Improving the Rectangle Attack on GIFT-64

Yincen Chen[1], Nana Zhang[2,3], Xuanyu Liang[1], Ling Song[1], Qianqian Yang[2], and Zhuohui Feng[1]

[1] College of Cyber Security, Jinan University, Guangzhou, 510632, China
icsnow98@gmail.com, xyljnu@gmail.com,
songling.qs@gmail.com, hhfzhfzh@163.com
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
zhangnana@iie.ac.cn, yangqianqian@iie.ac.cn
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. GIFT is a family of lightweight block ciphers based on SPN structure and composed of two versions named GIFT-64 and GIFT-128. In this paper, we reevaluate the security of GIFT-64 against the rectangle attack under the related-key setting. Investigating the previous rectangle key recovery attack on GIFT-64, we obtain the core idea of improving the attack——trading off the time complexity of each attack phase. We flexibly guess part of the involved subkey bits to balance the time cost of each phase so that the overall time complexity of the attack is reduced. Moreover, the reused subkey bits are identified according to the linear key schedule of GIFT-64 and bring additional advantages for our attacks. Furthermore, we incorporate the above ideas and propose a dedicated MILP model for finding the best rectangle key recovery attack on GIFT-64. As a result, we get the improved rectangle attacks on 26-round GIFT-64, which are the best attacks on it in terms of time complexity so far.

Keywords: symmetric cryptography · GIFT-64 · rectangle attack · key recovery attack · related-key scenario · key guessing strategy

1 Introduction

Accompanied by the momentous expansion in emerging ubiquitous technologies, securing resource-limited devices has become increasingly important. Lightweight block ciphers came into being in such a situation, which have more advantages in terms of cost, speed, power, and execution time than traditional block ciphers, but still provide a sufficiently high safety margin for resource-limited devices. Well-known lightweight block ciphers include SIMON [4], SPECK [4], PRESENT [8], GIFT [3], etc.

Over the past three decades, researchers have committed to researching algorithms to efficiently and accurately evaluate the security of block ciphers. In 1990, Biham and Shamir proposed differential cryptanalysis [6], which tracks the difference between a pair of inputs to outputs. One of the essential steps of differential cryptanalysis is to find a high-probability differential trail over the

target cipher. However, this goal is hard to achieve when the cipher contains many rounds. The boomerang attack [29] is an extension of differential cryptanalysis, which combines two short differential trails to get a long trail with a high probability. The rectangle attack [5] is a variant of the boomerang attack. The boomerang attack requires chosen plaintexts and chosen ciphertexts, while the rectangle attack only needs to choose plaintexts. Besides, the rectangle attack considers as many differences as possible in the middle to estimate the probability more accurately. The boomerang and rectangle attacks have been applied to many ciphers, and many good results have been obtained. For example, Biryukov *et al.* [7] put forward the rectangle attack on full AES-192 and AES-256, and Derbez *et al.* proposed the boomerang attack on full AES-192 in [13].

In recent years, many strategies emerged to mount key recovery attacks as efficiently as possible for the rectangle attack, such as [14, 25, 32]. Song *et al.* proposed the most efficient and generic rectangle key recovery algorithm at ASIACRYPT 2022 [25]. This algorithm supports flexible key guessing strategies and is compatible with all the previous rectangle key recovery algorithms. By trading off the overall complexity, Song *et al.* obtained the optimal results of rectangle key recovery attacks on a series of block ciphers.

GIFT is a family of SPN-based lightweight block ciphers proposed by Banik *et al.* at CHES'17 [3]. It is composed of two versions named GIFT-64 and GIFT-128, where the block sizes are 64 bits and 128 bits, and the numbers of rounds are 28 and 40, respectively. The key lengths of GIFT-64 and GIFT-128 are both 128 bits. As the inheritor of PRESENT [8], GIFT mends its weak points and achieves efficiency and security improvements. Because of the comprehensive treatment of the linear layer and the S-box, GIFT receives excellent performance in hardware and software implementations and has become one of the most energy-efficient ciphers. Benefiting from these advantages, GIFT plays the role of the underlying primitives of many lightweight authenticated encryption schemes, such as GIFT-COFB [2], HyENA [10], LOTUS-AEAD and LOCUS-AEAD [9], and SUNDAE-GIFT [1]. Notably, GIFT-COFB is one of the final round finalists of the NIST Lightweight Cryptography standardization project [4]. Thus, the security evaluation of GIFT is of great significance.

Previous attacks on GIFT-64. GIFT has attracted the attention of many researchers since its publication and has been the subject of many cryptanalyses. The best result of the meet-in-the-middle attack is from [21], which attacks 15-round GIFT-64 under the single key scenario. Sun *et al.* proposed the best linear attack on GIFT-64 at present, which is a linear attack on 19-round GIFT-64 in [27]. The most efficient differential analysis for GIFT-64 under the related-key scenario is currently the 26-round differential attack of Sun *et al.* [26]. Dong *et al.* proposed the most efficient attack on GIFT-64 for the moment, which is a 26-round rectangle key recovery attack we are interested in [14]. We summarize the state-of-the-art attacks against GIFT-64 in Table 1, where RK and SK denote

[4] https://csrc.nist.gov/projects/lightweight-cryptography.

related-key and single-key settings, respectively, and enc. and m.a. represent time complexity in units of encryption and memory access.

This paper focuses on the rectangle key recovery attack against GIFT-64. In 2019, Chen *et al.* executed the 23-round rectangle key recovery attack on GIFT-64 based on a 19-round related-key boomerang distinguisher [11]. Later, Zhao *et al.* [32] expanded this attack to 24 rounds with a more efficient key-guessing strategy. In 2020, Ji *et al.* [16] proposed the 20-round related-key boomerang distinguisher. Based on this distinguisher, they also proposed a 25-round related-key rectangle key recovery attack on GIFT-64 using Zhao *et al.*'s strategy in [16]. At EUROCRYPT 2022, Dong *et al.* [14] further improved the key guessing strategy and extended the attack of Ji *et al.* to 26 rounds, resulting in the most effective rectangle key recovery attack of GIFT-64 so far.

Table 1: Summary of relevant analysis results of GIFT-64

Method	Setting	Round	Time	Data	Memory	Source
Integral	**SK**	14	$2^{96.00}$ enc.	$2^{63.00}$	$2^{63.00}$	[3]
MITM	**SK**	15	$2^{112.00}$ enc.	$2^{64.00}$	$2^{16.00}$	[21]
Linear	**SK**	19	$2^{127.11}$ enc.	$2^{62.96}$	$2^{60.00}$	[27]
Boomerang	**RK**	23	$2^{126.60}$ enc.	$2^{63.30}$	-	[19]
Differential	**RK**	26	$2^{123.23}$ enc.	$2^{60.96}$	$2^{102.86}$	[26]
Rectangle	**RK**	23	$2^{107.00}$ m.a.	$2^{60.00}$	$2^{60.00}$	[11]
	RK	24	$2^{91.58}$ enc.	$2^{60.00}$	$2^{60.32}$	[32]
	RK	25	$2^{120.92}$ enc.	$2^{63.78}$	$2^{64.10}$	[16]
	RK	26	$2^{122.78}$ enc.	$2^{63.78}$	$2^{63.78}$	[14]
	RK	26	$2^{121.75}$ enc.	$2^{62.715}$	$2^{62.715}$	[18]
	RK	26	$2^{112.07}$ enc.	$2^{63.79}$	$2^{63.79}$	[31]
	RK	26	$\mathbf{2^{110.06}}$ **enc.** and $\mathbf{2^{115.8}}$**m.a.**	$2^{63.78}$	$2^{64.36}$	Sec. 3
	RK	26	$\mathbf{2^{111.51}}$ **enc.** and $\mathbf{2^{115.78}}$ **m.a.**	$2^{63.78}$	$2^{67.8}$	Sec. 3

Our contributions. We investigate the previous rectangle attacks on GIFT-64 and find that the time complexities of different attack phases are not balanced. Inspired by the work of Song *et al.* [25], we study how to find a better strategy for the rectangle key recovery attack on GIFT-64 to trade off the complexity of each phase.

GIFT has a bit-wise linear layer and a bit-wise key schedule. We carefully study each component of GIFT and, for the first time, apply the generic rectangle key recovery algorithm [25] to such ciphers. For ciphers like GIFT, which mostly have bit-wise operations, finding the best key-guessing strategy is much more sophisticated than for cell-wise ciphers. In the attack on GIFT-64, we carefully analyzed the key schedule and identified all the reused key bits. To find the best attack for a given rectangle distinguisher, we build a MILP model in which all possible key guessing strategies are allowed and minimize the overall time complexity. As a result, we improve the rectangle attacks on 26-round GIFT-64

with new key-guessing strategies. Our attacks on GIFT-64 are better than the previous rectangle key recovery attacks and are the best attacks on GIFT-64 in terms of time complexity to date. The comparison of our attacks with previous works is shown in Table 1. Apart from this, we also study the rectangle key recovery attack on GIFT-128 and eventually reduce the complexity of the attack in [16] by a factor of 2^2. We explain the attack in the Appendix B.

It's worth noting that Yu *et al.* [31] proposed remarkable cryptanalysis of GIFT-64 concurrently (available on-line on The Computer Journal on 14th July 2023). They constructed an automatic search model which treats the distinguisher and the key recovery phase as a whole for GIFT. Taking the linear key schedule into account, they also discovered a new boomerang distinguisher of GIFT-64. The complexity of their 26-round rectangle key recovery attack on GIFT-64 is $(T, D, M) = (2^{112.07}$ enc., $2^{63.79}, 2^{63.79})$.

Organization. The rest of the paper is organized as follows. In Sec. 2, we introduce the structure of GIFT-64 and review the rectangle attack and the key recovery algorithm. In Sec. 3, we propose the dedicated MILP model for finding the best rectangle key recovery attack on GIFT-64 and describe in detail the rectangle key recovery attack based on a new key guessing strategy. Sec. 4. concludes the paper.

2 Preliminary

2.1 Description of GIFT-64

GIFT is a block cipher with Substitution-Permutation-Network, which Banik *et al.* proposed at CHES' 2017 [3]. According to the 64-bit and 128-bit block sizes, GIFT has two versions, GIFT-64 and GIFT-128, with round numbers 28 and 40, respectively. Both versions of GIFT use a 128-bit master key. We only focus on GIFT-64 in this paper. r in this subsection represents the number of rounds, where $r \in \{1, 2, ..., 28\}$.

Round Function. The round function of GIFT-64 consists of three operations. For convenience, we consider the 64-bit round state as 16 4-bit nibbles. The three operations of the round function are as follows:

1. **SubCells**: Nonlinear S-box substitutions are applied to each nibble, as is shown in Table 2. Denote X_r and Y_r as the inputs and outputs of the 16 S-boxes in the round r.

Table 2: The S-box of GIFT

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$GS(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

2. **PermBits:** For each bit of input, linear bit permutation $b_{P(i)} \leftarrow b_i$, $\forall i \in \{0, 1, ..., 63\}$ is applied. The permutation $P(i)$ is shown in Table 3. Denote the state which is transformed from Y_r by PermBits in round r as Z_r.

Table 3: Specifications of GIFT-64 Bit Permutation

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	17	34	51	48	1	18	35	32	49	2	19	16	33	50	3
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	21	38	55	52	5	22	39	36	53	6	23	20	37	54	7
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	25	42	59	56	9	26	43	40	57	10	27	24	41	58	11
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	29	46	63	60	13	30	47	44	61	14	31	28	45	62	15

3. **AddRoundKey:** This step consists of adding the round key and round constants. At each round, a 32-bit round key is obtained from the master key. Denote round key as $RK_r = U||V = u_{15}, ..., u_0||v_{15}, ..., v_0$. For each round, U and V are XORed with the cipher state, *i.e.*, $b_{4i+1} \leftarrow b_{4i+1} \oplus u_i, b_{4i} \leftarrow b_{4i} \oplus v_i, \forall i \in \{0, ..., 15\}$. A single bit "1" and a 6-bit constant $C = c_5c_4c_3c_2c_1c_0$ are added to each state at bit position $63, 23, 19, 15, 11, 7, 3$ respectively, *i.e.*, $b_{63} \leftarrow b_{63} \oplus 1, b_{23} \leftarrow b_{23} \oplus c_5, b_{19} \leftarrow b_{19} \oplus c_4, b_{15} \leftarrow b_{15} \oplus c_3, b_{11} \leftarrow b_{11} \oplus c_2, b_7 \leftarrow b_7 \oplus c_1, b_3 \leftarrow b_3 \oplus c_0$. RK_r is added to the state Z_r in each round.

Key Schedule. Split the master key K into 8 16-bit subkeys $k_7||k_6||...||k_1||k_0 \leftarrow K$. For each round, the round key consists of the last two significant subkeys, and then, the key state is updated following $k_7||k_6||...||k_1||k_0 \leftarrow k_1 \ggg 2||k_0 \ggg 12||...||k_3||k_2$, where $\ggg i$ is an i-bit right rotation within a 16-bit word.

2.2 The Rectangle Attack

In this subsection, we review the rectangle attack and the generic rectangle key recovery algorithm [25] and explain the notations used in this paper.

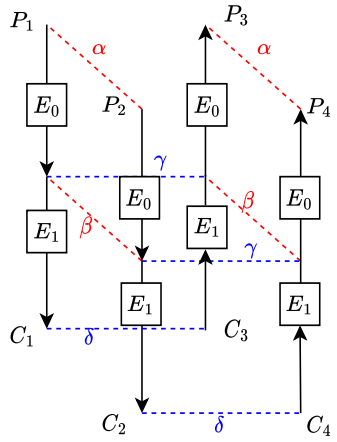

Fig. 1: Boomerang distinguisher

Before introducing the rectangle attack, we must first review the boomerang attack. The boomerang attack was proposed by Wanger [29] in 1999, which is an adaptive chosen plaintext/ciphertext attack. As is illustrated in Fig. 1, it regards the target cipher as a composition of two sub-ciphers E_0 and E_1, *i.e.*, $E = E_0 \circ E_1$. The differential trail $\alpha \to \beta$ travels in E_0 with probability p, and the differential trail $\gamma \to \delta$ travels in E_1 with probability q, which composes the boomerang distinguisher with the probability p^2q^2. In [17], Kelsey *et al.* developed a chosen-plaintext variant and formed the amplified boomerang attack with probability $p^2q^2 2^{-n}$, where n is the size of each block. The rectangle attack [5] improves the amplified boomerang attack, which estimates the probability more accurately by considering as many differences as possible in the middle. The probability of a rectangle distinguisher is $2^{-n}\hat{p}^2\hat{q}^2$, where $\hat{p} = \sqrt{\Sigma_i \mathrm{Pr}^2(\alpha \to \beta_i)}$, $\hat{q} = \sqrt{\Sigma_j \mathrm{Pr}^2(\gamma_j \to \delta)}$. Later, researchers discovered many methods to compute the probability more accurately and proposed an innovative tool named boomerang connectivity table (BCT) [12, 24]. The process of the rectangle attack consists of two steps:

1. Choose plaintexts P_1, P_2, P_3, P_4 and encrypt them to C_1, C_2, C_3, C_4, where (P_1, P_2) and (P_3, P_4) both satisfy input difference α, *i.e.*, $P_1 \oplus P_2 = P_3 \oplus P_4 = \alpha$.
2. If ciphertexts satisfy the output difference $C_1 \oplus C_3 = C_2 \oplus C_4 = \delta$, these four plaintext-ciphertexts can construct a right quartet $\{(P_1, C_1), (P_2, C_2), (P_3, C_3), (P_4, C_4)\}$.

The generic key recovery algorithm. Another line of research on the rectangle attack is to mount key recovery attacks as efficiently as possible. The rectangle key recovery algorithm includes four steps: (1) data collection, (2) pair construction, (3) quartet generation and processing, and (4) exhaustive search. In the past few years, efforts have been made to find more effective key guessing strategies to improve the efficiency of key recovery attacks, like [14, 32]. In the generic algorithm of Song *et al.* [25], which we are inspired by, one can select part of partial key bits involved in extended rounds to guess. Using the generic algorithm, the adversary can balance the complexities of each attack step by guessing the involved key reasonably. The outline of the key recovery algorithm can be profiled in Fig. 2.

The notations involved in the upcoming work will be described for a better understanding. As shown in Fig. 2, α' is the differential obtained by the propagation of α through E_b^{-1}, and δ' is the differential obtained by the propagation of δ through E_f. Note that not all quartets which satisfy the difference α' and δ' are useful to suggest and extract the right key. However, quartets that do not satisfy such conditions are necessarily useless. r_b and r_f are the numbers of unknown bits of input differential and output differential. k_b and k_f denote the subkey bits for verifying the differential propagation in E_b and E_f, respectively, where $m_b = |k_b|$ and $m_f = |k_f|$ are the size of k_b and k_f. In our attack, we guess part

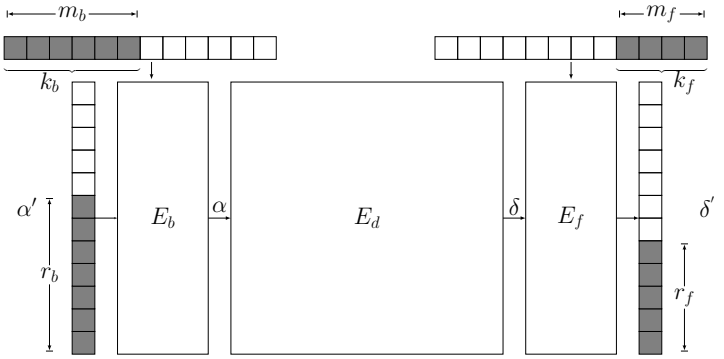

Fig. 2: Outline of rectangle key recovery attack [25]

of k_b and k_f, so we denote k'_b and k'_f as the bits in k_b and k_f which have been guessed. Similarly, $m'_b = |k'_b|$, $m'_f = |k'_f|$, and r'_b, r'_f are the number of inactive state bits which can be deduced by guessing key bits. Besides, in order to clearly describe the new attack, we define $r^*_b = r_b - r'_b$ and $m^*_b = m_b - m'_b$ (resp. r^*_f and m^*_f).

Related-key rectangle attack. Under the related-key scenario, the rectangle attack and the rectangle key recovery attack differ slightly from those under the single-key setting. Let ΔK and ∇K be the key differences for E_0 and E_1. In the phase of data collection, the adversary needs to access four related-key oracles with K_1, $K_2 = K_1 \oplus \Delta K$, $K_3 = K_1 \oplus \nabla K$ and $K_4 = K_1 \oplus \Delta K \oplus \nabla K$ to obtain four plaintext-ciphertexts of (P_1, C_1), (P_2, C_2), (P_3, C_3) and (P_4, C_4) respectively. The remaining steps should be performed under the related-key oracles as well.

The success probability. From the method of [23], the success probability of the rectangle key recovery attack is calculated according to the Eq. 1, where $S_N = \hat{p}^2\hat{q}^2/2^{-n}$ is the signal/noise ratio, with an h-bit or higher advantage. s is the expected number of right quartets.

$$P_s = \Phi\left(\frac{\sqrt{sS_N} - \Phi^{-1}\left(1 - 2^{-h}\right)}{\sqrt{S_N + 1}}\right) \tag{1}$$

2.3 The Rectangle Distinguisher of GIFT-64

Our attack is based on the 20-round related-key boomerang distinguisher of Ji *et al.* [16]:

$$\alpha = 00\ 00\ 00\ 00\ 00\ 00\ a0\ 00,$$
$$\delta = 04\ 00\ 00\ 00\ 01\ 20\ 10\ 00.$$

with:

$$\Delta K = 0004\ 0000\ 0000\ 0800\ 0000\ 0000\ 0000\ 0010,$$
$$\nabla K = 2000\ 0000\ 0000\ 0000\ 0800\ 0000\ 0200\ 0800.$$

The probability of the distinguisher above is $P_d = 2^{-58.557}$. Li *et al.* [18] have increased this probability to $2^{-57.43}$ by improving the BCT of the distinguisher (but otherwise using the same parameters as Ji *et al.*). Note that this paper does not discuss the calculation of the distinguisher probability but focuses on the key guessing strategy in the key recovery phase.

3 New Rectangle Key Recovery Attack on GIFT-64

In this section, we propose the new rectangle key recovery attack on GIFT-64. We begin by describing the specific parameters of the attack and the basic attack idea. In subsection 3.1, we will detail the dedicated MILP model and the key guessing strategies based on this model. In subsection 3.2, we will describe the exact process of the attack. The complexity of the two attacks will be calculated in subsection 3.3.

As illustrated in Table 4, we utilize the distinguisher of Ji *et al.* to attack 26-round GIFT-64, where ? denotes the bit with the unknown difference, 0 and 1 represent the bit with the fixed difference. E_b spans the first three rounds, and 44 unknown bits in E_b distribute over ΔY_1, thus $r_b = \boldsymbol{44}$. The number of subkey bits involved in the unknown difference in Z_1 and Z_2 is 24 and 6, respectively, so $m_b = \boldsymbol{30}$. E_f spans the last three rounds, and there are 64 unknown bits in E_f distributing over Z_{26}, thus $r_f = \boldsymbol{64}$. The involved 32 bits, 24 bits, and 8 bits subkeys are added to state Z_{26}, Z_{25}, and Z_{24} respectively, so $m_f = \boldsymbol{64}$. Hence, the number of subkey bits involved in k_b and k_f is $m_b + m_f = \boldsymbol{94}$ bits.

Table 4: The 26-round related-key rectangle attack on GIFT-64. For round r, ΔX_r and ΔY_r are the input and output differences of the S-boxes, and ΔZ_r is the output difference of the linear layer. $r \in \{1, 2, ..., 26\}$

input	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
ΔY_1	??0?	1??0	01??	?0??	1?0?	?1?0	0???	?0??	??0?	???0	0???	?0??	??0?	???0	0???	?0??
ΔZ_1	????	????	????	????	0000	0000	0000	0000	11??	????	????	????	????	11??	????	????
ΔX_2	????	????	????	????	0000	0000	0000	0000	11??	????	????	????	????	11??	????	????
ΔY_2	0?01	00?0	000?	?000	0000	0000	0000	0000	0100	00?0	000?	?000	?000	0100	00?0	000?
ΔZ_2	????	0000	?1??	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	?1??
ΔX_3	????	0000	?1??	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	?1??
ΔY_3	1000	0000	0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0010
ΔZ_3	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0010	1010	0000	0000	0000
$\Delta X_4(\alpha)$	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	1010	0000	0000	0000
:								...	...							
$\Delta X_{24}(\delta)$	0000	0100	0000	0000	0000	0000	0000	0000	0000	0001	0010	0000	0001	0000	0000	0000
ΔY_{24}	0000	???1	0000	0000	0000	0000	0000	0000	0000	????	????	0000	????	0000	0000	0000
ΔZ_{24}	00?0	0000	00??	0?00	0001	0000	?00?	00?0	?000	0000	??00	000?	0?00	0000	0??0	?000
ΔX_{25}	00?0	0000	00??	0?00	0001	0000	?00?	00?0	?010	0000	??00	000?	0?00	0000	0??0	?000
ΔY_{25}	????	0000	????	????	????	0000	????	????	????	0000	????	????	????	0000	????	????
ΔZ_{25}	??0?	??0?	??0?	??0?	???0	???0	???0	???0	0???	0???	0???	0???	?0??	?0??	?0??	?0??
ΔX_{26}	??0?	??0?	??0?	??0?	???0	???0	???0	???0	0???	0???	0???	0???	?0??	?0??	?0??	?0??
ΔY_{26}	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
ΔZ_{26}	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
output	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????

Thanks to the bit-wise linear key schedule of GIFT-64, the subkey bits are reused in certain rounds. For example, if we guess RK_1, we can obtain RK_{25} and vice versa. This relationship between subkey bits is also present in RK_2 and RK_{26}. As shown in Table 5, we mark the subkey bits involved in k_b and k_f as bold, the subkey bits shared by RK_1 and RK_{25} as green, and the subkey bits shared in RK_2 and RK_{26} as cyan. Note that only the bolded-and-colored subkey bits are reusable. The number of reusable subkey bits is ***26*** (20 bits between RK_1 and RK_{25}, say $k_1|k_0$ in Table 5. 6 bits between RK_2 and RK_{26}, say $k_3|k_2$ in Table 5). Therefore, the number of subkey bits involved in the attack is $94 - 26 =$ ***68***.

Table 5: The relation of the involved subkey bits in the key recovery phase

ΔZ_1	????	????	????	????	0000	0000	0000	0000	11??	????	????	????	????	11??	????	????
$k_1\|k_0$	**15**|**15**	**14**|**14**	**13**|**13**	**12**|**12**	11|11	10|10	9|9	8|8	**7**|**7**	**6**|**6**	**5**|**5**	**4**|**4**	**3**|**3**	**2**|**2**	**1**|**1**	**0**|**0**
ΔZ_2	????	0000	?1??	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	?1??
$k_3\|k_2$	**15**|**15**	14|14	**13**|**13**	12|12	11|11	10|10	9|9	8|8	7|7	6|6	5|5	4|4	3|3	2|2	1|1	**0**|**0**
...	... ...															
ΔZ_{25}	??0?	??0?	??0?	??0?	???0	???0	???0	???0	0???	0???	0???	0???	?0??	?0??	?0??	?0??
$k_1\|k_0$	11|**7**	10|**6**	9|**5**	8|**4**	**7**|3	**6**|2	**5**|1	**4**|0	**3**|**15**	**2**|**14**	**1**|**13**	**0**|**12**	**15**|**11**	**14**|**10**	**13**|**9**	**12**|**8**
ΔZ_{26}	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
$k_3\|k_2$	**11**|**7**	**10**|**6**	**9**|**5**	**8**|**4**	**7**|**3**	**6**|**2**	**5**|**1**	**4**|**0**	**3**|**15**	**2**|**14**	**1**|**13**	**0**|**12**	**15**|**11**	**14**|**10**	**13**|**9**	**12**|**8**

Based on the observation of the previous rectangle key recovery attack instance and the analysis of the GIFT-64 key schedule, we summarize the attack strategies into the following two observations:

- The phases of pair construction and quartet generation and processing dominate the time cost of our attacks. In general, there is a certain extent inverse relationship between the time consumption of pair construction and quartet generation and processing. Both of these parts are related to the number of guessed subkey bits and the number of filtering bits produced. Balancing the time complexity of these two parts with appropriate guessing subkey bits will be the core idea of our improvement.
- Our key guessing strategies are based on the idea of finding a method that requires less guessing time complexity but can obtain more conditional bits, *i.e.*, $m'_b + m'_f \leq 2(r'_b + r'_f)$. The reused subkey bits can provide additional filtering bits for some guesses. However, excessive guessing of these subkey bits will make the complexity of pair construction and quartet generation and processing unbalanced. Under the consideration of trading off the time complexity, guessing as many reused subkey bits as possible will maximize the advantage of our attack.

3.1 The Dedicated Model and New Key Guessing Strategy

As a block cipher with the bit-wise key schedule and bit-wise linear layer, the structure of GIFT-64 allows us to directly reap the benefits of guessing each

bit. These advantages provide greater flexibility for our attack. Our attack is under the related-key setting, so the filtering process corresponds to the guessed subkeys and their complement. Note that we propose Attack I and Attack II in this subsection.

Since mouha *et al.*'s seminal paper [20], Mixed Integer Linear Programming (MILP) has been widely used in automated cryptanalysis and has yielded promising results in numerous cryptographic key recovery attacks [15, 22, 28, 30]. In this paper, we present a dedicated model for automated key recovery analysis for GIFT-64 based on our new subkey guessing strategy. We start our modeling with a selection of plaintexts and ciphertexts that satisfy the difference of Z_1 and Z_{26}, respectively. We use binary variables to indicate whether each bit is known and mark all the output bits of an S-box as known if and only if both its input bits and the 2 bits subkey involved are known. Guessing the subkey bits allows the propagation of the knownness to obtain the filtering bits. Note that the differential propagation in E_b is the exact opposite of the differential propagation in E_f. We then count the filtering bits of the S-box for which both the input and output differences are known and calculate the time complexity of each attack step. Naturally, we set our objective function to balance the time complexity of each attack step optimally. The parameters we used are listed in Table 6, and the dedicated model is described in Model 1.

Attack I: The guessing of the keys involved in the forward and backward extend rounds, shown in Fig. 3, and described as follows. Note that we have marked in red (resp. blue) the keys guessed in the E_b (resp. E_f) and the filters that can be used. Next, the guessed key bits and filters involved in E_b and E_f are explained.

Involved in E_b: Choose the plaintexts that satisfy ΔZ_1. If $k_0[0]$ and $k_1[0]$ are guessed, we can obtain the value of $X_2[3:0]$ and filter out the pairs of plaintexts that do not satisfy the difference 000? by using the filter $GS(X_2[3:0]) \oplus GS(X_2[3:0] \oplus \Delta X_2[3:0]) = \Delta Y_2[3:0]$. As shown in Fig. 3, there are none bit in $\Delta X_2[3:0]$ with fixed difference and 3 bits in $\Delta Y_2[3:0]$ with fixed difference. When $k_1[0]$ and $k_0[0]$ are guessed, there exist 3 filtering bits (???? $\rightarrow$ 000?) via the corresponding S-box. We guess 24 bits $k_0[15, 14, 13, 12, 7, 6, 5, 4, 3, 2, 1, 0]$ and $k_1[15, 14, 13, 12, 7, 6, 5, 4, 3, 2, 1, 0]$ to obtain 34 filtering bits in round 2. We guess 6 bits $k_2[15, 13, 0]$ and $k_3[15, 13, 0]$ to obtain ten filtering bits in round 3. Therefore, we guess ***30 subkey bits*** and obtain ***44 filtering bits*** in E_b, *i.e.*, $m'_b =$***30*** can verify $r'_b =$***44***, and $r^*_b = r_b - r'_b =$***0***.

Involved in E_f: The filtering bits are obtained in the same way as described in E_b. We guess 16 bits $k_2[8, 7, 6, 5, 4, 3, 2, 1]$ and $k_3[14, 12, 11, 10, 9, 8, 7, 5]$, combined with the guessed subkey bits $k_2[15, 13, 0]$ and $k_3[15, 13, 0]$ which can be reused, this guess provides 11 filtering bits in round 26. We guess 2 bits $k_0[10, 9]$, combined with $k_0[14, 13, 6, 5]$ and $k_1[14, 13, 6, 5, 2, 1]$ that can be reused. This guess provides 17 filtering bits in round 25. Therefore, we guess ***18 subkey bits*** and obtain ***28 filtering bits*** in E_f, *i.e.*, $m'_f =$ ***18*** can verify $r'_f =$***28***, and $r^*_f = r_f - r'_f =$***36***.

Table 6: Parameters of dedicated model

Parameter	Implication
$R_0 \in \{1, 2, 3\}$	Set of extension rounds in E_b
$R_1 \in \{24, 25, 26\}$	Set of extension rounds in E_f
$r \in \{R_0 \| R_1\}$	Round number in extension rounds
$m \in \{0, ..., 16\}$	S-box position in each round
$n \in \{0, ..., 4\}$	Bit position in the input (resp. output) of S-box
$s \in \{0, ..., 4\}$	Number of attack step
Binary $KX_{r,m,n}, KY_{r,m,n}$	Bit difference is known or not
Binary $FX_{r,m,n}, FY_{r,m,n}$	Bit difference is fixed or not
Binary $GK_{r,m}$	Subkey bits is guessed or not
Integer $A_{r,m}$	Number of filtering bits of each filter
General T_s	Time complexity of each attack step
General T	Upper bound of T_s

Algorithm 1 Optimal key guessing strategy searching

Require: R_0 rounds in E_b and R_1 rounds in E_f, the system of inequalities for linear layer and its inverse, the number of S-box in each round m, the size of each S-box n, multi binary variables $KX_{r,m,n}, KX_{r,m,n}, FX_{r,m,n}, FY_{r,m,n}, GK_{r,m}$, multi integer variables $A_{r,m}$, multi general variables T_s and T

Ensure: system of inequalities

/* Initialization */

Constraints: all $KX_0 = KY_{26} = 1$

Constraints: $FX_{r,m,n}$ and $FY_{r,m,n}$ follow the differential in E_b and E_f

/* Counting the number of filtering bits */

for r in $\{R_0\}$ **do**
  for $i = 0$ to m **do**
    Constraints: all $KY_{r,i,*} = 1$ if and only if all $KX_{r,i,*} = GK_{r,i} = GK_{r,i'} = 1$, otherwise all $KY_{r,i,*} = 0$
    Constraints: $A_{r,i} = KY_{r,i,*} \times (sum(FY_{r,i}) - sum(FX_{r,i}))$
  end for
  Constraints: Linear permutation from $KY_{r,*,*}$ to $KX_{r+1,*,*}$
end for
for r in $\{R_1\}$ **do**
  for $i = 0$ to m **do**
    Constraints: all $KX_{r,i,*} = 1$ if and only if all $KY_{r,i,*} = GK_{r,i} = GK_{r,i'} = 1$, otherwise all $KX_{r,i,*} = 0$
    Constraints: $A_{r,i} = KX_{r,i,*} \times (sum(FX_{r,i}) - sum(FY_{r,i}))$
  end for
  Constraints: Inversed linear permutation from $KX_{r,*,*}$ to $KY_{r-1,*,*}$
end for

/* Computing time complexity */

for s in the number of attack steps **do**
  Constraints: T_s follow Eq. 6
  Constraints: $T >= T_s$
end for

/* Objective function */

MINIMIZE T

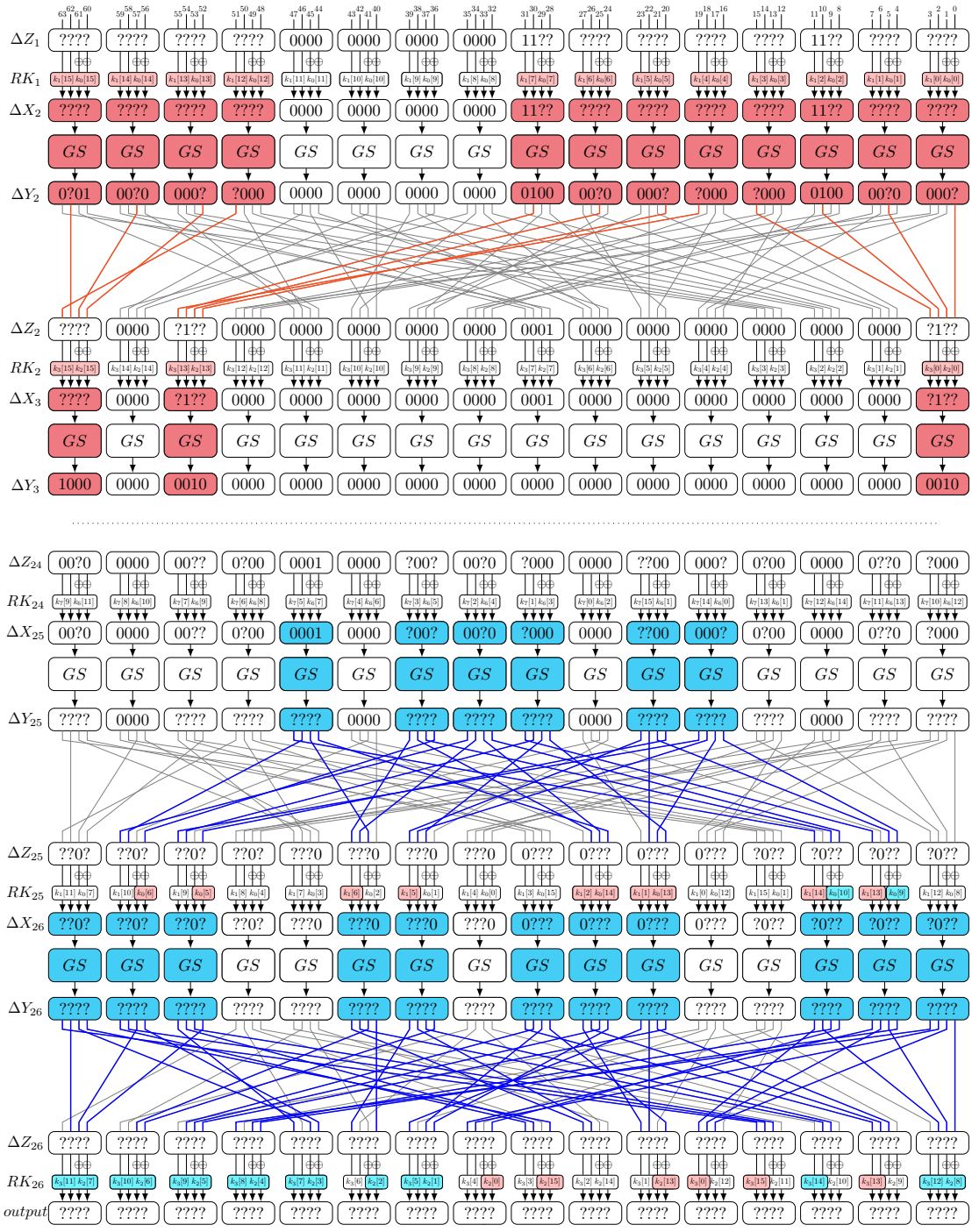

Fig. 3: Guessed key bits and the corresponding propagation relations for Attack I

Attack II: We also performed the attack using another similar key-guessing strategy. This attack has a slightly lower time complexity when computing ϵ using a different method than Attack I. The parameters of the attack are only briefly described here, and the attack processes are the same as in Attack I. We calculate the complexity of this attack in subsection 3.3 and include a detailed description in Fig. 4 of Appendix A. The key guessing strategy for this attack is as follows:

Involved in E_b: In round 1, we guess 22 bits $k_0[15, 14, 13, 12, 7, 6, 5, 4, 3, 1, 0]$ and $k_1[15, 14, 13, 12, 7, 6, 5, 4, 3, 1, 0]$ to obtain 32 filtering bits. In round 2, we guess 4 bits $k_2[13, 15]$ and $k_3[13, 15]$ to obtain 7 filtering bits. Therefore, we have guessed 26 bits of subkeys and obtained 39 filtering bits, *i.e.*, $m'_b = \boldsymbol{26}$, $r'_b = \boldsymbol{39}$, $r^*_b = r_b - r'_b = \boldsymbol{5}$.

Involved in E_f: In round 26, we guess 20 bits $k_2[14, 12, 7, 6, 5, 4, 3, 2, 1, 0]$ and $k_3[14, 12, 11, 10, 9, 8, 7, 6, 5, 4]$, combine with the reused $k_2[15, 13]$ and $k_3[15, 13]$, this guess provides 12 filtering bits. In round 25, we guess 2 bits $k_0[11, 9]$, combine with the reused $k_0[15, 13, 7, 6, 5]$ and $k_1[15, 14, 13, 7, 6, 5, 3, 1]$, this guess provides 22 filtering bits. Therefore, we have guessed 22 bits of subkeys and obtained 34 filtering bits, *i.e.*, $m'_f = \boldsymbol{22}$, $r'_f = \boldsymbol{34}$, $r^*_b = r_b - r'_b = \boldsymbol{30}$.

3.2 New Rectangle Attack on GIFT-64

In this subsection, we conducted the rectangle key recovery attack against GIFT-64 using different key guessing strategies. For both attacks, we choose $y = \sqrt{s} \cdot 2^{\frac{n}{2} - r_b} / \sqrt{P_d}$ to be the number of structures that should be pre-constructed, where s is the number of quartets expected to be correct.

The key guessing strategy is detailed in subsection 3.1. The complexities will be calculated in subsection 3.3. The attack process is as follows:

1. Collect and store y structures of 2^{r_b} plaintexts each. Encrypt these structures under four related keys K_1, K_2, K_3, and K_4. We obtain four datasets containing the plaintext-ciphertexts under four related keys, denote as L_1, L_2, L_3, and L_4. Let $D = y \cdot 2^{r_b}$ for convenience, so the time, data, and memory complexity of this step is $T_0 = M_0 = D_{total} = 4 \cdot D = y \cdot 2^{r_b+2}$.
2. Guess $(m'_b + m'_f)$-bit key, and for each guess:
   (a) Initialize a list of key counters for the unguessed subkey bits of k_b and k_f, the memory complexity of key counters is $M_c = 2^{m^*_b + m^*_f}$.
   (b) For each plaintext-ciphertext collected in step 1, partially encrypt P_i and partially decrypt C_i under the key bits we guessed, *i.e.*, $P^*_i = Enc_{k'_b}(P_i)$, $C^*_i = Dec_{k'_f}(C_i)$, where $i \in \{1, 2, 3, 4\}$. The time complexity of this step is $4 \cdot D = y \cdot 2^{r_b+2}$ partial encryptions.
   (c) Choose to construct pairs in the input of E_b. The reason is that constructing pairs in the input of E_b can get more filters for the phase of constructing pairs and balance the time complexity
      i. Insert L_1 into a hash table indexed by the r'_b inactive bits of P^*_1, so there are $2^{r'_b}$ items, each of which comprises $2^{r^*_b}$ pairs (P^*_1, C^*_1). For

L_2, find matches in the hash table according to the r'_f inactive bits of P_2^*. Each match corresponds to a pair $\{(P_1^*, C_1^*), (P_2^*, C_2^*)\}$. There are $y \cdot 2^{r'_b}$ items in total, and there will be $2 \cdot \binom{2^{r_b^*}}{2}$ different combinations for each item. Perform the same operation to L_3 and L_4, so we can get two sets $S_1 = \{(P_1^*, C_1^*), (P_2^*, C_2^*)\}$ and $S_2 = \{(P_3^*, C_3^*), (P_4^*, C_4^*)\}$. The size of each is:

$$y \cdot 2^{r'_b} \cdot 2 \cdot \binom{2^{r_b^*}}{2} = D \cdot 2^{r_b^*}. \tag{2}$$

The memory complexity of this step is $M_1 = 2 \cdot D \cdot 2^{r_b^*}$ for storing sets S_1 and S_2. We need $D \cdot 2^{r_b^*}$ look-up tables to construct each set. The time complexity of this step is $2 \cdot D \cdot 2^{r_b^*}$ memory accesses.

ii. Insert S_1 into a hash table indexed by the $2r'_f$ inactive bits of both C_1^* and C_2^*. Since each set contains $D \cdot 2^{r_b^*}$ pairs, there are $2^{2r'_f}$ items, each of which comprises $D \cdot 2^{r_b^* - 2r'_f}$ pairs $\{(P_1^*, C_1^*), (P_2^*, C_2^*)\}$. For S_2, find matches in the hash table according to the $2r'_f$ inactive bits of both C_3^* and C_4^*. Each matching provides a quartet $\{(P_1^*, C_1^*), (P_2^*, C_2^*), (P_3^*, C_3^*), (P_4^*, C_4^*)\}$ for us. For each item, similar to step 2(c)i, there are $2 \cdot \binom{\frac{D \cdot 2^{r_b^*}}{2^{2r'_f}}}{2}$ different combinations. So the number of quartets we generate is

$$2^{2r'_f} \cdot 2 \cdot \binom{\frac{D \cdot 2^{r_b^*}}{2^{2r'_f}}}{2} = D^2 \cdot 2^{2r_b^* + 2r_f^* - 2r_f}. \tag{3}$$

The time complexity of this step is $D^2 \cdot 2^{2r_b^* + 2r_f^* - 2r_f}$ memory accesses.

(d) Utilize the quartets we obtain above to determine the key candidates involved in E_b and E_f and increase the corresponding key counters. Denote the time of the process of each quartet as ϵ. The time complexity of this step is $D^2 \cdot 2^{2r_b^* + 2r_f^* - 2n} \cdot \epsilon$. We will discuss the estimation of ϵ in detail in subsection 3.3.

(e) Select the top $2^{m_b^* + m_f^* - h}$ hits in the counters to be the key candidates, which delivers a h-bit or higher advantage, where $0 < h \leq m_b^* + m_f^*$.

(f) Guess the remaining $k - m_b - m_f$ bits key according to the key schedule and exhaustively search over them to recover the correct key, where k is the key size of GIFT-64. The time complexity of this step is $2^{k - m'_b - m'_f - h}$ encryptions.

3.3 Complexity Analysis

We will introduce two different methods to process quartets and extract key information, *i.e.*, calculation of ϵ, which will bring different time complexity

to our attacks. We apply ϵ computed using the method of the pre-constructed hash table into the MILP model and compute the time complexity of Attack I accordingly. We also bring ϵ computed using the method of the guess and filter to the MILP model and compute the time complexity of Attack II accordingly. We mount this approach to our two attacks and take Attack I as an example to introduce it in detail. The complexity analysis of Attack II is calculated in detail in the appendix A.

Guess and filter: Suppose we obtain N_q quartets after step 2d in the first attack. Guessing $k_2[14, 12]$ and $k_3[6, 4]$, we can use filters $GS(Y_{26}[15 : 12]) \oplus GS(Y_{26}[15 : 12] \oplus \Delta Y_{26}[15 : 12] = ?0??$ and $GS(Y_{26}[47 : 44]) \oplus GS(Y_{26}[47 : 44] \oplus \Delta Y_{26}[47 : 44] = ?0??$. Furthermore, combined with the subkey bits $k_0[15, 7, 3, 1]$ and $k_1[15, 7, 3]$, which have been guessed before, we can use filters $GS(Y_{25}[63 : 60]) \oplus GS(Y_{25}[63 : 60] \oplus \Delta Y_{25}[63 : 48] = 00?0$ and $GS(Y_{25}[55 : 48]) \oplus GS(Y_{25}[55 : 48] \oplus \Delta Y_{25}[63 : 48] = 00??, 0?00$. There are 10 filtering bits for pairs, so 20 filtering bits for quartets. Therefore, there are $2^4 \cdot N_q \cdot 2^{-20} = N_q \cdot 2^{-16}$ quartets remain, and the time cost is $2^4 \cdot N_q$ S-box accesses. Note that after this step, the number of quartets remaining is much lower than before. The time consumption of the remaining steps will be far less than $N_q \cdot 2^4$ S-box accesses. Therefore, $\epsilon \approx 2^4/26 \approx 2^{-0.7}$ encryption.

Table 7: Precomputation tables for the 26-round attack on GIFT-64

No.	Starting bits	Subkey bits	Bits deduced	Filter condition	T & M	Filter effect
1	$\underline{Y_{26}[47 : 44]}$ $\underline{Y_{26}[15 : 12]}$ $\underline{X_{26}[63 : 60]}$ $\underline{X_{26}[31 : 28]}$	$\underline{k_2[14, 12]}$ $\underline{k_3[6, 4]}$	$Y'_{26}[47 : 44]$ $Y'_{26}[15 : 12]$ $X_{26}[47 : 44]$ $X'_{26}[47 : 44]$ $X_{26}[15 : 12]$ $X'_{26}[15 : 12]$ $X_{25}[63 : 60]$ $X'_{25}[63 : 60]$ $X_{25}[55 : 48]$ $X'_{25}[55 : 48]$	$X_{26}[14] \oplus X'_{26}[14] = 0$ $X_{26}[44] \oplus X'_{26}[44] = 0$ $X_{25}[63 : 60] \oplus X'_{25}[63 : 60]$ $= 00?0$ $X_{25}[55 : 48] \oplus X'_{25}[55 : 48]$ $= 00??, 0?00$	2^{48}	2^{-16}
	$Y^i_{26}[47 : 44], Y^i_{26}[15 : 12], X_{26}[63 : 60], X_{26}[31 : 28], i = 1, 2, 3, 4$: $k_2[14, 12], k_3[6, 4]$					
2	$\underline{Y_{26}[51 : 48]}$ $\underline{Y_{26}[35 : 32]}$ $\underline{Y_{26}[19 : 16]}$ $\underline{X_{26}[3 : 0]}$	$\underline{k_0[8]}$ $\underline{k_1[8]}$ $\underline{k_2[14, 10, 9]}$ $\underline{k_3[6, 2, 1]}$	$Y'_{26}[51 : 48]$ $Y'_{26}[35 : 32]$ $Y'_{26}[19 : 16]$ $X_{26}[51 : 48]$ $X'_{26}[51 : 48]$ $X_{26}[35 : 32]$ $X'_{26}[35 : 32]$ $X_{26}[19 : 16]$ $X'_{26}[19 : 16]$ $X_{25}[15 : 12]$ $X'_{25}[15 : 12]$ $X_{25}[7 : 0]$ $X'_{25}[7 : 0]$	$X_{26}[49] \oplus X'_{26}[49] = 0$ $X_{26}[32] \oplus X'_{26}[32] = 0$ $X_{26}[19] \oplus X'_{26}[19] = 0$ $X_{25}[15 : 12] \oplus X'_{25}[15 : 12]$ $= 0?00$ $X_{25}[7 : 0] \oplus X'_{25}[7 : 0]$ $= 0??0, ?000$	2^{50}	2^{-14}
	$Y^i_{26}[51 : 48], Y_{26}[35 : 32], Y_{26}[19 : 16], X_{26}[3 : 1], i = 1, 2, 3, 4$: $k_0[8], k_1[8], k_2[14, 10, 9], k_3[6, 2, 1]$					

Pre-construct hash tables: As shown in Table 7, the time complexity of step 2d can be reduced by accessing the hash table instead of traversing subkeys. We can obtain the bits deduced according to the inputs of the filters, *i.e.*, starting bits and subkey bits. Utilize these bits to match pairs in the quartets. Each match can provide the corresponding subkey information. The time and memory cost of each subtable is determined by the underlined bits, and the filtering effect indicates the number of candidate subkeys obtained for each subtable access. Note that these hash tables in our paper are all built for quartets but can also be built for pairs in memory-limited cases.

As shown in Fig. 3, these uncoloured cells of ΔRK_i denote the subkey bits not used in previous steps, where $i = 24, 25, 26$. We should access subtable No.1 in Table 7 first. Combining state $Y_{26}[47:44], Y_{26}[15:12], X_{26}[63:60], X_{26}[31:28]$ with subkey bits $k_2[14,12]$, $k_3[6,4]$, we can deduce the corresponding $Y'_{26}[47:44]$, $Y'_{26}[15:12]$, $X_{26}[47:44]$, $X'_{26}[47:44]$, $X_{26}[15:12]$, $X'_{26}[15:12]$, $X_{25}[63:60]$, $X'_{25}[63:60]$, $X_{25}[55:48]$, $X'_{25}[55:48]$. We can obtain filtering bits from $X_{26}[14] \oplus X'_{26}[14] = 0$, $X_{26}[44] \oplus X'_{26}[44] = 0$, $X_{25}[63:60] \oplus X'_{25}[63:60] = 00?0$, $X_{25}[55:48] \oplus X'_{25}[55:48] = 00??, 0?00$. Specifically, for a pair of plaintext, there are 10 filtering bits from $\Delta Y_{26}[47:44] = ???? \rightarrow \Delta X_{26}[47:44] = ???0, \Delta Y_{26}[15:12] = ???? \rightarrow \Delta X_{26}[15:12] = ?0??, \Delta Y_{25}[63:60] = ???? \rightarrow \Delta X_{25}[63:60] = 00?0, \Delta Y_{25}[31:28] = ????, ???? \rightarrow \Delta X_{25}[31:28] = 00??, 0?00$. Therefore, we can get 20 filtering bits for a quartet using these filters. According to these filters, we can extract information of $k_2[14,12]$, $k_3[6,4]$ and discard quartets that suggest nothing. We need to store subkey bits $k_2[14,12]$, $k_3[6,4]$ into a hash table indexed by 64-bit $Y^i_{26}[47:44], Y^i_{26}[15:12], X^i_{26}[63:60], X^i_{26}[31:28]$ in total, where $i = 1, 2, 3, 4$. The filter effect is 2^{-16}, which means 2^{16} quartets will be filtered out, so the time and memory cost is $2^{64-16} = 2^{48}$.

For now, the number of quartets is much lower than before. To process the rest of the quartets and increase key counters, we need to access the remaining subtable No.2. The time cost of the following steps will be far less than that of accessing subtable No.1. Finally, we get $\epsilon \approx 1$ memory access.

The complexity of Attack I: Considering the total complexity and success rate, we choose $s = 2$ and $h = 20$, so we need construct $y = \sqrt{s} \cdot 2^{\frac{n}{2}-r_b}/\sqrt{P_d} = 2^{17.78}$ structures in step 1 and $D = y \cdot 2^{r_b} = 2^{61.78}$.

- *Data complexity*: The plaintexts in step 1 are the total data we need to collect in this attack.

$$D_{total} = 4 \cdot D = 2^{63.78} \tag{4}$$

- *Memory complexity*: We need store data collected in step 1, key counters described in step 2a, and datasets S_1 and S_2 generated in step 2(c)i.

$$
\begin{aligned}
M_{total} &= M_0 + M_1 + M_c \\
&= 4 \cdot D + 2 \cdot D \cdot 2^{r_b^*} + 2^{m_b^* + m_f^*} \\
&= 2^{63.78} + 2^{62.78} + 2^{36} \\
&\approx 2^{64.36}
\end{aligned} \tag{5}
$$

- *Time complexity*: The time complexity consists of five parts: (1) data collection in step 1 denote as T_0, (2) partial encryption and partial decryption in step 2b denote as T_1, (3) pairs generation in step 2(c)i denote as T_2, (4) quartets generation and processing in step 2d denote as T_3 and (5) the exhaustive search of the $k - m_b - m_f$-bit key in step 2f denote as T_4. Recall that our key guessing strategy provides the following parameters: $m_b' = 30, r_b' = r_b = 44, r_b^* = 0$ and $m_f' = 18, r_f' = 28, r_f^* = r_f - r_f' = 36$. For the last four parts, we need to consider all guessing of $(m_b' + m_f') = 48$ subkey bits.

$$
\begin{aligned}
T_0 &= 4 \cdot D = 2^{63.78} \\
T_1 &= 2^{m_b' + m_f'} \cdot 4 \cdot D = 2^{111.78} \\
T_2 &= 2^{m_b' + m_f'} \cdot 2 \cdot D \cdot 2^{r_b^*} = 2^{110.78} \\
T_3 &= 2^{m_b' + m_f'} \cdot D^2 \cdot 2^{2r_b^* + 2r_f^* - 2n} \cdot \epsilon = 2^{115.56} \cdot \epsilon \\
T_4 &= 2^{m_b' + m_f' + k - m_b' - m_f' - h} = 2^{k-h} = 2^{108}
\end{aligned} \tag{6}
$$

We obtain $\epsilon \approx 2^{-0.7}$ encryption with the method of the guess and filter. The time complexity of Attack I is $T_1 + T_3 + T_4 = \frac{6}{26} \cdot 2^{111.78} + 2^{115.56-0.7} + 2^{108} \approx \mathbf{2^{114.86}}$ encryption and $T_2 = \mathbf{2^{110.78}}$ memory access.

We obtain $\epsilon \approx 1$ memory access with the method of the pre-constructed hash table. The time complexity of Attack I is $T_1 + T_4 = \frac{6}{26} \cdot 2^{111.78} + 2^{108} \approx 2^{111.78-2.11} + 2^{108} \approx \mathbf{2^{110.06}}$ 26-round encryptions and $T_2 + T_3 = 2^{110.78} + 2^{115.56} \approx \mathbf{2^{115.80}}$ memory accesses.

According to Eq. 1, the success rate of Attack I is around 75%.

The complexity of Attack II: We choose $s = 2$ and $h = 20$, so $D = 2^{61.78}$. The data complexity is $D_{total} = 2^{63.78}$, memory complexity is $M'_{total} = 2^{67.8}$. We obtain that $\epsilon \approx 2^{-2.7}$ encryptions with the method of the guess and filter, so time complexity is $\mathbf{2^{111.51}}$ 26-round encryptions and $\mathbf{2^{115.78}}$ memory accesses. We obtain that $\epsilon \approx 1$ memory access with the method of the pre-construct hash table. Thence, the time complexity is $\mathbf{2^{110.06}}$ 26-round encryptions and $\mathbf{2^{116.06}}$ memory accesses. The complexity calculation of Attack II is detailed in Appendix A.

According to Eq. 1, the success rate of Attack II is around 75%.

In summary, we obtained that the best current rectangle key recovery attack on GIFT-64 is Attack I, which pre-constructs hash tables to process quartets. The attack complexity is: data complexity is $\boldsymbol{D_{total} = 2^{63.78}}$, memory complexity is $\boldsymbol{M_{total} = 2^{64.36}}$, time complexity is $\boldsymbol{T_{total} = 2^{110.06}}$ 26-round encryption

and $\mathbf{2^{115.80}}$ memory accesses. Further, calculating using the probability which improved by Li *et al.*, the overall complexity of the attack is: $D^*_{total} = 2^{63.22}$, $M^*_{total} = 2^{63.8}$, $T^*_{total} = 2^{109.5}$ 26-round encryption and $2^{115.24}$ memory accesses.

Reevaluate the Security of GIFT-128 We also re-evaluate the ability of GIFT-128 to resist rectangle key recovery attacks on the basis of [16]. In [16], Ji *et al.* guess all the subkey bits involved in E_b, non-guess any subkey bits in E_f, and they obtain the time complexities are T.

4 Discussion and Conclusion

We propose new key guessing strategies to improve the attack of 26-round GIFT-64. The MILP model of the best key guessing strategy for GIFT-64 rectangle key recovery attack is constructed, and thus the optimal key guessing strategy for our attack scenario is obtained by searching and comparing. Our attacks against GIFT-64 are the best in terms of time complexity so far. For GIFT-64, its bit-wise linear permutation gives us great flexibility, and the reused subkey bits derived by the bit-wise key schedule provide additional filter bits for each guessing. Our attack starts from the bit-wise operations of GIFT-64 and considers the more accurate selection of the subkey bits involved in the attack so as to guess the key more effectively. This idea also provides a new possibility for better attacks on block ciphers structured by bit-wise operations.

We have observed that the designer of GIFT had not claimed the related-key security. For the community of symmetric cryptography, however, we believe that cryptographic security analysis under related-key settings is indispensable.

5 Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This paper is supported by the National Key Research and Development Program (No. 2018YFA0704704, No.2022YFB2701900, No.2022YFB2703003) and the National Natural Science Foundation of China (Grants 62022036, 62132008, 62202460, 62172410).

A Complexity analysis of Attack II

Recall that the key guessing strategy for Attack II has the following parameters: $m'_b = 26$, $r'_b = 39$, $r^*_b = 5$, $m'_f = 22$, $r'_f = 34$, $r^*_f = 30$, which are also visually identified in Fig. 4. As in Attack I, we choose $s = 2$ and $h = 20$, so $D = 2^{61.78}$.

Guess and filter In the second attack, supposing the number of quartets to be processed is N'_q. We guess $k_0[2]$ and $k_1[2]$, which provide 4 filtering bits for quartets ($\Delta X_2[11:8] = 11?? \rightarrow \Delta Y_2[11:8] = 0100$), the filter is $GS(X_2[11:8]) \oplus GS(X_2[11:8] \oplus \Delta X_2[11:8]) = 0100$. Therefore, there are $2^2 \cdot N'_q \cdot 2^{-4} = N'_q \cdot 2^{-2}$

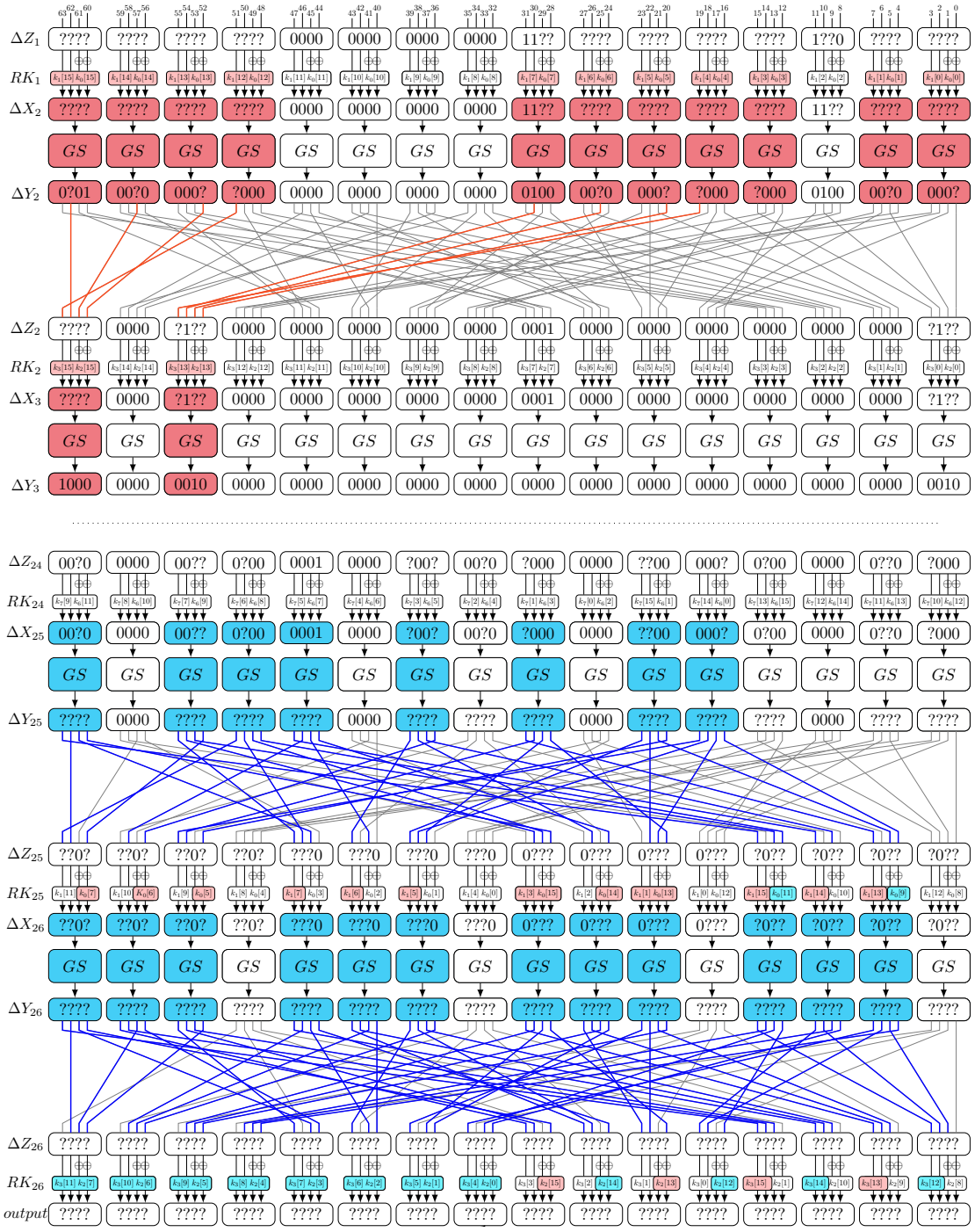

Fig. 4: Guessed key bits and the corresponding propagation relations for Attack II

quartets remain, and the time cost is $2^2 \cdot N'_q$ S-box accesses. Note that after this step, the number of quartets remaining is much lower than before. The time consumption of the remaining steps will be far less than $N'_q \cdot 2^2$ S-box accesses. Therefore, $\epsilon \approx 2^2/26 \approx 2^{-2.7}$ encryption.

Pre-construct hash tables: We also pre-construct hash tables in the second attack to calculate ϵ. The detail is in Table 8. With similar processes as Attack I, we obtain $\epsilon \approx 1$ memory access.

Table 8: Precomputation tables for the 26-round attack on GIFT-64

No.	Starting bits	Subkey bits	Bits deduced	Filter condition	TM	Filter effect
1	$\underline{X_2[11:8]}$ $\underline{X_3[3:0]}$ $\underline{Y_{25}[35:32]}$	$\underline{k_0[0]}$ $\underline{k_1[0]}$ $\underline{k_2[0]}$ $\underline{k_3[0]}$	$X'_2[11:8]$ $X'_3[3:0]$ $Y_2[11:8]$ $Y'_2[11:8]$ $Y_3[3:0]$ $Y'_3[3:0]$ $X_{25}[35:32]$ $X'_{25}[35:32]$	$Y_2[35:32] \oplus Y'_2[35:32] = 0100$ $Y_3[3:0] \oplus Y'_3[3:0] = 0010$ $X_{25}[11:8] \oplus Y'_2[11:8] = 00?0$	2^{36}	2^{-12}
	$X_2[11:8], X_3[3:0], Y_{25}[35:32], i = 1,2,3,4$: $k_0[0], k_1[0], k_2[0], k_3[0]$					
2	$\underline{Y_{26}[51:48]}$ $\underline{Y_{26}[35:32]}$ $\underline{Y_{26}[19:16]}$ $\underline{Y_{26}[3:0]}$	$\underline{k_2[3,2,1,0]}$ $\underline{k_3[10,9,8,1]}$	$Y'_{26}[51:48]$ $Y'_{26}[35:32]$ $Y'_{26}[19:16]$ $X'_{26}[3:0]$ $X_{26}[51:48]$ $X'_{26}[51:48]$ $X_{26}[35:32]$ $X'_{26}[35:32]$ $X_{26}[19:16]$ $X'_{26}[19:16]$ $X_{26}[3:0]$ $X'_{26}[3:0]$ $X_{25}[15:12]$ $X'_{25}[15:12]$ $X_{25}[7:0]$ $X'_{25}[7:0]$	$X_{26}[49] \oplus X'_{26}[49] = 0$ $X_{26}[32] \oplus X'_{26}[32] = 0$ $X_{26}[19] \oplus X'_{26}[19] = 0$ $X_{26}[2] \oplus X'_{26}[2] = 0$ $X_{25}[15:12] \oplus X'_{25}[15:12]$ $= 0?00$ $X_{25}[7:0] \oplus X'_{25}[7:0]$ $= 0??0, ?000$	2^{48}	2^{-16}
	$Y^i_{26}[51:48], Y_{26}[35:32], Y_{26}[19:16], X_{26}[3:1], i = 1,2,3,4$: $k_2[10,9,8,1], k_3[3,2,1,0]$					

- *Data complexity:* The required data are the same as described in Eq. (4). $D_{total} = 4 \cdot D = 2^{63.78}$
- *Memory complexity:* Let M'_0, M'_1 and M'_c denote the memory complexity of storing pairs, quartets, and key counters respectively.

$$\begin{aligned} M'_{total} &= M'_0 + M'_1 + M'_c \\ &= 4 \cdot D + 2 \cdot D \cdot 2^{r^*_b} + 2^{m^*_b + m^*_f} \\ &= 2^{63.78} + 2^{67.78} + 2^{35} \\ &\approx 2^{67.8} \end{aligned} \tag{7}$$

- *Time complexity:* The time complexity consists of five parts: (1) data collection denoted as T'_0, (2) partial encryption and partial decryption denoted as T'_1, (3) two sets S'_1, S'_2 generation denoted as T'_2, (4) quartets generation and processing denoted as T'_3, and (5) exhaustively searching the remaining key bits denoted as T'_4.

$$\begin{aligned} T'_0 &= 4 \cdot D = 2^{63.78} \\ T'_1 &= 2^{m'_b+m'_f} \cdot 4 \cdot D = 2^{111.78} \\ T'_2 &= 2^{m'_b+m'_f} \cdot 2 \cdot D = 2^{115.78} \\ T'_3 &= 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_f-2n} \cdot \epsilon = 2^{113.56} \cdot \epsilon \\ T'_4 &= 2^{k-h} = 2^{108} \end{aligned} \tag{8}$$

We obtain that $\epsilon \approx 2^{-2.7}$ encryptions with the method of the guess and filter, so $T'_3 \approx 2^{110.86}$ memory access. The time complexity is $T'_1 + T'_3 + T'_4 = \frac{6}{26} \cdot 2^{111.78} + 2^{110.86} + 2^{108} \approx \mathbf{2^{111.51}}$ 26-round encryptions and $T'_2 = \mathbf{2^{115.78}}$ memory accesses.

We obtain that ϵ is 1 memory access with the method of the pre-construct hash table. Thence, the time complexity is $T'_1 + T'_4 = \frac{6}{26} \cdot 2^{111.78} + 2^{108} = \mathbf{2^{110.06}}$ 26-round encryptions and $T'_2 + T'_3 = 2^{115.78} + 2^{113.56} \approx \mathbf{2^{116.06}}$ memory accesses.

B Rectangle key recovery attack on GITF-128

The 19-round related-key rectangle distinguisher of GIFT-128 [16]:

$$\begin{aligned} \alpha &= 00000000000000a00000000060000000, \\ \delta &= 00200000000000000000004000002020. \end{aligned}$$

with:

$$\begin{aligned} \Delta K &= 8000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0002\ 0000, \\ \nabla K &= 0000\ 0000\ 0000\ 0000\ 0002\ 0000\ 0002\ 0000. \end{aligned}$$

The probability of this distinguisher is $P_d = 2^{109.626}$. As shown in Table 9, The unknown state difference is distributed in $\Delta X_2[23:20]$, $\Delta X_2[119:116]$ and $\Delta X_2[123:120]$. The subkey bits involved in the forward expansion round are $k_5[14]$, $k_1[14]$, $k_5[13]$, $k_1[13]$, $k_4[5]$, and $k_0[5]$.

Ji *et al.* [16] guessed all these 6-bit subkeys involved in ΔX_2. By applying filters $GS(X_2[23:20]) \oplus GS(\Delta X_2[23:20] \oplus X_2[23:20]) = \Delta Y_2[23:20] = 1000$, $GS(X_2[119:116]) \oplus GS(\Delta X_2[119:116] \oplus X_2[119:116]) = \Delta Y_2[119:116] = 0010$, and $GS(X_2[123:120]) \oplus GS(\Delta X_2[123:120] \oplus X_2[123:120]) = \Delta Y_2[123:120] = 0100$, they obtained 9 filtering bits, which are highlighted in red in Table 9. The parameters they used were $r_b = 9$, $m_b = m'_b = 6$, so $m^*_b = 0$; $r_f = 52$, $m_f = 34$, $m'_f = 0$, so $m^*_f = 34$; $s = 2$ and $h = 22$, yielding $y = \sqrt{s} \cdot 2^{n/2-r_b}/P_d = 2^{110.31}$ and $D = y \cdot 2^{r_b} = 2^{119.31}$.

Table 9: The 23-round related-key rectangle attack on GIFT-128. For round r, ΔX_r and ΔY_r are the input and output differences of the S-boxes, and ΔZ_r is the output difference of the linear layer. $r \in \{1, 2, ..., 23\}$

input	0000 0000 0000 0000 11?? ???? ???? ???? ???? ???? ???? ???? 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 11?? 0000 0000 0000 0000
ΔY_1	0000 0000 0000 0000 0100 00?0 000? 1000 ?100 0??0 00?? ?00? 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000
ΔZ_1	0000 11?? ?1?? 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ???? 0000 0000 0000 0000 0000
ΔX_2	0000 11?? ?1?? 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ???? 0000 0000 0000 0000 0000
ΔY_2	0000 0100 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1000 0000 0000 0000 0000 0000
ΔZ_2	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0110 0000 0000 0000 0000 0000 0000 0000
$\Delta X_3(\alpha)$	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0110 0000 0000 0000 0000 0000 0000 0000
:	... ...
$\Delta X_{22}(\delta)$	0000 0000 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 0010 0000 0010 0000
ΔY_{22}	0000 0000 ???? 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 ???1 0000 0000 0000 0000 0000 ???? 0000 ???? 0000
ΔZ_{22}	000? 0000 0000 0000 0000 0001 0000 0?0? ?000 0000 0000 0000 0000 ?000 0000 ?0?0 0?00 0000 0000 0000 0000 0?00 0000 0?0? 00?0 0000 0000 0000 0000 00?0 0000 ?0?0
ΔX_{23}	000? 0000 0010 0000 0000 0001 0000 0?0? ?000 0000 0000 0000 0000 ?000 0000 ?0?0 0?00 0000 0000 0000 0000 0?00 0000 0?0? 00?0 0000 0000 0000 0000 00?0 0000 ?0?0
ΔY_{23}	???? 0000 ???? 0000 0000 ???? 0000 ???? ???? 0000 0000 0000 0000 ???? 0000 ???? ???? 0000 0000 0000 0000 ???? 0000 ???? ???? 0000 0000 0000 0000 ???? 0000 ????
ΔZ_{23}	0?0? ?0?0 0?00 ?0?0 0?00 ?0?0 0?00 ?0?0 ?0?0 0?0? 00?0 0?0? 00?0 0?0? 00?0 0?0? 0?0? ?0?0 000? ?0?0 000? ?0?0 000? ?0?0 ?0?0 0?0? ?000 0?0? ?000 0?0? ?000 0?0?
output	0?0? ?0?0 0?00 ?0?0 0?00 ?0?0 0?00 ?0?0 ?0?0 0?0? 00?0 0?0? 00?0 0?0? 00?0 0?0? 0?0? ?0?0 000? ?0?0 000? ?0?0 000? ?0?0 ?0?0 0?0? ?000 0?0? ?000 0?0? ?000 0?0?

However, they overlooked the time consumption of partial encryption and decryption. To recalculate the time complexity based on their parameters: $T_0 = 4 \cdot D = 2^{121.31}$ encryptions, $T_1 = 2^{m'_b+m'_f} \cdot 4 \cdot D = 2^{127.31}$ partial encryptions, $T_2 = 2^{m'_b+m'_f} \cdot 2 \cdot D \cdot 2^{r^*_b} = 2^{126.31}$ memory accesses, and $T_3 = 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r_b+2r^*_f-2n} \cdot \epsilon = 2^{92.62}$ encryptions using the guess and filter method, where $\epsilon = 4 \cdot 2^2/23 \approx 2^{-0.52}$ encryptions. Additionally, $T_4 = 2^{m'_b+m'_f+k-m'_b-m'_f-h} = 2^{k-h} = 2^{106}$ encryptions. The total time complexity is $T_0 + T_1 + T_3 + T_4 = 2^{121.31} + 2^{127.31} \cdot 4/26 + 2^{92.62} + 2^{106} \approx \mathbf{2^{124.72}}$ encryptions and $T_2 = \mathbf{2^{126.31}}$ memory accesses.

We chose to guess only 4 subkey bits involved in the forward expansion round, namely $k_5[13]$, $k_1[13]$, $k_4[5]$, and $k_0[5]$. By applying the filter $GS(X_2[23:20]) \oplus GS(\Delta X_2[23:20] \oplus X_2[23:20]) = \Delta Y_2[23:20] = 1000$ and $GS(X_2[119:116]) \oplus GS(\Delta X_2[119:116] \oplus X_2[119:116]) = \Delta Y_2[119:116] = 0010$, we obtained 7 filtering bits. Therefore, $m'_b = 4$ and $m^*_b = 9 - 7 = 2$, the rest of the parameters are the same as [16]. The time complexities of this attack are $T_0 = 4 \cdot D = 2^{121.31}$ encryptions, $T_1 = 2^{m'_b+m'_f} \cdot 4 \cdot D = 2^{125.31}$ partial encryptions, $T_2 = 2^{m'_b+m'_f} \cdot 2 \cdot D \cdot 2^{r^*_b} = 2^{126.31}$ memory accesses, and $T_3 = 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^+_b 2r^*_f-2n} \cdot \epsilon = 2^{94.62}$ encryptions using the guess and filter method, where $\epsilon = 4 \cdot 2^2/23 \approx 2^{-0.52}$ encryptions. The total time complexity is $T_0 + T_1 + T_3 + T_4 = 2^{121.31} + 2^{125.31} \cdot 4/26 + 2^{94.62} + 2^{106} \approx \mathbf{2^{123.1}}$ encryptions, and $T_2 = \mathbf{2^{126.31}}$ memory accesses.

References

1. Banik, S., Bogdanov, A., Peyrin, T., Sasaki, Y., Sim, S.M., Tischhauser, E., Todo, Y.: SUNDAE-GIFT. In: Submission to the NIST Lightweight Cryptography project (2019)
2. Banik, S., Chakraborti, A., Inoue, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT-COFB. In: Cryptology ePrint Archive (2020)
3. 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: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer (2017). https://doi.org/10.1007/978-3-319-66787-4_16, https://doi.org/10.1007/978-3-319-66787-4_16
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. **2013**, 404 (2013)
5. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the serpent. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 340–357. Springer (2001). https://doi.org/10.1007/3-540-44987-6_21, https://doi.org/10.1007/3-540-44987-6_21
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991). https://doi.org/10.1007/BF00630563, https://doi.org/10.1007/BF00630563
7. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 1–18. Springer (2009). https://doi.org/10.1007/978-3-642-10366-7_1, https://doi.org/10.1007/978-3-642-10366-7_1
8. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: International workshop on cryptographic hardware and embedded systems. pp. 450–466. Springer (2007)
9. Chakraborti, A., Datta, N., Jha, A., Lopez, C.M., Nandi, M., Sasaki, Y.: LOTUS-AEAD and LOCUS-AEAD. In: Submission to the NIST Lightweight Cryptography project (2019)
10. Chakraborti, A., Datta, N., Jha, A., Nandi, M.: HYENA. In: Submission to the NIST Lightweight Cryptography project (2019)
11. Chen, L., Wang, G., Zhang, G.: MILP-based related-key rectangle attack and its application to GIFT, Khudra, MIBS. Comput. J. **62**(12), 1805–1821 (2019). https://doi.org/10.1093/comjnl/bxz076, https://doi.org/10.1093/comjnl/bxz076
12. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II. Lecture Notes in Computer Science, vol. 10821,

pp. 683–714. Springer (2018). https://doi.org/10.1007/978-3-319-78375-8_22, https://doi.org/10.1007/978-3-319-78375-8_22
13. Derbez, P., Euler, M., Fouque, P., Nguyen, P.H.: Revisiting related-key boomerang attacks on AES using computer-aided tool. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13793, pp. 68–88. Springer (2022). https://doi.org/10.1007/978-3-031-22969-5_3, https://doi.org/10.1007/978-3-031-22969-5_3
14. Dong, X., Qin, L., Sun, S., Wang, X.: Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 3–33. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_1, https://doi.org/10.1007/978-3-031-07082-2_1
15. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: Milp-based automatic search algorithms for differential and linear trails for speck. In: Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23. pp. 268–288. Springer (2016)
16. Ji, F., Zhang, W., Zhou, C., Ding, T.: Improved (related-key) differential cryptanalysis on GIFT. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12804, pp. 198–228. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_8, https://doi.org/10.1007/978-3-030-81652-0_8
17. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and serpent. In: Schneier, B. (ed.) Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1978, pp. 75–93. Springer (2000). https://doi.org/10.1007/3-540-44706-7_6, https://doi.org/10.1007/3-540-44706-7_6
18. Li, C., Wu, B., Lin, D.: Generalized boomerang connectivity table and improved cryptanalysis of gift. In: International Conference on Information Security and Cryptology. pp. 213–233. Springer (2023)
19. Liu, Y., Sasaki, Y.: Related-key boomerang attacks on GIFT with automated trail search including BCT effect. In: Jang-Jaccard, J., Guo, F. (eds.) Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11547, pp. 555–572. Springer (2019). https://doi.org/10.1007/978-3-030-21548-4_30, https://doi.org/10.1007/978-3-030-21548-4_30
20. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Information Security and Cryptology: 7th International Conference, Inscrypt 2011, Beijing, China, November 30–December 3, 2011. Revised Selected Papers 7. pp. 57–76. Springer (2012)
21. Sasaki, Y.: Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In: Inomata, A., Yasuda, K. (eds.) Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings. Lecture Notes in Computer

Science, vol. 11049, pp. 227–243. Springer (2018). https://doi.org/10.1007/978-3-319-97916-8_15, https://doi.org/10.1007/978-3-319-97916-8_15
22. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects: Revealing structural properties of several ciphers. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36. pp. 185–215. Springer (2017)
23. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. J. Cryptol. **21**(1), 131–147 (2008). https://doi.org/10.1007/s00145-007-9013-7, https://doi.org/10.1007/s00145-007-9013-7
24. Song, L., Qin, X., Hu, L.: Boomerang connectivity table revisited. application to SKINNY and AES. IACR Trans. Symmetric Cryptol. **2019**(1), 118–141 (2019). https://doi.org/10.13154/tosc.v2019.i1.118-141, https://doi.org/10.13154/tosc.v2019.i1.118-141
25. Song, L., Zhang, N., Yang, Q., Shi, D., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle attacks: a unified and generic framework for key recovery. In: Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I. pp. 410–440. Springer (2023)
26. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. IACR Trans. Symmetric Cryptol. **2021**(1), 269–315 (2021). https://doi.org/10.46586/tosc.v2021.i1.269-315, https://doi.org/10.46586/tosc.v2021.i1.269-315
27. Sun, L., Wang, W., Wang, M.: Improved attacks on GIFT-64. In: AlTawy, R., Hülsing, A. (eds.) Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13203, pp. 246–265. Springer (2021). https://doi.org/10.1007/978-3-030-99277-4_12, https://doi.org/10.1007/978-3-030-99277-4_12
28. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des (l) and other bit-oriented block ciphers. In: Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20. pp. 158–178. Springer (2014)
29. Wagner, D.A.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999). https://doi.org/10.1007/3-540-48519-8_12, https://doi.org/10.1007/3-540-48519-8_12
30. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying milp method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 648–678. Springer (2016)
31. Yu, Q., Qin, L., Dong, X., Jia, K.: Improved Related-Key Rectangle Attacks On GIFT. The Computer Journal p. bxad071 (07 2023). https://doi.org/10.1093/comjnl/bxad071, https://doi.org/10.1093/comjnl/bxad071
32. Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G.: Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and

GIFT. Des. Codes Cryptogr. **88**(6), 1103–1126 (2020). https://doi.org/10.1007/s10623-020-00730-1, https://doi.org/10.1007/s10623-020-00730-1