which combines 5 encryption rounds of AES with 5 decryption rounds.

In most cases, attacks on symmetric primitives 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 the ZIP structure. In [15], 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 designs. 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 constructing 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 check whether the distinguisher holds. For ZIP constructions this strategy is problematic because 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 knows or chooses the input to both branches. However, the distinguisher needs to hold for both branches, and extending such two-branch distinguishers back to the plaintext is particularly difficult for chosen-plaintext attacks. The situation is further complicated as the two branches use different round functions and round keys. Given these interesting challenges, we show key recovery attacks for 2 concrete instances: ZIP-AES and ZIP-GIFT.

Our Contributions. In this paper, we present differential, linear, and integral key-recovery attacks on two concrete ZIP constructions, ZIP-AES and ZIP-GIFT, as summarized in Table 1. These attacks provide the first concrete analysis of key-recovery attacks on these ZIP instances, highlighting the unique challenges in this setting and proposing techniques to overcome them:

- *Differential key recovery.* Motivated by the criterion in [15, 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. Concretely, we find that it is possible to mount key recovery attacks on more rounds than can be covered by a single differential characteristic or cluster. Our strategy is to extend the distinguisher at the output side but still perform key recovery at the input. 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) by prepending key-recovery rounds to the linear characteristics we find for both ciphers. To lower the time complexity we apply 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 [15], 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 obtain integral key-recovery attacks on (3,3)-round ZIP-AES and (8,8)-round ZIP-GIFT.

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	Differential	–	–	$(3,2)$	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	$(4,4)$	[15]	$(3,3)$	2^{64}	2^{42}	≈ 1	Sect. 5.1
		$(4,4)$	[15]	$(4,2)$	2^{96}	2^{40}	2^{32}	Sect. A
ZIP-GIFT	Differential	$(6,7)$	Sect. 3.2	$(8,8)$	$2^{125.3}$	2^{51}	$2^{23.29}$	Sect. 3.2
		$(6,7)$	Sect. 3.2	$(7,8)$	$2^{109.93}$	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	$(9,9)$	$2^{122.48}$	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 information. 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 most important findings and conclude in Section 6. Our code is available on Github[3].

2 Background

2.1 Brief Specification of ZIP-AES

ZIP-AES is a pseudorandom function (PRF) proposed by Flórez-Gutiérrez et al. at ASIACRYPT 2024 [15]. It uses the AES round function $R = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$ and its inverse R^{-1} [10]. 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 $\mathsf{AES}_5^{-1}(x)$ uses $K_6, \ldots, K_{10}, 0^{128}$. We depict a round-reduced version, $\mathsf{ZIP\text{-}AES}_{2,2}$, in Figure 1.

2.2 Brief Specification of ZIP-GIFT

ZIP-GIFT is a 64-bit PRF, derived from the GIFT-64 [2] block cipher:

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

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

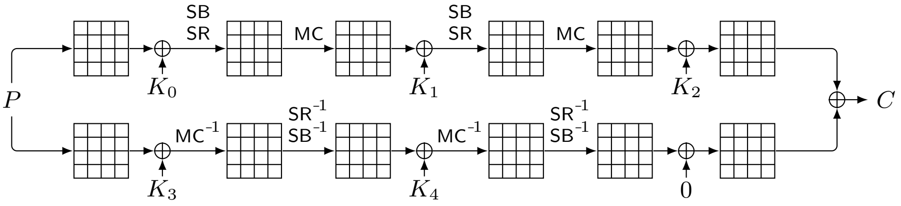


Fig. 1: Specification of ZIP-AES, illustrated for $(r^{\mathrm{f}}, r^{\mathrm{b}}) = (2, 2)$.

$\mathsf{GIFT}_{r^{\mathrm{f}}}$ 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 the 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 $K_{\text{-}i}$ are created using the inverse function L^{-1}. We show a reduced-round version of ZIP-GIFT in Figure 2. Note that, consistent with the GIFT-64 specification, 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 $EY_{\text{-}i} = P_{64}(Y_i)$ for $0 \leq i \leq r^{\mathrm{f}} - 1$. Similarly, in the backward branch, we define $EX_{\text{-}j+1} = P^{-1}_{64}(X_{\text{-}j+1})$ and $EK_{\text{-}j} = P^{-1}_{64}(K_{\text{-}j})$ for $1 \leq j \leq r^{\mathrm{b}}$. Throughout our discussion of differential and linear attacks on ZIP-GIFT, we adopt this equivalent-key view for clarity and simplicity.

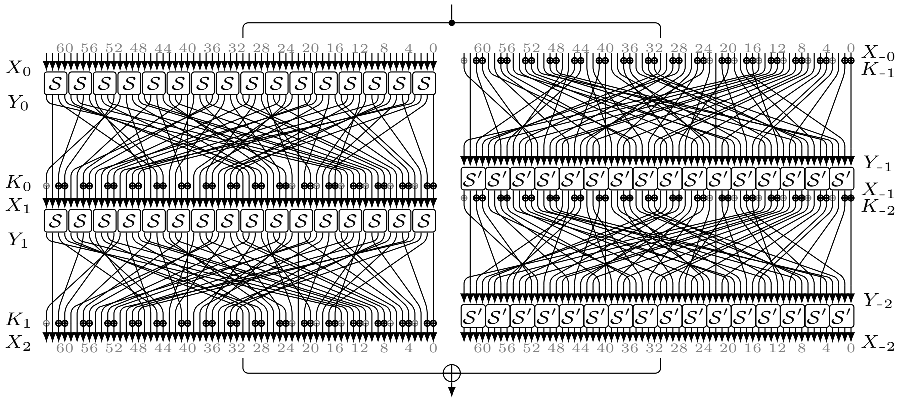


Fig. 2: Specification of ZIP-GIFT, illustrated for $(r^{\mathrm{f}}, r^{\mathrm{b}}) = (2, 2)$.

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.

Differential Cryptanalysis [5]. In differential cryptanalysis, a statistical chosen-plaintext attack, the distinguisher is given the differences $(\Delta_{\text{in}}, \Delta_{\text{out}})$ 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:

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

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 [21,31,7].

Linear Cryptanalysis [27]. Linear cryptanalysis is a statistical known-plaintext technique that exploits linear relations 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:

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

Matsui's Algorithm 2 provides a practical last-round partial-key recovery method for such approximations [27]. Subsequent improvements, most notably FFT-based optimization by Collard et al. [8] and the partial-sums technique by Ferguson et al. [13], substantially lowered the complexity of the partial key recovery.

Integral Cryptanalysis [23,9,36]. 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 sum of outputs then represents the d-th derivative. Output bits whose Algebraic Normal Form (ANF) lacks terms combining active inputs and key bits are constant, yielding balanced properties. To find specific monomials in the ANF, monomial prediction is commonly used [36,20]. 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 [13,37].

2.4 Notation

We first fix notation. The ZIP cipher $E^{\oplus}$ can be decomposed as

$$E^{\oplus} = (E^{\text{f}}_{\text{dist}} \circ E^{\text{f}}_{\text{key}}) \oplus (E^{\text{b}}_{\text{dist}} \circ E^{\text{b}}_{\text{key}}),$$

where $E^{\mathrm{f}}_{\mathrm{key}}, E^{\mathrm{b}}_{\mathrm{key}}$ cover $r^{\mathrm{f}}_{\mathrm{key}}, r^{\mathrm{b}}_{\mathrm{key}}$ rounds and $E^{\mathrm{f}}_{\mathrm{dist}}, E^{\mathrm{b}}_{\mathrm{dist}}$ cover $r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}}$ rounds. An attack is built by first constructing a distinguisher for $E^{\mathrm{f}}_{\mathrm{dist}} \oplus E^{\mathrm{b}}_{\mathrm{dist}}$ (spanning $(r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}})$ rounds) and then recovering the key for $E^{\mathrm{f}}_{\mathrm{key}}$ and $E^{\mathrm{b}}_{\mathrm{key}}$. Below, we show the constraints for differential, linear, and integral key-recovery attacks for ZIP ciphers, explain how they are overcome, and give the resulting procedures.

3 Differential Cryptanalysis on ZIP Ciphers

In this section, we explore the differential cryptanalysis for 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 ZIP ciphers, as initially studied in [15]. Then, we show differential key recovery attacks on ZIP-AES and ZIP-GIFT.

In [15], the authors showed that differential characteristics of the sum construction are related to those of the composition:

Proposition 1 ([15]). *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 probability at most probability 2^{-150}, ZIP-AES cannot admit any $(2, 2)$-round characteristic with probability above 2^{-150}. Likewise, longest differential characteristic for GIFT-64 spans 13 rounds with probability 2^{-62} [24]. Hence no differential characteristic for $(7, 7)$-round ZIP-GIFT can have probability better than 2^{-64}.

In [15], the authors highlighted the inherent difficulty of mounting differential key-recovery attacks on ZIP constructions, as one must always prepend rounds above the differential distinguisher in both branches. In this context, the following result was established:

Proposition 2 ([15]). *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*

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

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 [5].

Furthermore, Proposition 2 [15, 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 of all, 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 an 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 probability 1 truncated extension to attack more rounds, recovering key bits at the output is hardly possible. The only option left is to recover the key (mostly) 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 adress 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 considernig 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 addtive, every step can possibly slow down the entire attack.

Building on these observations, we proceed to mount differential key-recovery attacks on ZIP-AES and ZIP-GIFT, as described in the following sections.

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 optimized 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.

Attack Procedure.

1. Encrypt 2^{42} plaintexts (as explained below) to obtain 2^{42} ciphertexts.
2. Build 2^{65} ciphertext pairs (as explained below) and filter them by the output pattern, resulting in $\frac{2^{65}}{2^{32}} = 2^{33}$ surviving pairs.
3. For every surviving pair, with plaintext difference Δ_{in}, do the following:
   (a) 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 approximately 2^{32}.
   (b) For every $\Delta_{\text{R}} \in M_{\Delta}$, let $\Delta_{\text{out}} = \mathsf{SR}^{-1}(\Delta_{\text{R}})$.
      i. Denote by $\Delta_{\text{in}}^{(i)}, \Delta_{\text{out}}^{(i)}$ the i-th cell of the corresponding AES states and let $a^{(i)}$ be the i-th cell of one of the plaintexts corresponding to the plaintext difference Δ_{in}.

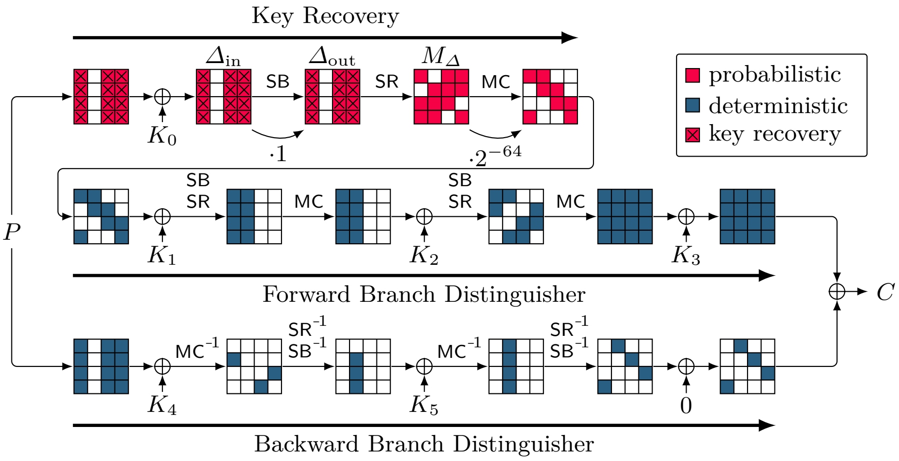


Fig. 3: Truncated differential distinguisher for (3,2)-rounds of ZIP-AES, where key recovery is performed in the first round of the forward branch.

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

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

iii. Combine all sets K_i via the cartesian product, producing a set of 96 bit candidate keys with expected size 1.

iv. Recover the remaining 32 bits of the key using brute force by testing each candidate key through encryption of the plaintext.

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 probabilistic 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 $n = 2^{65}$ plaintext pairs, so we expected 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 with a linear space $\mathcal{L}$ of dimension 24. For an AES state P_o, the affine space $\mathcal{L} \oplus P_o$ then 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}.

Runtime Evaluation. On average, we expect that $|X_i| = 1$ in step 3(b)ii 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

Proposition 2 establishes that a differential key-recovery attack on an r-round ZIP construction $E^{\oplus}$ requires an r-round differential characteristic with probability strictly greater than 2^{-n}. For GIFT-64, the longest known differential characteristics cover 13 rounds. By applying 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 should be 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 exploit a single differential characteristic of ZIP-GIFT together with a truncated extension at the output. This approach enables key recovery beyond the $(7, 6)$ or $(6, 7)$-round for ZIP-GIFT.

Specifically, using this technique, we mount a differential key-recovery attack on $(8, 8)$ rounds of ZIP-GIFT with data complexity 2^{51} and time complexity $2^{125.3}$. For comparison, the best known differential attack on GIFT-64—based on extending a 13-round distinguisher to 20 rounds—requires $2^{62.58}$ data, $2^{125.50}$ time, and approximately $2^{62.58}$ memory [34]. 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 [24]. Our key-recovery attack is based on a (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 Sbox and inverse Sbox. 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 behaviour for differential characteristics [30], we generate multiple characteristics and analyze them for key dependencies with AutoDiVer [28]. We select a characteristic without key dependencies.

Our $(8, 8)$-round truncated differential characteristic $\varDelta \xrightarrow{E^{\oplus}} \varDelta^{\oplus}$ with probability $p = 2^{-60}$, can be described as follows:

$$\varDelta = \texttt{0a00 0000 0a00 0000}.$$

The resulting output-difference $\varDelta^{\oplus} \in \mathbb{F}_2^{64}$ is constrained on a set of bit positions

$$\mathcal{I}^{\oplus} = \{1, 3, 5, 7, 9, 11, 13, 15, 33, 35, 37, 39, 41, 43, 45\} \subseteq \{0, \ldots, 63\}.$$

Then the output bits $i \in \mathcal{I}^{\oplus}$ are known to have a zero difference:

$$\varDelta^{\oplus}[i] = 0, \quad \forall i \in \mathcal{I}^{\oplus},$$

while the other bits are unknown. This gives us an initial upper bound for the set of ciphertext differences of $|\varDelta^{\oplus}| \leq |2^{64-|\mathcal{I}^{\oplus}|} = 2^{49}$.

Precise Estimation of Ciphertext Filter. To improve the estimate for the amount of possible ciphertext differences, we start by listing all possible 64-bit differences in the forward and backward branches just before they Xor. Concretely, we start with the differences after the (6,6) rounds of differential propagation:

$$\varDelta_I^{\mathrm{f}} = \mathtt{0000\ 0800\ 0000\ 0800},$$
$$\varDelta_I^{\mathrm{b}} = \mathtt{0000\ 00a0\ 0000\ 00a0}.$$

For both $\varDelta_I^{\mathrm{f}}$ and $\varDelta_I^{\mathrm{b}}$, we generate all possible 64-bit differences after two rounds, resulting in two sets $\varDelta_O^{\mathrm{f}}$ and $\varDelta_O^{\mathrm{b}}$, with $|\varDelta_O^{\mathrm{f}}| = 2^{20.98}$ and $|\varDelta_O^{\mathrm{b}}| = 2^{17.93}$. This gives an improved upper bound of $|\varDelta^{\oplus}| \leq |\varDelta_O^{\mathrm{f}}| \cdot |\varDelta_O^{\mathrm{b}}| = 2^{38.82}$. Counting the distinct values of $\varDelta^{\oplus}$ 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 [14,18]. The algorithm provides an estimate of $|\varDelta^{\oplus}| \approx 2^{37.29}$, 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. Using $b = 18$ results in a standard error of $\frac{1.04}{\sqrt{2^b}} \approx 2^{-8.9}$ corresponding to a standard deviation of $|\varDelta^{\oplus}| \cdot 2^{-8.9} \approx 2^{28.4}$. Note that to verify whether a given difference $\varDelta C$ is a member of $\varDelta^{\oplus}$, we can iterate over $\varDelta_O^{\mathrm{f}}$, xor with $\varDelta C$, and perform a constant-time lookup in $\varDelta_O^{\mathrm{b}}$; this does not significantly affect overall time and memory complexity of the attack.

Key Recovery Steps. As we already know, to mount a differential key recovery attack for an n-bit $(r^{\mathrm{f}}, r^{\mathrm{b}})$ round of ZIP cipher $E^{\oplus}$, we need a differential distinguisher $\varDelta \xrightarrow{E^{\oplus}} \varDelta^{\oplus}$ with probability $p \gg 2^{-n}$. Given this constraint, we present $(8, 8)$-round attack based on the $(8, 8)$-round truncated differential characteristic, where the input difference is fixed and the output difference is a set of $|\varDelta^{\oplus}| \approx 2^{37.29}$ differences. We recover the key in the first 4 rounds of both branches as depicted in Figure 4 using the following steps:

1. We choose $N = 2^{50}$ plaintext pairs according to the structure outline below, and obtain the corresponding 2^{50}ciphertext pairs. We expect at least one correct pair with $p \approx 63\,\%$ as we amplify the probability of the characteristic by 2^{10} by deterministically satisfying some differential transitions.
2. We examine each ciphertext pair and retain only those for which the difference belongs to $\varDelta^{\oplus}$. Using this filtering criterion, the expected number of surviving pairs is roughly $N \cdot \frac{|\varDelta^{\oplus}|}{2^{64}} = N \cdot 2^{-26.71} = 2^{23.29}$. For each surviving pair (P, P'), we perform the following steps.
   (a) To simplify the later explanation, we now define $\mathrm{XDDT}^{\mathrm{f}}_{\varDelta_{\mathrm{i}} \to \varDelta_{\mathrm{o}}} = \{\, x \mid \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \varDelta_{\mathrm{i}}) = \varDelta_{\mathrm{o}}\}$ for the GIFT S-box $\mathcal{S}$. Analogously, we define $\mathrm{XDDT}^{\mathrm{b}}_{\varDelta_{\mathrm{i}} \to \varDelta_{\mathrm{o}}} = \{\, x \mid \mathcal{S}'(x) \oplus \mathcal{S}'(x \oplus \varDelta_{\mathrm{i}}) = \varDelta_{\mathrm{o}}\}$ for the inverse S-box $\mathcal{S}'$.

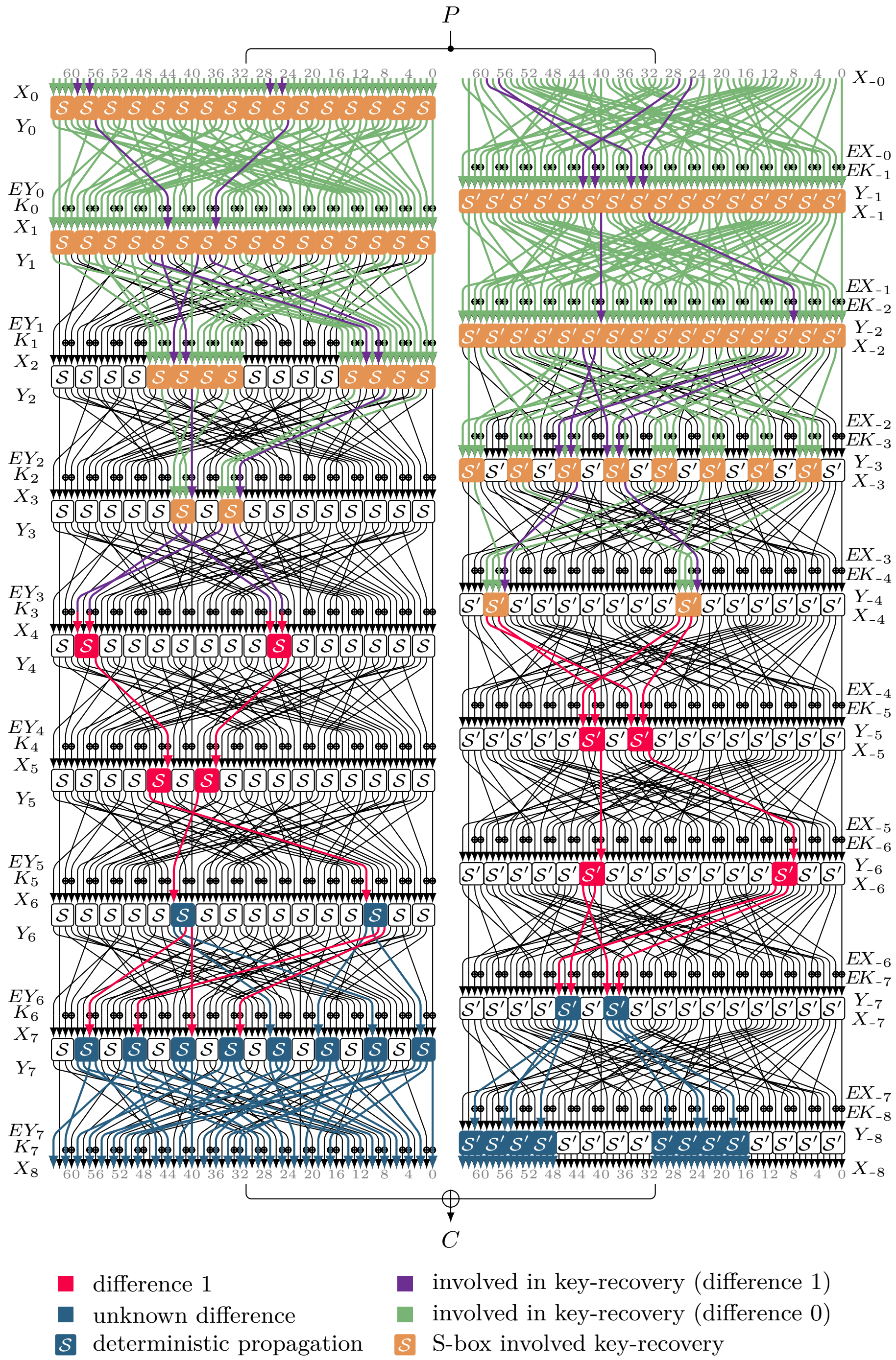


Fig. 4: Differential attack on $(8, 8)$ rounds of ZIP-GIFT where the key recovery is performed on the first 4 rounds of the forward and the backward branch.

(b) First, we compute $EY_0[0..63]$ for each plaintext in the pair. Given $\mathit{\Delta} X_1[44..47] \to \mathit{\Delta} Y_1[44..47] = \mathtt{1} \to \mathtt{a}$, we find

$$X_1[44..47] \in \mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}.$$

Therefore, we always have $X_1[45] = 1$. Consequently, we can determine one key bit: $K_0[23] = EY_0[45] \oplus X_1[45]$. A similar analysis further yields the value of $K_0[19]$. After recovering these bits, we guess the remaining 30 unknown key bits of K_0 and subsequently compute EY_1 for each guess.

(c) Given $\mathit{\Delta} X_2[40..43] \to \mathit{\Delta} Y_2[40..43] = \mathtt{a} \to \mathtt{1}$, we find

$$X_2[40..43] \in \mathrm{XDDT}^{\mathrm{f}}_{\mathtt{a}\to\mathtt{1}} = \{\mathtt{3}, \mathtt{5}, \mathtt{9}, \mathtt{f}\}.$$

Consequently, each computed value of $EY_1[40..43]$ uniquely determines $X_2[40, 41]$, from which the two key bits $K_1[20, 21]$ can be derived. A similar analysis further yields $K_1[4, 5]$. Moreover, we have $\mathit{\Delta} X_3[32..35] \to \mathit{\Delta} Y_3[32..35] = \mathit{\Delta} X_3[40..43] \to \mathit{\Delta} Y_3[40..43] = \mathtt{1} \to \mathtt{a}$ and $\mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}$ which forces $X_3[34] = X_3[42] = 1$ and $X_3[35] = X_3[43] = 0$ for each plaintext. Additionally, from the bit permutation in the forward branch, $X_3[34] = Y_2[2]$, and with the definition of

$$\mathcal{Y}_2 = \{x \mid \mathcal{S}(x) = \mathtt{*1**}\} = \{\mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{9}, \mathtt{b}, \mathtt{c}, \mathtt{f}\}$$

we deduce that the possible values for $X_2[0..3]$ belongs to the set $\mathcal{Y}_2$. Together with

$$\begin{cases} EY_1[3] = X_2[3], \quad EY_1[2] = X_2[2], \\ EY_1[1] \oplus K_1[1] = X_2[1], \quad EY_1[0] \oplus K_1[0] = X_2[0], \end{cases}$$

we deduce that $X_2[0, 1]$ can only take two values (depending on $X_2[2, 3]$ from $\mathcal{Y}_2$), which in turn restricts $K_1[0, 1]$ to two possibilities. Similarly, from $X_3[35] = 0 = Y_2[7]$, $X_3[42] = 1 = Y_2[34]$, and $X_3[43] = 0 = Y_2[39]$, we derive two possible values each for $K_1[2, 3]$, $K_1[16, 17]$, and $K_1[18, 19]$. Thus, the possible candidates for $(K_1[0..3, 16..19])$ are 2^4. Finally, we guess the full set $(K_1[0..7, 16..23])$, resulting in 2^8 candidates, and compute $EY_2[32..35, 40..43]$.

(d) We already know that

$$X_3[32..35] \in \mathrm{XDDT}^{\mathrm{f}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{6}, \mathtt{7}\}$$

which implies $X_3[33] = 1$. Consequently, for each computed value of $EY_2[32..35, 40..43]$, we can deduce one key bit as $K_2[17] = EY_2[33] \oplus X_3[33]$. By applying the same reasoning, we also obtain $K_2[21]$. To proceed, we guess $K_2[16, 20]$ and compute $EY_3[25, 27, 57, 59]$ for each plaintext in the pair. This allows us to derive $\mathit{\Delta} X_4$, which is then verified against the expected difference pattern at X_4, consistent with the given differential characteristic. As a result, the number of possible key candidates in the first 4 rounds of the forward branch of $E^{\oplus}$ that satisfy

$$E^{\mathrm{f}}_{\mathrm{key}}(P) \oplus E^{\mathrm{f}}_{\mathrm{key}}(P') = \texttt{0a00 0000 0a00 0000}$$

for every valid pair (P, P') is 2^{40}, where $E^{\mathrm{f}}_{\mathrm{key}}$ denotes the first 4 rounds of the forward branch of $E^{\oplus}$.

(e) From (P, P'), we know $EX_{-0}[0..63]$. Given $\mathit{\Delta} Y_{-1}[40..43] \rightarrow \mathit{\Delta} X_{-1}[40..43] = \mathit{\Delta} Y_{-1}[32..35] \rightarrow \mathit{\Delta} X_{-1}[32..35] = \texttt{1} \rightarrow \texttt{a}$, we can construct the set $\mathcal{X}_{-1}$:

$$\mathcal{X}_{-1} = \mathrm{XDDT}^{\mathrm{b}}_{\texttt{1}\rightarrow\texttt{a}} = \{\texttt{3}, \texttt{9}\}.$$

From $\mathcal{X}_{-1}$ and each value of $EX_{-0}[40..43]$ one can uniquely determine $Y_{-1}[40..43]$, from which the two key bits $EK_{-1}[20, 21]$ can be derived. A similar approach helps to determine $EK_{-1}[16, 17]$. Due to $\mathit{\Delta} Y_{-2}[8..11] \rightarrow \mathit{\Delta} X_{-2}[8..11] = \mathit{\Delta} Y_{-2}[40..43] \rightarrow \mathit{\Delta} X_{-2}[40..43] = \texttt{1} \rightarrow \texttt{a}$, we have

$$Y_{-2}[8..11] \in \mathrm{XDDT}^{\mathrm{f}}_{\texttt{1}\rightarrow\texttt{a}} = \{\texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\},$$
$$Y_{-2}[40..43] \in \mathrm{XDDT}^{\mathrm{f}}_{\texttt{1}\rightarrow\texttt{a}} = \{\texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\},$$

which forces $X_{-1}[2] = X_{-1}[10] = X_{-1}[19] = X_{-1}[27] = 1$ for each plaintext. Focusing on $X_{-1}[2] = 1$, we can compute for the GIFT S-box:

$$\mathcal{Y}_{-1} = \{x \mid \mathcal{S}'(x) = \texttt{*1**}\}. = \{\texttt{0}, \texttt{3}, \texttt{5}, \texttt{6}, \texttt{8}, \texttt{9}, \texttt{e}, \texttt{f}\}$$

Therefore, we conclude that the possible values for $Y_{-1}[0..3]$ belongs to the set $\mathcal{Y}_{-1}$. Finally, from $\mathcal{Y}_{-1}$ and each value of $EX_{-0}[0..3]$, we deduce that $(Y_{-1}[0], Y_{-1}[1])$ can take only two values (depending on $(Y_{-1}[2], Y_{-1}[3])$ from $\mathcal{Y}_{-1}$) which eventually restricts $(EK_{-1}[0, 1])$ to two possibilities. Similarly, from $X_{-1}[10] = X_{-1}[19] = X_{-1}[27] = 1$, we derive two possible values each for $EK_{-1}[4, 5], EK_{-1}[8, 9]$, and $EK_{-1}[12, 13]$. Therefore, the possible key candidates for $EK_{-1}[0, 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21]$ are 2^4. Finally, we guess the remaining 20 key bits in EK_{-1}, resulting in 2^{24} key candidates, and compute EX_{-1}.

(f) Based on $\mathit{\Delta} Y_{-3}[44..47] \rightarrow \mathit{\Delta} X_{-3}[44..47] = Y_{-3}[36..39] \rightarrow \mathit{\Delta} X_{-3}[36..39] = \texttt{a} \rightarrow \texttt{1}$, we find

$$Y_{-3}[44..47] \in \mathrm{XDDT}^{\mathrm{b}}_{\texttt{a}\rightarrow\texttt{1}} = \{\texttt{3}, \texttt{9}\}$$
$$Y_{-3}[36..39] \in \mathrm{XDDT}^{\mathrm{b}}_{\texttt{a}\rightarrow\texttt{1}} = \{\texttt{3}, \texttt{9}\}$$

which forces $X_{-2}[58] = X_{-2}[26] = 0$ for each plaintext. Focusing on $X_{-2}[58] = 0$, we compute the following:

$$\mathcal{Y}_{-2} = \{x \mid \mathcal{S}'(x) = \texttt{*0**}\} = \{\texttt{1}, \texttt{2}, \texttt{4}, \texttt{7}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$$

Then, we can conclude that the possible values for $Y_{-2}[56..59]$ belongs to the set $\mathcal{Y}_{-2}$. Finally, from $\mathcal{Y}_{-2}$ and each value of $EX_{-1}[56..59]$, we can deduce that $(Y_{-2}[56, 57])$ can take only two values (depending on $Y_{-2}[58, 59]$

from $\mathcal{Y}_{\text{-}2}$), which eventually restricts $EK_{\text{-}2}[28, 29]$ to two possibilities. From $X_{\text{-}2}[26] = 0$, with a similar analysis, we derive two possible values $EK_{\text{-}2}[12, 13]$. Hence, the possible key candidates for $EK_{\text{-}2}[12, 13, 28, 29]$ is 2^2. Additionally, taking the key scheduling algorithm of ZIP-GIFT into account, the key bits $EK_{\text{-}2}[5, 6, 22, 23]$ has already been either guessed or determined in Step 2d. Finally, we guess the remaining 24 bits of $EK_{\text{-}2}$ and compute $EX_{\text{-}2}$.

(g) In this step, given the differential transitions we have

$$Y_{\text{-}3}[44..47] \in \text{XDDT}^{\text{b}}_{\mathtt{a}\to\mathtt{1}} = \{3, 9\}$$
$$Y_{\text{-}3}[36..39] \in \text{XDDT}^{\text{b}}_{\mathtt{a}\to\mathtt{1}} = \{3, 9\}.$$

Utilizing an analysis similar to before, we can uniquely determine $EK_{\text{-}3}[18, 19]$. Furthermore, we have

$$Y_{\text{-}4}[24..27] \in \text{XDDT}^{\text{b}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\},$$
$$Y_{\text{-}4}[56..59] \in \text{XDDT}^{\text{b}}_{\mathtt{1}\to\mathtt{a}} = \{\mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}.$$

which forces $X_{\text{-}3}[6] = X_{\text{-}3}[14] = X_{\text{-}3}[23] = X_{\text{-}3}[31] = 1$ for each plaintext in the pair. Therefore, using the similar arguments to Step 2e, we can reduce the number of key candidates for $EK_{\text{-}3}[2, 3, 6, 7, 10, 11, 14, 15]$ from 2^8 to 2^4. Furthermore, using key scheduling algorithm of ZIP-GIFT we have already guessed $EK_{\text{-}3}[2, 6, 10, 14, 26, 27, 30, 31]$ in Step 2c. This helps us to determine the value of $EK_{\text{-}3}[3, 7, 11, 15]$ uniquely for each guess of $EK_{\text{-}3}[2, 6, 10, 14]$. Hence, we do not have to guess any new key bits in this step to determine $EX_{\text{-}3}$[24..27,56..59].

(h) To compute, $\Delta X_{\text{-}4}$ for each of the plaintext (P, P'), we need to guess $EK_{\text{-}4}[12, 13, 28, 29]$. Although, using the key scheduling algorithm of ZIP-GIFT, we already guessed $EK_{\text{-}4}[12, 13, 28, 29]$ in Step 2b. Hence, the number of possible key candidates in the first 4 rounds of both the branches of $E^{\oplus}$ that satisfy

$$E^{\text{f}}_{\text{key}}(P) \oplus E^{\text{f}}_{\text{key}}(P') = \mathtt{0a00\,0000\,0a00\,0000},$$
$$E^{\text{b}}_{\text{key}}(P) \oplus E^{\text{b}}_{\text{key}}(P') = \mathtt{0a00\,0000\,0a00\,0000}.$$

for every valid pair (P, P') is 2^{90}, where the total number of involved keys are 116, and the remaining key bits are 12. Here, $E^{\text{f}}_{\text{key}}$, and $E^{\text{b}}_{\text{key}}$ denote the first 4 rounds of the forward and backward branch of $E^{\oplus}$, respectively. To recover the 128-bit master key, we perform a brute force search over $2^{90+12} = 2^{102}$ candidates for each surviving pair (P, P').

Complexity analysis. To mount a valid differential key-recovery attack, we must construct a sufficient number of plaintext pairs that follow the plaintext differences in both branches, i.e. $\Delta_{\text{fwd}} = \Delta_{\text{bwd}} = \mathtt{0a00\ 0000\ 0a00\ 0000}$. Based on our $(8, 8)$-round differential characteristic, we therefore select plaintext pairs

(P, P') satisfying

$$\begin{aligned}
&P[24..28] \oplus P'[24..28] = \mathtt{a}, \\
&P[56..59] \oplus P'[56..59] = \mathtt{a}, \\
&P[52..55] = P'[52..55] \in \{\mathtt{0}, \mathtt{2}, \mathtt{4}, \mathtt{6}, \mathtt{8}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}, \\
&P[48..51] = P'[48..51] \in \{\mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{9}, \mathtt{b}, \mathtt{c}, \mathtt{f}\}, \\
&P[10] = P'[10] = 0, \quad P[42] = P'[42] = 0, \\
&P[i] = P'[i] \in \{0, 1\} \quad \text{for all } i \in \{0, \ldots, 63\} \setminus \{10, 24..28, 42, 48..59\}.
\end{aligned}$$

Under these constraints the differential $E^{\oplus}(P) \oplus E^{\oplus}(P') = \Delta^{\oplus}$ holds with probability 2^{-50}. Hence, we need to construct $N = 2^{50}$ such plaintext pairs, which corresponds to a data complexity of 2^{51} plaintexts (counting both elements of each pair).

Following the attack procedure, Step 2 yields an expected number of surviving pairs of approximately $2^{23.29}$. For every surviving pair, we perform the remaining key-recovery steps. Thus, the time complexity for recovering the master key bits is dominated by $2^{102} \cdot 2^{23.29} = 2^{125.29} \approx 2^{125.3}$. It is noteworthy that, the $(7, 8)$ truncated differential (same as the previous one) yields a differential key-recovery on $(7, 8)$-round ZIP-GIFT with time $\approx 2^{109.93}$, data 2^{51} and memory $2^{7.93}$.

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. Our discussion begins with the observation made in [15].

Linear Characteristics. In [15], the authors demonstrated that the linear trails of the sum construction are related to those of the composition construction, as stated in the following result:

Proposition 3 ([15]). *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 established that any 4-round linear trail in AES involves at least 25 active S-boxes, leading to a squared correlation of 2^{-150}. Consequently, using Proposition 3, for ZIP-AES, no $(2, 2)$-round linear trail exists with $c^2 > 2^{-150}$. For ZIP-GIFT, the authors in [34] conjecture—based on the experimental findings of [33]—that GIFT-64 has no 13-round linear characteristic with linear potential exceeding 2^{-64}, since, the maximum correlation observed for 13-round trails is 2^{-34}, and the cipher exhibits only a weak linear hull effect. From this, we infer that the longest linear trails for ZIP-GIFT cover $(6, 6)$ rounds.

On Key Recovery in Linear Cryptanalysis. As noted in [15], linear key recovery attacks on ZIP constructions can be achieved by prepending rounds before the linear distinguisher in both branches. Concretely, given a linear characteristic of $E^{\mathrm{f}}_{\mathrm{dist}} \oplus E^{\mathrm{b}}_{\mathrm{dist}}$, one can prepend $E^{\mathrm{f}}_{\mathrm{key}}$ and $E^{\mathrm{b}}_{\mathrm{key}}$ and attempt key recovery on these parts, provided the total key material needed to evaluate the input mask parities to $E^{\mathrm{f}}_{\mathrm{dist}}$ and $E^{\mathrm{b}}_{\mathrm{dist}}$ 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 what follows, we identify suitable linear characteristics for ZIP-AES and ZIP-GIFT and demonstrate corresponding key recovery attacks, which are detailed in the subsequent sections.

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) [8] 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 approximately $2^{123.7}$ 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.

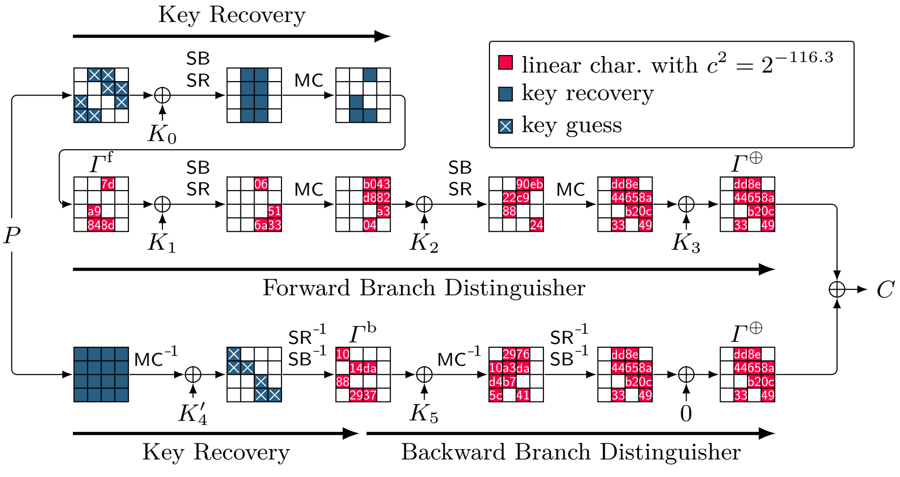


Fig. 5: Linear attack on (3,2)-rounds of ZIP-AES with a (2,1)-round linear characteristic with correlation $\approx 2^{-58.15}$ where we prepend one key recovery round at the beginning of the forward and backward branches.

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 [12] and heuristically found a solution that mostly uses the optimal S-box transitions with $c^2 = 2^{-6}$. We observe that there are no characteristics using only optimal S-box transitions for this cellwise pattern.

Attack procedure. We base our attack on the observations given in [16] and extend its specifics to the ZIP structure. 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 $R_F(x)$ and $R_B(x)$ denote one round of forward and backward AES, respectively, each with an all-zero key.

1. Gather a set $\mathcal{D}$ of approximately $N = 2^{123.7}$ plaintext-ciphertext pairs.
2. For the distillation phase compute the vector $\mathbf{w} \in \mathbb{Z}^{2^{|k|}}$ with
$$w_j = \sum_{\substack{(x,y)\in\mathcal{D} \\ x|_{\mathcal{K}}=j}} (-1)^{\Gamma^{\oplus}\cdot y}$$
where $x|_{\mathcal{K}}$ denotes to gathering the bits of x given by the mask $\mathcal{K}$.
3. Split the integer j into two parts $j = j_F || j_B$ where j_F and j_B are designated for the key recovery cells of K_0 and K_4', respectively.
4. Compute the vector $\mathbf{v} \in \mathbb{Z}^{2^{|k|}}$ with entries
$$v_j = (-1)^{\Gamma^{\mathrm{f}}\cdot R_F(j_F \uparrow_{\mathcal{K}_F}) \,\oplus\, \Gamma^{\mathrm{b}}\cdot R_B(j_B \uparrow_{\mathcal{K}_B})}$$
where $j_F \uparrow_{\mathcal{K}_F}$ denotes scattering bits of j_F according to the mask $\mathcal{K}_F$.
5. Finally, with the FWHT, we can efficiently compute
$$\mathbf{q} = \frac{1}{2^{|k|}} \mathbf{H}_{2^{|k|}} \cdot \operatorname{diag}(\mathbf{H}_{2^{|k|}} \cdot \mathbf{v}) \cdot \mathbf{H}_{2^{|k|}} \cdot \mathbf{w}$$
where $\mathbf{H}_{2^{|k|}}$ stands for the $2^{|k|}$-dimensional Walsh matrix.
6. We recover our partial key k by selecting the biggest absolute entry $|q_k|$ in $\mathbf{q}$.

Data Complexity. Our attack uses a linear characteristic with a correlation of $c \approx 2^{-58.15}$, which is close to optimal. Based on the formulas by Selçuk [32], we choose 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, this steps 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 propose an $(r^{\mathrm{f}}, r^{\mathrm{b}}) = (10, 9)$-round linear key recovery attack on ZIP-GIFT. For this, first we find a $(r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}}) = (6, 6)$-round linear characteristic $((\mathit{\Gamma}^{\mathrm{f}}, \mathit{\Gamma}^{\mathrm{b}}) \rightarrow \mathit{\Gamma}^{\oplus})$ with correlation $c = 2^{-31}$, where

$$\begin{cases} \mathit{\Gamma}^{\mathrm{f}} = \texttt{0000 0c0c 0000 0000} \\ \mathit{\Gamma}^{\mathrm{b}} = \texttt{0040 0020 0000 8000} \\ \mathit{\Gamma}^{\oplus} = \texttt{a0a0 0000 a0a0 0000} \end{cases}$$

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.

The $(6, 6)$-round linear characteristic is finally exploited to mount a linear key recovery attack on ZIP-GIFT. More precisely, we prepend 4 and 3 rounds to the linear distinguisher. The key-recovery attack is depicted in Figure 6. Notably, the best linear key-recovery on GIFT-64 extends a 12-round linear trail to a 19-round attack by prepending 3 rounds and appending 4 rounds [34].

We use Matsui's Algorithm 2 combined with the partial-sum technique to perform $(10, 9)$-round key recovery attack on ZIP-GIFT. Given a collection of N plaintext-ciphertext pairs, we analyze for each key guess how often our linear approximation $S = \mathit{\Gamma}^{\mathrm{f}} \cdot X_4 \oplus \mathit{\Gamma}^{\mathrm{b}} \cdot X_{\text{-}3} \oplus \mathit{\Gamma}^{\oplus} \cdot C = 0$ holds. More precisely, we compute $\bigoplus_{i\in\mathcal{I}^{\mathrm{f}}} X_4[i] \oplus \bigoplus_{i\in\mathcal{I}^{\mathrm{b}}} X_{\text{-}3}[i] \bigoplus_{i\in\mathcal{I}^{\oplus}} C[i]$, where $\mathcal{I}^{\mathrm{f}} = \{34, 35, 42, 43\}$, $\mathcal{I}^{\mathrm{b}} = \{15, 37, 54\}$, and $\mathcal{I}^{\oplus} = \{21, 23, 29, 31, 53, 55, 61, 63\}$ for each key guess. Table 2 on page 36 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

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

   we compute the value of

   $$z_0 = \mathit{EX}_0[0\text{ - }19, 24\text{ - }35, 40\text{ - }51, 56\text{ - }63] \,\|\, \mathit{EX}_{\text{-}1}[17\text{ - }19, 24, 25, 27, 28] \,\|\, S_0,$$

   where $S_0 = \mathit{\Gamma}^{\mathrm{f}} \cdot X_4 \oplus \mathit{\Gamma}^{\oplus} \cdot C \oplus Y_{\text{-}2}[29] \oplus Y_{\text{-}2}[30]$ and update $\mathcal{L}_0[z_0]$ with $\mathcal{L}_0[z_0] + 1$. From algebraic normal form (ANF) of inverse S-box of GIFT, we

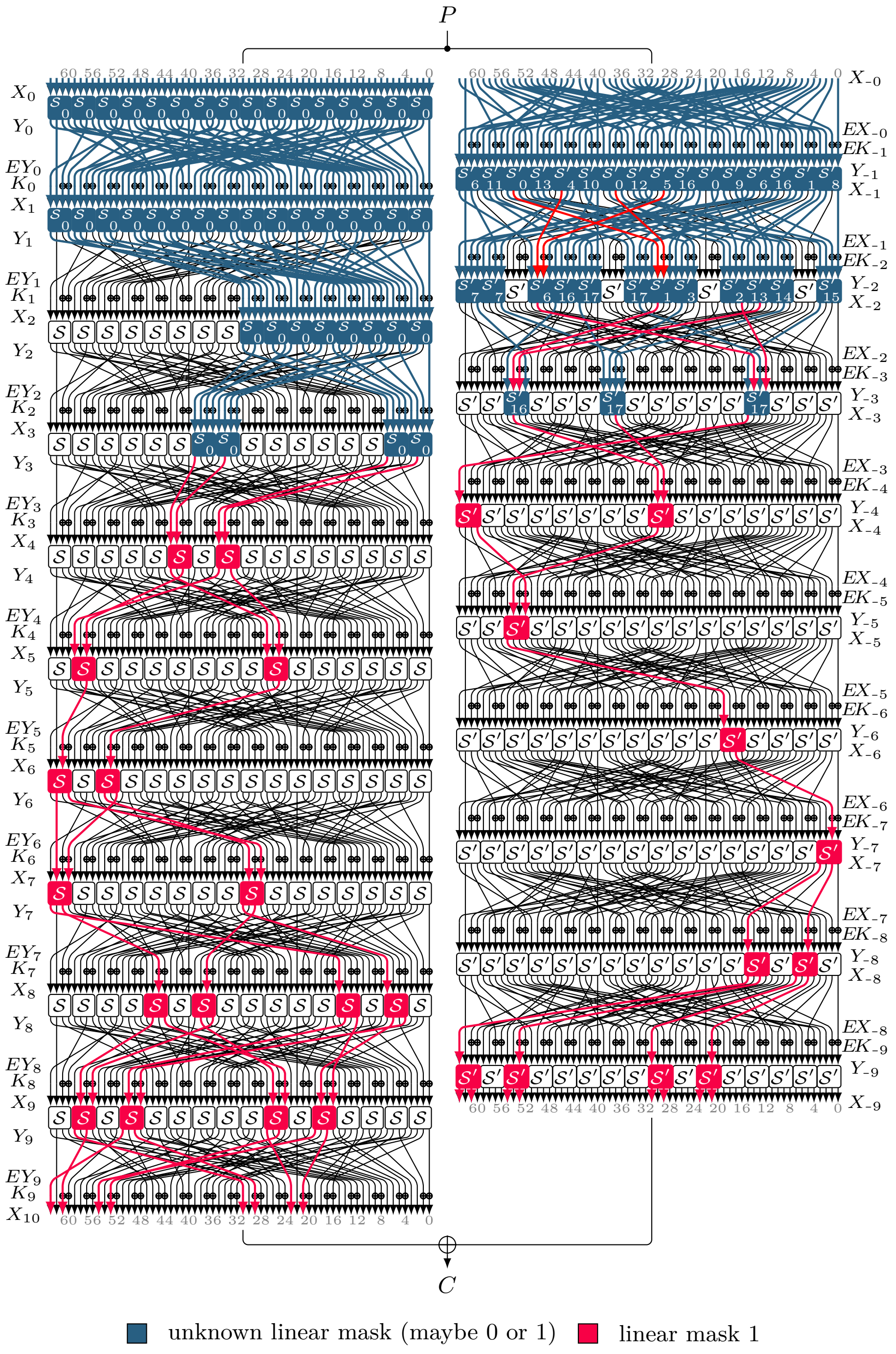


Fig. 6: Linear attack on (10, 9) rounds of ZIP-GIFT where the key recovery is performed on the first 4 rounds of the forward and the first 3 rounds of the backward branch

find out that $Y_{\text{-}2}[29]$, and $Y_{\text{-}2}[30]$ goes linearly $Y_{\text{-}3}[54]$, which is then linearly goes into $X_{\text{-}3}[54]$. Then, $Y_{\text{-}2}[29]$, and $Y_{\text{-}2}[30]$ can usually be merged with S and does not need to store these bits 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. Hence, every repetition of this step costs $47/304 \approx 2^{-2.7}$ encryptions. Consequently, the total complexity of this step is about $N \times 2^{62} \times 2^{-2.7}$.

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

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

where $S_1 = S_0 \oplus Y_{\text{-}3}[54]$. 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 $2^{62} \times 2^2 \times 2^{60} \times 2^{-7.2}$ 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. In Table 2 on page 36, these steps are summarized.

17. Initialize $\Sigma = 0$. For each 4-bit key $EK_{\text{-}2}[16, 17, 20, 21]$, compute

$$S' = S \oplus Y_{\text{-}2}[29] \oplus Y_{\text{-}2}[30] \oplus Y_{\text{-}2}[49] \oplus Y_{\text{-}2}[50] \oplus Y_{\text{-}3}[12] \oplus Y_{\text{-}3}[14] \oplus Y_{\text{-}3}[53] \oplus Y_{\text{-}3}[54].$$

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 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 [32,6], if we set

$$\theta = \sqrt{1/N} \cdot \Phi^{-1}(1 - 2^{-a-1})$$

then, the success probability will be:

$$\mathbb{P}_S \approx \Phi\left(c \cdot \sqrt{N} - \Phi^{-1}(1 - 2^{-a-1}) \cdot \sqrt{1 + N \cdot 2^{-n}}\right),$$

where $\Phi(\cdot)$ signifies the cumulative distribution function of the standard normal distribution. Here, we set $a = 2$, and $N = 2^{63}$, the success probability of this attack will be 60%. Here, the time complexity of Step 0 to Step 17 is about $2^{122.48}$ (from Table 2). Additionally, the time complexity of Step 18 is about $2^{128} \cdot 2^{-a} \cdot (1 + 2^{-64})$ $(10, 9)$-round of encryption. Overall, the attack requires approximately 2^{126} operations and uses about $2^{60.18}$ memory. Hence, the $(10, 9)$-round linear key-recovery attack runs in roughly 2^{126} time and achieves an estimated success probability of $\approx 60\%$.

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 [15] 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 construction $E^{\oplus} = E^{\mathrm{f}} \oplus E^{\mathrm{b}}$, 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 constructions complicate integral key recovery as rounds cannot be appended after the distinguisher and must be prepended instead. As observed in [15], the two branches seldom permit plaintext sets that fulfil the conditions for both integral distinguishers after some key inital recovery rounds. Next, we give integral distinguishers for ZIP-AES and ZIP-GIFT, present 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 Section 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 [22], 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 attack, described in the following, 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. Further, let Λ_I denote the set of 2^{16} AES states in which

the cells c_{02} and c_{12} are active and equal, the 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 want to guess the difference Δ_x of the cells c_{02} and c_{12} and the difference Δ_y of the cells c_{20} and c_{30}.
3. For all 2^{16} differences Δ_x and Δ_y do the following:
   (a) 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.
   (b) Build the set Λ^{b} as follows: $\Lambda^{\mathrm{b}} = \{\mathsf{MC}(x \oplus K_g) : x \in \Lambda_I\}$
   (c) For every pairwise combination of Λ^{f} and Λ^{b}, query the ciphertext sum:

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

   (d) If $S = 0$, then Δ_x and Δ_y corresponds to the relevant differences of K'_4.

Key-Dependent Behavior. Let us consider the forward and backward branches separately: For the forward branch, it is known that $\bigoplus_{x \in \Lambda^{\mathrm{f}}} \text{AES}_3(x) = 0$, which can also be observed in Figure 7. For the backward branch, suppose we correctly 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

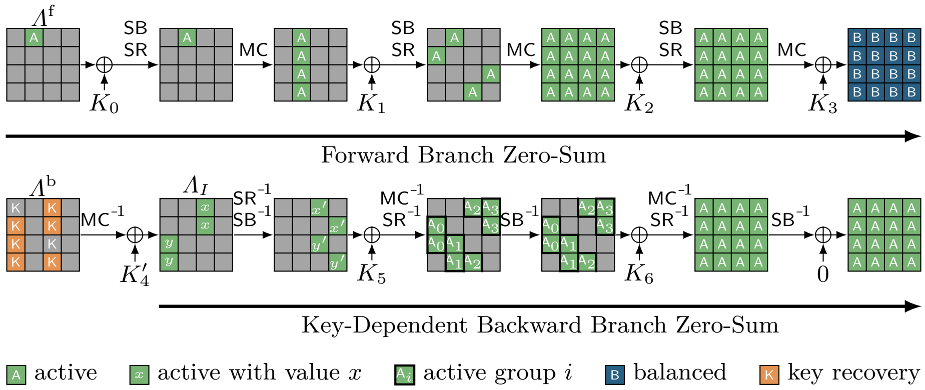


Fig. 7: Integral attack on (3,3)-rounds of ZIP-AES with one key recovery round in the backward branch. Active cells take all possible values across all states, active groups take all possible values for the whole group, other cells take the same value across all states, and the Xor-sum for balanced cells is zero.

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 backwards branch behavior experimentally.

To explain the combination of forward and backward branches, we observe

$$\begin{aligned} S &= \bigoplus_{P_1 \in \Lambda^{\mathrm{f}}} \bigoplus_{P_2 \in \Lambda^{\mathrm{b}}} \text{ZIP-AES}_{3,3}(P_1 \oplus P_2) \\ &= \bigoplus_{P_2 \in \Lambda^{\mathrm{b}}} \underbrace{\bigoplus_{P_1 \in \Lambda^{\mathrm{f}}} \text{AES}_3(P_1 \oplus P_2)}_{=0} \oplus \bigoplus_{P_1 \in \Lambda^{\mathrm{f}}} \underbrace{\bigoplus_{P_2 \in \Lambda^{\mathrm{b}}} \text{AES}_3^{-1}(P_1 \oplus P_2)}_{=0} . \end{aligned}$$

Note that the active cells 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, in our case 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 this possibility, we run the attack under a given key for all 2^{16} key guesses and observe a zero sum only for the correct key.

Runtime Evaluation. Our attack procedure recovers 16 bits by guessing 2^{16} key differences and, for every such key guess, encrypts the combined set $\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ of size 2^{24}. Therefore, we require $2^{16} \cdot 2^{24} = 2^{40}$ oracle encryptions to recover 16 bits of key material, leaving a brute force step of 2^{112}. Note that we can repeat the attack 4 times by shifting our zero sum to recover 64 key bits using 2^{42} data.

Difficulty of Higher Round Attacks. In the original paper on ZIP-AES, Flórez-Gutiérrez et al. [15] showed a distinguishing attack for (4,4)-rounds. However, based on our analysis, we believe that key integral recovery attacks for (4,3)-round or (4,4)-round are unlikely. Because of the ZIP structure, we need to ensure that no key-dependent cell overlaps with an active or another key-dependent cell in Λ^{f} and Λ^{b}. Suppose two such cells would overlap, then when we evaluate the combined XOR-sum of $\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}$ we would cancel out 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.

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 have to first 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 while marking all other plaintext bits as active, and then use the MILP model. Using this procedure we identify integral distinguishers up to $(9, 9)$-rounds for ZIP-GIFT as follows:

$$\bigoplus_{x \in \mathcal{P}} E^{\oplus}(x)[i] = 0, \ \forall i \in \{8, 16, 40, 48, 60\}.$$

Here, $E^{\oplus}$ denotes $(9, 9)$-rounds of ZIP-GIFT, and $\mathcal{P}$ is the set consisting of 64-bit vectors, where bit 60 is fixed, and the remaining bits span all 2^{63} possible values.

Difficulty of Integral Key Recovery Attacks on $(\boldsymbol{r}^{\mathbf{f}}, \boldsymbol{r}^{\mathbf{b}})$ Rounds of ZIP-GIFT with $\boldsymbol{r}^{\mathbf{f}}, \boldsymbol{r}^{\mathbf{b}} \geq \mathbf{9}$. For normal block ciphers, an integral distinguisher is usually turned into a key-recovery by appending rounds and guessing partial key bits to decrypt the ciphertext. This strategy fails for ZIP-GIFT because 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 a $(r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}})$-round integral distinguisher for ZIP-GIFT. Let $\mathcal{I}^{\mathrm{f}}, \mathcal{I}^{\mathrm{b}} \subseteq \{0, 1, \ldots, 63\}$ denote the active bit positions at the input of the distinguishers for the forward and backward branches, respectively. Moreover, Λ^{f} (resp. Λ^{b}) denotes the corresponding set of values where the bits located at $\mathcal{I}^{\mathrm{f}}$ (resp. $\mathcal{I}^{\mathrm{b}}$) take all possible $2^{|\mathcal{I}^{\mathrm{f}}|}$ (resp. $2^{|\mathcal{I}^{\mathrm{b}}|}$) values. Then,

$$\bigoplus_{x \in \Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}}} (E^{\mathrm{f}}_{\mathrm{dist}} \oplus E^{\mathrm{b}}_{\mathrm{dist}})(x) = 0$$

where $E^{\mathrm{f}}_{\mathrm{dist}} \oplus E^{\mathrm{b}}_{\mathrm{dist}}$ denote $(r^{\mathrm{f}}_{\mathrm{dist}}, r^{\mathrm{b}}_{\mathrm{dist}})$ rounds of ZIP-GIFT. 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 the cardinalities of the index sets $\mathcal{I}^{\mathrm{f}}$ and $\mathcal{I}^{\mathrm{b}}$ are large, then the common plaintext set $\mathcal{P}$ will include all active bits. Consequently, if the common plaintext set equals the full 2^{64} input space, the ciphertext sum is always zero and contains no key-dependent information, preventing a valid integral key-recovery attack. Even if one manages to find a suitable common plaintext set $\mathcal{P}$, when querying $\mathcal{P}$, we would indeed observe a zero-sum, however, partitioning the space to perform key recovery would not be possible: If we guess all involved key bits in the forward branch, for example, and partition the set of plaintexts according to the input for the forward branch integral distinguisher, we would observe essentially random inputs for the distinguisher of the backwards branch. This difficulty stems from the fundamental fact that we are trying to perform two chosen-plaintext attacks simultaneously. To overcome this difficulty, we present a dedicated attack based on a property of 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 one round to one or both branches, the common plaintext set becomes the full 2^{64} input space; consequently the ciphertext sum is always zero, eliminating any key-dependency.

Integral Key-Recovery Attack on $(8, 8)$ Rounds of ZIP-GIFT. In this section, we present a $(8, 8)$-round integral attack on ZIP-GIFT, while based on a 9-round integral distinguisher on GIFT-64, the authors in [2], presented 14-round integral key recovery attack on GIFT-64. Figure 8 illustrates the $(8, 8)$-round attack in ZIP-GIFT. For this, we construct a 7-round backward branch zero-sum distinguisher, prepend one round of key recovery, and combine it with a 8-round forward branch zero-sum distinguisher. More precisely, we utilize

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

where $E^{\mathrm{f}}_{\mathrm{dist}}$ and $E^{\mathrm{b}}_{\mathrm{dist}}$ denote 8 and 7 forward and backward rounds, respectively. Moreover, let $\Lambda^{\mathrm{f}}_{\mathrm{dist}}$ be the set of 2^{61} 64-bit vectors with bits 0, 17, and 34 fixed. Similarly, let $\Lambda^{\mathrm{b}}_{\mathrm{dist}}$ be the set of 2^{61} 64-bit vectors with bit 1, 2, and 3 fixed. The set of balanced bits at the output is $\mathcal{I}^{\oplus} = \{0, 4, 8, 12, 16, 20, 24, 28, 40, 44, 48, 56\}$. We mount our $(8, 8)$-round key-recovery attack as follows:

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

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

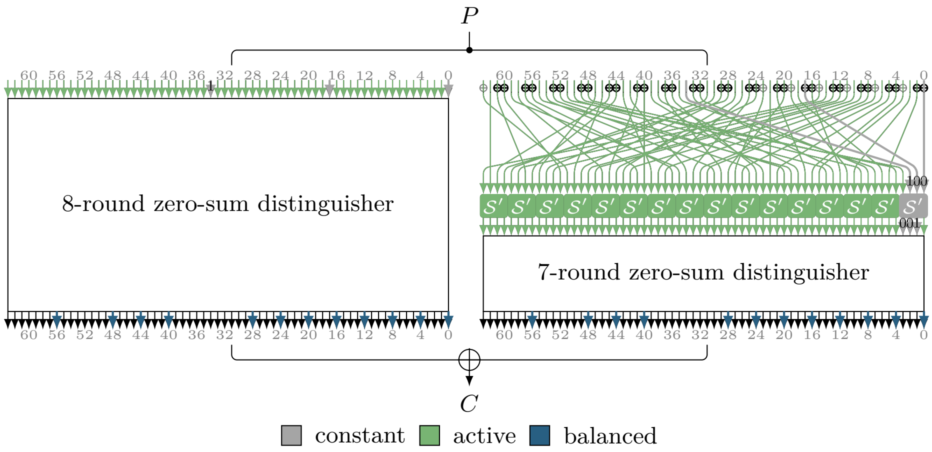


Fig. 8: Integral key-recovery attack on $(8, 8)$-round ZIP-GIFT with one backward-branch key-recovery round. Here, the activity pattern of the input set to the 7-round backward branch for the correct key guess is shown.

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

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

2. Guess bits 0 and 17 of $K_{\text{-}1}$ and store in K_g:
   (a) Decrypt each element of Λ^{b} by one round to construct the plaintext set:

   $$\mathcal{P} = \left\{P_{64}(\mathsf{SubCells}(x)) \oplus K_g | x \in \Lambda^{\mathrm{b}}\right\}$$

   (b) Query the encryption oracle and compute the output sum:

   $$S = \bigoplus_{x \in \mathcal{P}} \text{ZIP-GIFT}_{8,8}(x).$$

   (c) If for $S[i] = 0$ for the 12 bits $i \in \mathcal{I}^{\oplus}$, then our key guess K_g is correct.
3. Recover the remaining 126 bits using brute force search.

Key-Dependent Behavior. From the attack procedure, we see that in the backward branch, for the correct key guess, $\{R^{\text{-}1}(x) \,|\, x \in \mathcal{P}\} = \Lambda^{\mathrm{b}}$ 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 $\{R^{\text{-}1}(x) \mid x \in \mathcal{P}\} \neq \Lambda^{\mathrm{b}}$ 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 for those affine spaces. Since we cannot directly formulate an affine space, we analyze the function $R^{\text{-}7} \circ L$, 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 $L^{\text{-}1}$ transforms the affine space into a cube of the form `CCC*`. By analyzing the input cube for the different functions $R^{\text{-}7} \circ L$, 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 two key bits are involved, there is only a negligible effect on the 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 most important findings for differential, linear, and integral key recovery attacks on ZIP ciphers.

Differential key recovery. 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 [15, 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 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 key recovery. 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 well-known 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 key recovery. 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 that permit some 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 after key addition. An interesting direction for future work could be applying cube attacks to ZIP-ciphers, as they combine key recovery and distinguisher.

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 resistance of ZIP ciphers against key recovery attacks 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 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

1. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency PRF. IACR Transactions on Symmetric Cryptology **2021**(1), 37–77 (2021). https://doi.org/10.46586/tosc.v2021.i1.37-77
2. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present – towards reaching the limit of lightweight encryption. In: CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_16
3. Bellare, M., Impagliazzo, R.: 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), http://eprint.iacr.org/1999/024
4. Bellare, M., Krovetz, T., Rogaway, P.: Luby-rackoff backwards: Increasing security by making block ciphers non-invertible. In: EUROCRYPT '98. LNCS, vol. 1403, pp. 266–280. Springer (1998). https://doi.org/10.1007/bfb0054132
5. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology **4**(1), 3–72 (1991). https://doi.org/10.1007/bf00630563
6. Blondeau, C., Nyberg, K.: Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. Des. Codes Cryptogr. **82**(1-2), 319–349 (2017). https://doi.org/10.1007/s10623-016-0268-6

7. Boura, C., David, N., Derbez, P., Boissier, R.H., Naya-Plasencia, M.: A generic algorithm for efficient key recovery in differential attacks - and its associated tool. In: EUROCRYPT 2024. LNCS, vol. 14651, pp. 217–248. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_8
8. Collard, B., Standaert, F.X., Quisquater, J.J.: Improving the time complexity of Matsui's linear cryptanalysis. In: ICISC 2007. LNCS, vol. 4817, pp. 77–88. Springer (2007). https://doi.org/10.1007/978-3-540-76788-6_7
9. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: FSE '97. LNCS, vol. 1267, pp. 149–165. Springer (1997). https://doi.org/10.1007/bfb0052343
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES – The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002). https://doi.org/10.1007/978-3-662-04722-4
11. Dinur, I.: Tight indistinguishability bounds for the XOR of independent random permutations by fourier analysis. In: EUROCRYPT 2024. LNCS, vol. 14651, pp. 33–62. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_2
12. Eichlseder, M., Nageler, M., Primas, R.: Analyzing the linear keystream biases in AEGIS. IACR Transactions on Symmetric Cryptology **2019**(4), 348–368 (2019). https://doi.org/10.13154/tosc.v2019.i4.348-368
13. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D.A., Whiting, D.: Improved cryptanalysis of Rijndael. In: FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer (2000). https://doi.org/10.1007/3-540-44706-7_15
14. Flajolet, P., Fusy, É., Gandouet, O., Meunier, F.: Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. Discrete Mathematics and Theoretical Computer Science **AOFA**, 137–156 (2007). https://doi.org/10.46298/dmtcs.3545
15. Flórez-Gutiérrez, A., Grassi, L., Leander, G., Sibleyras, F., Todo, Y.: General practical cryptanalysis of the sum of round-reduced block ciphers and ZIP-AES. In: ASIACRYPT 2024. LNCS, vol. 15492, pp. 280–311. Springer (2024). https://doi.org/10.1007/978-981-96-0947-5_10
16. Flórez-Gutiérrez, A., Naya-Plasencia, M.: Improving key-recovery in linear attacks: Application to 28-round PRESENT. In: EUROCRYPT 2020. LNCS, vol. 12105, pp. 221–249. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_9
17. Hadipour, H., Todo, Y., Rahman, M., Eichlseder, M., Anand, R., Isobe, T.: Turning multiple key-dependent attacks into universal attacks. Cryptology ePrint Archive, Paper 2025/1996 (2025), https://eprint.iacr.org/2025/1996
18. Heule, S., Nunkesser, M., Hall, A.: Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In: EDBT '13. pp. 683–692. ACM (2013). https://doi.org/10.1145/2452376.2452456
19. Hou, S., Wu, B., Wang, S., Guo, H., Lin, D.: Truncated differential attacks on symmetric primitives with linear key schedule: WARP and orthros. Comput. J. **67**(4), 1483–1500 (2024). https://doi.org/10.1093/comjnl/bxad075
20. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_15
21. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of nlfsr-based cryptosystems. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_8
22. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer (2007). https://doi.org/10.1007/978-3-540-76900-2_19

23. Lai, X.: Higher Order Derivatives and Differential Cryptanalysis, pp. 227–233. Springer (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
24. Li, L., Wu, W., Zheng, Y., Zhang, L.: The relationship between the construction and solution of the MILP models and applications. IACR Cryptology ePrint Archive, Paper 2019/049 (2019), https://eprint.iacr.org/2019/049
25. Li, M., Sun, L., Wang, M.: Automated key recovery attacks on round-reduced orthros. In: AFRICACRYPT 2022. LNCS, vol. 13503, pp. 189–213. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-17433-9_9
26. Lucks, S.: The sum of prps is a secure PRF. In: EUROCRYPT 2000. LNCS, vol. 1807, pp. 470–484. Springer (2000). https://doi.org/10.1007/3-540-45539-6_34
27. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT '93. LNCS, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33
28. Nageler, M., Ghosh, S., Jüttler, M., Eichlseder, M.: Autodiver: Automatically verifying differential characteristics and learning key conditions. IACR Transactions on Symmetric Cryptology **2025**(1), 471–514 (2025). https://doi.org/10.46586/tosc.v2025.i1.471-514
29. Patarin, J.: Introduction to mirror theory: Analysis of systems of linear equalities and linear non equalities for cryptography. IACR Cryptology ePrint Archive, Paper 2010/287 (2010), http://eprint.iacr.org/2010/287
30. Peyrin, T., Tan, Q.Q.: Mind your path: On (key) dependencies in differential characteristics. IACR Transactions on Symmetric Cryptology **2022**(4), 179–207 (2022). https://doi.org/10.46586/tosc.v2022.i4.179-207
31. Qiao, K., wei Sun, S., Hu, L., Wang, X.: Differential security evaluation of simeck with dynamic key-guessing techniques. Differential Security Evaluation of Simeck with Dynamic Key-guessing Techniques (2016)
32. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology **21**(1), 131–147 (2008). https://doi.org/10.1007/s00145-007-9013-7
33. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. IACR Transactions on Symmetric Cryptology **2021**(1), 269–315 (2021). https://doi.org/10.46586/tosc.v2021.i1.269-315
34. Sun, L., Wang, W., Wang, M.: Improved attacks on GIFT-64. In: Selected Areas in Cryptography. pp. 246–265. Springer International Publishing, Cham (2022)
35. Taka, K., Ishikawa, T., Sakamoto, K., Isobe, T.: An efficient strategy to construct a better differential on multiple-branch-based designs: Application to orthros. In: CT-RSA 2023. LNCS, vol. 13871, pp. 277–304. Springer (2023). https://doi.org/10.1007/978-3-031-30872-7_11
36. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_12
37. Todo, Y., Aoki, K.: FFT key recovery for integral attack. In: CANS 2014. LNCS, vol. 8813, pp. 64–81. Springer (2014). https://doi.org/10.1007/978-3-319-12280-9_5

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, prepending 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, 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:
   $$\Lambda^{\mathrm{f}} = \left\{\mathrm{SB}^{-1}\left(\mathrm{SR}^{-1}\left(\mathrm{MC}^{-1}(x)\right)\right) \oplus K : x \in \Lambda_I\right\}.$$
   (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
   $$S := \bigoplus_{x \in (\Lambda^{\mathrm{f}} \oplus \Lambda^{\mathrm{b}})} \text{ZIP-AES}_{4,2}(x).$$

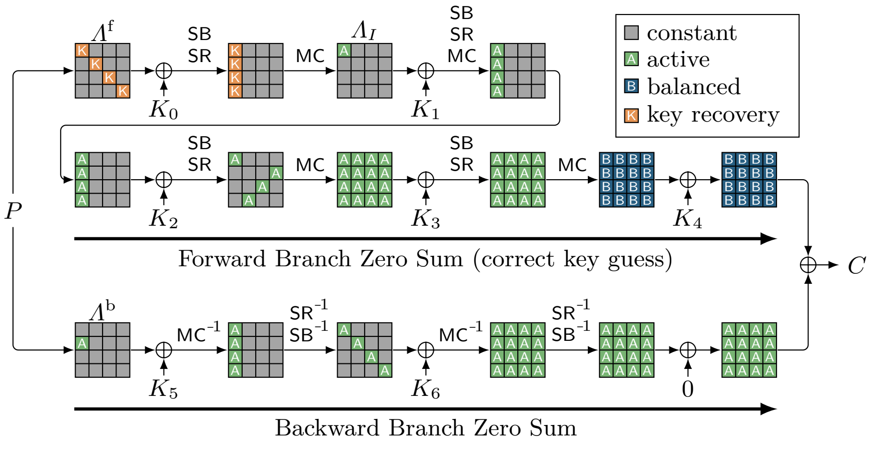


Fig. 9: Integral attack on (4,2)-rounds of ZIP-AES with one key recovery round in the forward branch.

(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

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

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. As a slight improvement, 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.

With additional memory overhead, we can further improve the attack. 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. The runtime of this approach is $2^{40} + 2^{40} = 2^{41}$ but requires 64 GiB of memory.

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

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

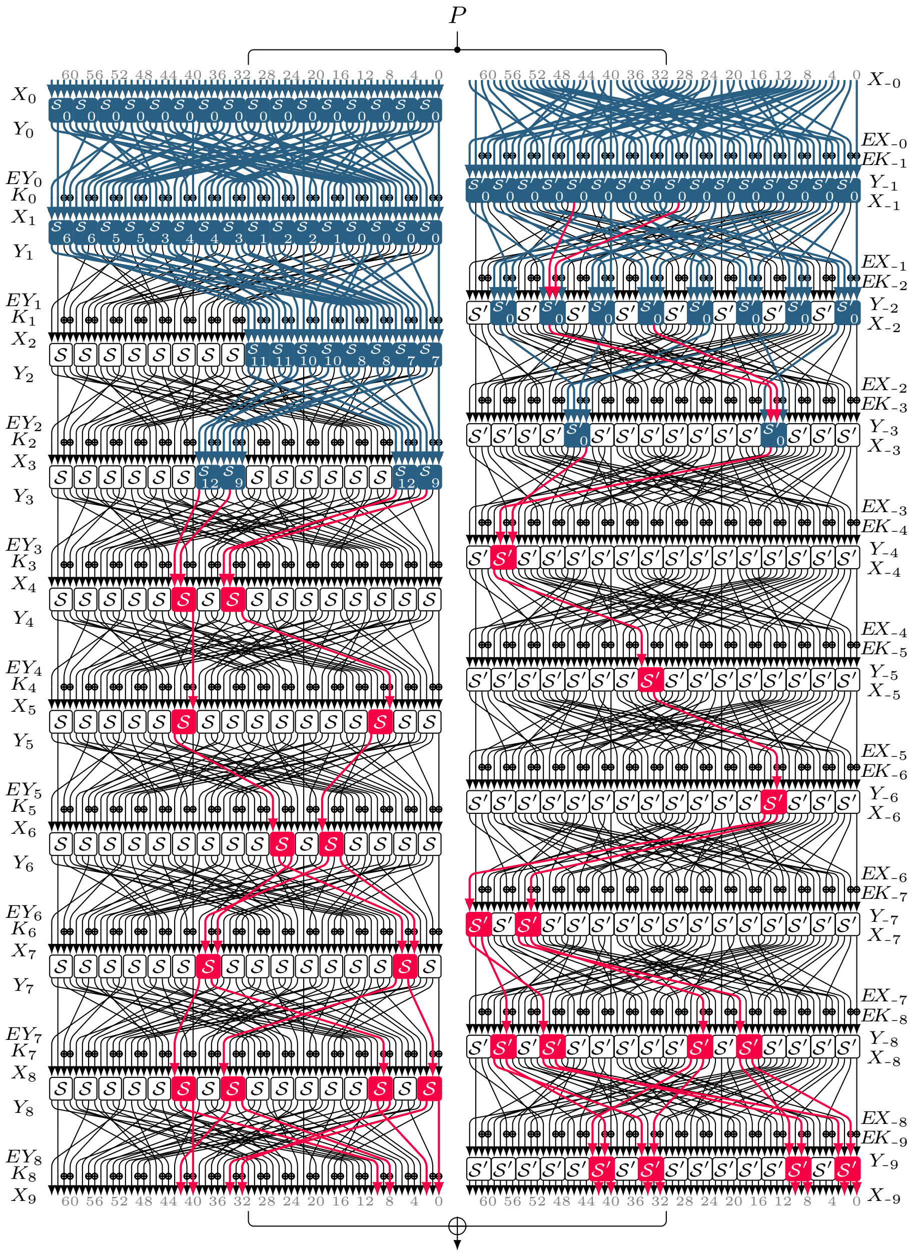
$$S = \bigoplus_{i \in \mathcal{I}^{\mathrm{f}}} X_4[i] \ \oplus \ \bigoplus_{i \in \mathcal{I}^{\mathrm{b}}} X_{-3}[i] \ \oplus \ \bigoplus_{i \in \mathcal{I}^{\oplus}} C[i]$$

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	$K_0[0\,\text{-}\,31]$, $K_1[0\,\text{-}\,15]$, $K_2[0..3, 16..19]$, $EK_{-1}[10, 11, 18, 19, 26, 27]$, $EK_{-2}[15]$	$2^{62+63-2.7}$	$EX_0[0..19, 24..35, 40..51, 56..63]$, $EX_{-1}[17..19, 24, 25, 27, 28]$, 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}
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		$2^{122.4}$		$2^{60.18}$

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


P
C
60 56 52 48 44 40 36 32 28 24 20 16 12 8 4 0
X_0 Y_0 EY_0 K_0 X_1 Y_1 EY_1 K_1 X_2 Y_2 EY_2 K_2 X_3 Y_3 EY_3 K_3 X_4 Y_4 EY_4 K_4 X_5 Y_5 EY_5 K_5 X_6 Y_6 EY_6 K_6 X_7 Y_7 EY_7 K_7 X_8 Y_8 EY_8 K_8 X_9
X_{-0} EX_{-0} EK_{-1} Y_{-1} X_{-1} EX_{-1} EK_{-2} Y_{-2} X_{-2} EX_{-2} EK_{-3} Y_{-3} X_{-3} EX_{-3} EK_{-4} Y_{-4} X_{-4} EX_{-4} EK_{-5} Y_{-5} X_{-5} EX_{-5} EK_{-6} Y_{-6} X_{-6} EX_{-6} EK_{-7} Y_{-7} X_{-7} EX_{-7} EK_{-8} Y_{-8} X_{-8} EX_{-8} EK_{-9} Y_{-9} X_{-9}
S
S′

🟦 unknown linear mask (maybe 0 or 1) 🟥 linear mask 1

Fig. 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 $\mathcal{I}^{\oplus} = \{0, 2, 8, 10, 32, 34, 40, 42\}$. $S_1 : \bigoplus_{i \in \mathcal{I}^{\mathrm{b}}} X_{\text{-}3}[i] \oplus \bigoplus_{i \in \mathcal{I}^{\oplus}} C[i]$. $\mathcal{K}_{\text{-}2} : \{0, 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 28, 29\}$. $EY_i : P_{64}(Y_i)$, where $(0 \leq i \leq 2)$. $EK_j : P_{64}^{-1}(K_j)$, where $(-3 \leq j \leq -1)$. gray: redundant key bits

Step	Key	Time Complexity	Stored Texts	Memory
0	$EK_{\text{-}1}[0 - 31], EK_{\text{-}2}[\mathcal{K}_{-2}]$, $EK_{\text{-}3}[6, 22, 23], K_0[0 - 7]$, $EK_{\text{-}3}[5], EK_{\text{-}2}[25]$	$2^{58} \times 2^{62}$	$EY_0[16 - 63], EY_1[0 - 3, 16 - 19], \mathcal{S}_1$	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)		$2^{57.09}$