IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 1, pp. 156–184. DOI:10.46586/tosc.v2021.i1.156-184

Towards Key-recovery-attack Friendly Distinguishers: Application to GIFT-128

Rui Zong[1], Xiaoyang Dong[2] ✉, Huaifeng Chen[3], Yiyuan Luo[4,5], Si Wang[6] and Zheng Li[7,8]

[1] Verification & Validation Technology Corporation Limited, Shenzhen, China, zongr@vv.tech
[2] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China, xiaoyangdong@tsinghua.edu.cn
[3] The 6-th Research Institute of China Electronics Corporation, Beijing, China, chenhf@ncse.com.cn
[4] School of Computer Science and Engineering, Huizhou University, Huizhou, China
[5] Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu, China, luoyy@hzu.edu.cn
[6] China Telecom Corporation Limited, Guangzhou, China, wangs5.gd@chinatelecom.cn
[7] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[8] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, lizhengcn@bjut.edu.cn

Abstract. When analyzing a block cipher, the first step is to search for some valid distinguishers, for example, the differential trails in the differential cryptanalysis and the linear trails in the linear cryptanalysis. A distinguisher is advantageous if it can be utilized to attack more rounds and the amount of the involved key bits during the key-recovery process is small, as this leads to a long attack with a low complexity.
In this article, we propose a two-step strategy to search for such advantageous distinguishers. This strategy is inspired by the intuition that if a differential is advantageous only when some properties are satisfied, then we can predefine some constraints describing these properties and search for the differentials in the small set.
As applications, our strategy is used to analyze GIFT-128, which was proposed in CHES 2017. Based on some 20-round differentials, we give the first 27-round differential attack on GIFT-128, which covers one more round than the best previous result. Also, based on two 17-round linear trails, we give the first linear hull attack on GIFT-128, which covers 22 rounds. In addition, we also give some results on two GIFT-128 based AEADs GIFT-COFB and SUNDAE-GIFT.

Keywords: GIFT · Differential Trail · Linear Trail · Distinguisher Search Strategy · SUNDAE-GIFT · GIFT-COFB

1 Introduction

Differential cryptanalysis was proposed by Biham and Shamir in [BS91] and linear cryptanalysis was proposed by Matsui in [Mat94]. These are the most two famous methods for analyzing block ciphers. For both these two methods, the first step is to search out some valid distinguishers: the differential trails for differential cryptanalysis and the linear trails for linear cryptanalysis. When searching out a distinguisher: Input→ Output, the cryptanalysts extend some round functions backward from the Input and forward from the Output. After that, a key recovery attack is executed by guessing the involved key bits in the extended rounds. Usually, more extended rounds and less involved key bits are

Received: 2020-06-01 Revised: 2020-11-23 Accepted: 2021-01-23 Published: 2021-03-19

expected, as it leads to a more efficient attack with a longer length and a lower complexity, respectively. Both these two expectations are affected by the distinguisher's `Input` and `Output`.

To search for differential and linear trails efficiently, several automatic methods have been introduced and they have facilitated many fruitful works [KLT15, AK18, SS14, FWG+16, AST+17, KLT15, AK18]. Among them, the MILP based method and the SMT/SAT based method are most widely used. These methods perform well when the searching space is not too large and the model is very easy to implement. However, when searching for long trails, they may be inefficient as the search space is too large. What's more, the underlying solvers of these methods, such as `Gurobi` [Gur] in MILP and `STP` [STP] in SMT/SAT, are used as black boxes. We can not iteratively make adjustments according to their outputs. In addition to these new introduced methods, Matsui's branch-and-bound algorithm may be the most well known method to search for differential and linear trails. It also has two sides. On one side, this algorithm employs a depth-first with pruning search strategy and does guarantee to return all best trails for any initial value. On the other side, the cryptanalysts need to know what a good initial value might be before the search process.

In another direction, `GIFT` [BPP+17] lightweight block cipher is designed by Banik et al. [BPP+17], which includes two versions: `GIFT-64` and `GIFT-128`. Both of them have a 128-bit key size and inherit the design framework from `PRESENT` [BKL+07], but correcting the weakness in linear cryptanalysis. Specially, by a dedicated selection of the Sbox and the linear layer, it avoids the single active bit transitions for two consecutive rounds in both differential and linear trails, which stops the very effective linear hull attacks. `GIFT` gains more efficiency in various domains, *i.e.*, much smaller hardware implementation, faster encryptions and more secure against the known attacks. Moreover, the hardware cost of the `GIFT` Sbox is smaller than that of `PRESENT` Sbox and its key schedule is also much simpler, which makes it more lightweight. In addition, in the round based hardware implementation, the area of `GIFT` is even smaller than the recently proposed lightweight block ciphers `SKINNY` [BJK+16] and `SIMON` [BSS+15]. Due to the nice performance and high security level, many lightweight designs choose `GIFT` as their basic primitives, such as `SUNDAE-GIFT` [BBP+], `TGIF` [IKM+], `GIFT-COFB` [BCI+] and `Elastic-Tweak` [CDJ+19]. Notably, `GIFT-COFB` [BCI+] and `SUNDAE-GIFT` [BBP+] have been recently selected as the second round candidates of the ongoing NIST Lightweight Cryptography (LWC) Standardization Project [NIS].

Hence, it is quite important to understand the security level of `GIFT` and many evaluation results of `GIFT` have been published. In the `GIFT` [BPP+17] document, the designers claimed a 9-round differential with probability $2^{-44.415}$ and a 9-round linear with probability $2^{-49.997}$. At CT-RSA 2019, Zhu et al. [ZDY18] gave the first third-party cryptanalysis on `GIFT`, including a 19-round and a 22-round key-recovery attack on GIFT-64 and GIFT-128, respectively. Sasaki et al. [Sas18] improved the meet-in-the-middle (`MitM`) attack on 15-round `GIFT-64`. Zhou et al. [ZZDX19] gave the minimum number of differential/linear active Sboxes for up to 16/15 rounds and found the best differential/linear characteristics for up to 15/13 rounds of `GIFT-64`. Li et al. [LWZZ19] reduced the searching time to 4 seconds to obtain a 9-round `GIFT-64` differential trail with probability 2^{-42}. For `GIFT-128`, they found a 21-round differential trail with probability $2^{-126.415}$. Due to too many active bits on the output of the 21-round differential trail, they utilized the last 20 rounds of the 21-round differential trail as the distinguisher to attack 26-round `GIFT-128`. Later, the 26-round attack is improved by Ji et al. [JZZD20]. In [JZD19], Ji et al. improved Matsui's algorithm by using three new methods. They claimed the highest probability of the differential trails of `GIFT-128` up to 19-round and the highest probability of the best linear trails up to 10-round. They presented a 19-round differential trail and a 10-round linear trail, both with the highest probability. What's more, they also claimed that a 20-round

GIFT-128 differential trail with probability $2^{-121.415}$ and a 21-round one with probability $2^{-126.415}$ were found. In [LLL+19], the authors found another 21-round differential trail on GIFT-128 with probability $2^{-126.415}$. All the above results are under the single-key setting. In related-key setting, Liu and Sasaki [LS19] gave a 23-round and a 21-round boomerang attack on GIFT-64 and GIFT-128, respectively. Chen et al. [CWZ] gave a 23-round related-key rectangle attack on GIFT-64, and Zhao et al. [ZDM+19] improved it to a 24-round attack. Ji et al. [JZZD20] improved the related-key attack on GIFT-128 to 23 rounds. In the GIFT document, the designers claimed no security under the related-key setting. In [LWZZ19], the authors study the influences between the solution and construction of MILP models, and give good results on PRESENT, GIFT-64 and GIFT-128. However, the exact relationship is still ambiguous. To study the dependence of key bits, there are two articles [HV18, Sas18] published. In [HV18], the algorithm can be used to estimate the key dependent correlation distribution of a linear approximation to facilitate advanced linear attacks and also search for a large number of trails by converting the diffferential/linear trails into paths in a multistage graph. However, no results on GIFT-128 are given in this article. In [Sas18], the algorithm studying the dependence between key bits is used to facilitate advanced meet-in-the-middle attack on GIFT-64.

Our Contributions

We propose a two-step strategy for searching advantageous distinguishers which can lead to long attacks with a small number of involved key bits. The overall concept is inspired by that, when mounting an attack, the two expectations: 1) more rounds are extended by the distinguisher and 2) less key bits are involved during the key-recovery process, are both determined by the distinguisher's Input and Output. In the first step, we specify the Input(Output) values in a set called the InitialSet which need to satisfy the following two conditions: 1) a distinguisher with an input (output) from the InitialSet can be extended by many rounds at the top (the bottom), to lead a long attack, 2) the amount of involved key bits in the extended rounds is small, to lead a low attack complexity. In the second step, we only search for advantageous distinguishers with input and output values from the InitialSet. This provides two benefits: the searching space is reduced and an efficient attack can be mounted once a distinguisher is found out. As a first application, we use it to search differential and linear trails of GIFT-128 and give cryptanalysis results on GIFT-128 and two GIFT-128-based proposals: SUNDAE-GIFT and GIFT-COFB. In more detail, we achieve the following:

a. We utilize the MILP technique and revisit Matsui's branch-and-bound algorithm to implement a two-step strategy of searching for advantageous differential and linear distinguishers. In the first step, we construct some MILP models describing the difference (linear mask) propagation in a block cipher's round function and marking the involved key bits. This step can output the InitialSet including all possible values of the Input(Output) that most rounds can be added at the top (the bottom) of a distinguisher while the involved key bits are the least. In the second step, we revisit Matsui's algorithm to search out some advantageous distinguishers that can lead to efficient attacks. The initial value of this step is only chosen from the InitialSet.

   These two steps make full use of the good sides of both the MILP method and Matsui's algorithm, while avoiding their bad sides. Usually, the searching space of the first step is small, the MILP method can be very efficient to searching the solutions. In the second step, the problem that Matsui's algorithm needs some good initial values is solved by only searching in a smaller space limited by the InitialSet.

b. We apply our strategy to analyze GIFT-128 and search for both differential and linear trails. For differential cryptanalysis, we find a 20-round differential that can be used to attack 27-round GIFT-128. This is the first 27-round attack on GIFT-128, which covers one more round than the other results. Although some 21-round differential are found in some other works, they are all weaker than our 20-round differential trail. And this also proves the validity of our search strategy. For linear cryptanalysis, we find a 17-round linear hull and give the first linear key-recovery attack on GIFT-128, which covers 22 rounds.

c. We mount linear cryptanalysis on the most two notable GIFT-based proposals: GIFT-COFB and SUNDAE-GIFT. For GIFT-COFB, we analyze the security of its 15-round GIFT-128 version; for SUNDAE-GIFT, we analyze the security of its 16-round GIFT-128 version.

Remarks. In [ZSCH18], the authors show how to tweak the objective functions of the MILP models for finding better trails, with some constraints derived from the bounding condition of Matsui's algorithm. The key different point of this work and ours is that the MILP technique is utilized to search for trails, while for our strategy, the MILP technique is used to search for the InitialSet and the Matsui's algorithm is to search for specific trails.

Table 1: Summary of differential and linear trails of GIFT-128, trail means one single differential(linear) trail, trails means differential(linear hull), the number in the Attack column means the length of the attack based on the given distinguisher.

	Type	Rounds	Probability	Attack	Ref
Differential	trail	9	$2^{-44.415}$	–	[BPP+17]
	trail	18	2^{-109}	22	[ZDY18]
	trails	20	$2^{-120.245}$	26	[JZZD20]
	trail	20	$2^{-121.415}$	26	[LWZZ19]
	trail	20	$2^{-121.415}$	–	[JZD19]
	trails	20	$2^{-121.813}$	27	Section 5
	trail	21	$2^{-126.415}$	–	[JZD19, LLL+19, LWZZ19]
Linear	trails	9	$2^{-49.997}$	–	[BPP+17]
	trail	10	2^{-52}	–	[JZD19]
	trails	15	2^{-109}	22	Section 6

Table 2: Cryptanalysis results of GIFT-128. DC stands for differential cryptanalysis; LC stands for linear cryptanalysis; SK stands for single-key setting; RK stands for related-key setting; Boomerang stands for boomerang attack. Note that there is no security claim of GIFT under the related-key setting.

Rounds	Approach	Setting	Time	Data	Memory	Ref
22	DC	SK	2^{120}	2^{120}	2^{86}	[ZDY18]
26	DC	SK	$2^{124.415}$	2^{109}	$2^{124.415}$	[LWZZ19]
26	DC	SK	$2^{123.245}$	$2^{123.245}$	2^{109}	[JZZD20]
27	DC	SK	$2^{124.83}$	$2^{123.53}$	2^{80}	Section 5
22	LC	SK	2^{117}	2^{117}	2^{78}	Section 6
21	Boomerang	RK	$2^{126.6}$	$2^{126.6}$	$2^{126.6}$	[LS19]
23	Rectangle	RK	$2^{126.89}$	$2^{121.31}$	$2^{121.63}$	[JZZD20]

Table 3: Cryptanalysis results of SUNDAE-GIFT and GIFT-COFB.

Scheme	Rounds	Approach	Setting	Time	Data	Memory	Ref
GIFT-COFB	15	LC	SK	$2^{90.7}$	2^{62}	2^{96}-bit	Section 7
SUNDAE-GIFT	16	LC	SK	$2^{91.2}$	2^{60}	2^{96}-bit	Section 8

Table 4: The notations of GIFT

P	:	the plaintext,
C	:	the ciphertext,
X_i	:	the input state of the i-th round fuction,
X_i^S	:	the state after the SubCells operation,
X_i^P	:	the state after the PermBits operation,
X_i^K	:	the state after the AddRoundKey operation,
$X[i]$	:	the i-the bit of X, $X[0]$ is the rightmost bit,
RK_i	:	the i-th round key,
k_i	:	the i-th 16-bit word of the master key,
k_i^j	:	the j-th bit of k_i,
f^{-1}	:	the inverse operation of f,
ΔX	:	the difference in the state X,
$X[j \cdots k]$	:	j^{th} bit, $\cdots$, k^{th} bit of state X, note that $X[0]$ is the LSB of X.
$X[j \sim k]$	:	the successive bit from the j^{th} bit to the k^{th} bit of state X,
$\ggg i$	:	an i-bit right rotation within a 16-bit word.
RK_i'	:	equal to $\texttt{PermBits}^{-1}(RK_i)$.
$X_i'^K$	:	equal to $\texttt{PermBits}^{-1}(X_{i+1})$
$X_i'^P$	:	equal to X_{i+1}

2 Preliminaries

2.1 GIFT-128

GIFT [BPP+17] lightweight block cipher was proposed by Banik et al. at CHES 2017. Similar to PRESENT, GIFT adopts an SPN structure with an Sbox layer and a bitwise permutation layer. The authors define two versions of GIFT, namely, GIFT-64 and GIFT-128 according to the block size. Both versions have a 128-bit key. The round numbers of them are 28 and 40, respectively. Since our paper is mainly about GIFT-128, so we omit the description on GIFT-64. Table 4 gives some notations used throughout this paper. In each round function, three operations are performed in sequence, *i.e.*, SubCells, PermBits and AddRoundKey:

- SubCells : Apply 32 4-bit Sboxes in parallel to every nibble of the internal state of GIFT-128. The Sbox is given in Table 5.

Table 5: The Sbox of GIFT-128

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

- PermBits : Linear bit permutations $b_{P(i)} \leftarrow b_i,\ \forall i \in \{0, 1, ...127\}$, where the $P(i)$s are

$$P_{128}(i) = 4\lfloor \frac{i}{16} \rfloor + 32\left(3\lfloor \frac{i \bmod 16}{4} \rfloor + (i \bmod 4) \bmod 4\right) + (i \bmod 4).$$

- AddRoundKey : The round keys RK is 64-bit, which is generated by the key state. Let $RK = U \| V = u_{31}...u_0 || v_{31}...v_0$.

The round key is Xored to the state $b_{127}...b_0$ in the following way:

$$b_{4i+2} \leftarrow b_{4i+2} \oplus u_i,\ b_{4i+1} \leftarrow b_{4i+1} \oplus v_i,\ \forall i \in \{0, ..., 31\}.$$

A single bit "1" and a 6-bit constant C are Xored into the internal state $b_{127}...b_0$ at positions 127, 23, 19, 15, 11, 7 and 3 respectively.

`The Key schedule.` The 128-bit master key is initialized as $K = k_7\|k_6\|...\|k_0$, where $|k_i| = 16$. For `GIFT-128`, the round key RK is $RK = U\|V = k_5\|k_4\|k_1\|k_0$. The key state is updated as follows,

$$k_7\|k_6\|...\|k_0 \leftarrow (k_1 \ggg 2)\|(k_0 \ggg 12)\|...\|k_3\|k_2.$$

For more details of `GIFT-128`, we refer the readers to [BPP+17].

2.2 `GIFT-COFB` and `SUNDAE-GIFT`

`GIFT-COFB` [BCI+] and `SUNDAE-GIFT` [BBP+] are among of the 2nd round AE (authenticated encryption) candidates of the ongoing NIST lightweight standardisation process. `GIFT-COFB` uses a mode of operation `COFB` proposed by Chakraborti et al. at CHES 2017 [CIMN17]. `SUNDAE-GIFT` adopts the mode of operation `SUNDAE` introduced by Banik et al. [BBLT18] at ToSC 2018. The underlying block cipher is also `GIFT-128` [BPP+17]. We show the specifications of `GIFT-COFB` and `SUNDAE-GIFT` in Appendix A and B .

3 The Strategy for Searching Differential Trails

Our strategy can be used to search for both differential trails and linear trails. To facilitate the narrative, we first introduce the process of searching for differential trails. After that, we give the process of searching for linear trails and list the different points between these two processes.

When mounting a differential key-recovery attack after searching out a valid differential $\Delta in \rightarrow \Delta out$, the cryptanalysts usually have two expectations:

1. The amount of the involved key bits during the key-recovery process is less as it will lead to a lower attack complexity,

2. More rounds can be extended at the top and the bottom of the differential without activating all the bits of the plaintext and ciphertext, as it will lead to a longer attack.

A differential is advantageous if both two expectations are achieved and the value of Δin and Δout plays a decisive role. Motivated by this, we propose a two-step strategy that only concentrates on searching for the advantageous differentials. The first step is to find out the values of the Δin and Δout that can satisfy the above two expectations. All possible values of Δin and Δout are stored in a set called the `InitialSet`. After that, the second step is to search for differentials whose input and output difference is only from the `InitialSet`. This provides two important benefits: 1). the searching space is greatly reduced as the potential distinguisher's input and output difference is limited by the `InitialSet`, 2) the attack can cover more rounds with a lower complexity once a advantageous differential is found.

The first step utilizes the MILP techniques and the second step is a revisit of Matsui's branch-and-bound search algorithm. We give a detailed introduction of these two steps in the following.

3.1 The MILP Model Searching for the InitialSet

After searching out a valid differential, we try to extend some round functions at its top and bottom and check the state to avoid all bits are activated. In this step, what matters is usually the state's activeness but not the potential specific difference value. For example, if an r-round GIFT-128 differential with output difference Δout is found and Δout has one active bit, the input of the first added round has only one active Sbox. Suppose the Sbox's input difference is 0001, according to the difference distribution table (DDT), it can propagate to 8 output difference values: $\{0101, 0110, 1000, 1001, 1010, 1011, 1100, 1111\}$. We would mark all 4 output bits as uncertain bits as they have different activeness in different output differences.

Constraints describing the activeness of the Sbox's input and output difference. We use 0 to denote an inactive bit and 1 to denote an active bit or an uncertain bit. The activeness of the input and output difference can be denoted as some 8-bit points. And the propagation rule is as follows: the 4 output bits are 1 as long as the input has at least one active bit. We use 8 boolean variables $(x[3], x[2], x[1], x[0], y[3], y[2], y[1], y[0])$ denoting the Sbox's input and output difference. The rule can be constrained by the following inequalities, and there are 20 inequalities for each Sbox in total.

$$\begin{aligned} &y[i] - x[3] \geq 0, \\ &y[i] - x[2] \geq 0, \\ &y[i] - x[1] \geq 0, \\ &y[i] - x[0] \geq 0, \\ &x[3] + x[2] + x[1] + x[0] - y[i] \geq 0. \end{aligned} \qquad 0 \leq i \leq 3$$

Constraints describing the forward round function. We use 128 boolean variables, $x_r[i], 0 \leq i \leq 127$, describing the activeness of input state of the r-th extended round. Since the PermBits operation is a linear bitwise permutation, no extra variables are needed to describe it. For example, for a Sbox in the r-th round, the input $x_r[i3, i2, i1, i0]$ is related to $x_{r+1}[\texttt{PermBits}[i3, i2, i1, i0]]$. We construct inequalities describing all relations between the state in two consecutive rounds.

Other constraints. When extended at the bottom of a differential, the differential's output difference, denoted by x_0, should have at least one active bit. And the output difference of the last extended round, denoted by x_r, should have at least one inactive bit. These two constraints can be described by the following two inequalities:

$$\sum_{i=0}^{127} x_0[i] \geq 1, \sum_{i=0}^{127} x_r \leq 127.$$

Till now, we construct the MILP model describing the state's activeness in the added r rounds at the bottom of the distinguisher. We start to solve the MILP model with $r = 1$, and if the r-round model is feasible, we construct the $r+1$-round model and see whether it is feasible... Then the feasible model with the largest r tell us that we can extend at most r rounds at the bottom of a differential. Since we just want to see the model is infeasible or not, the objective function can be optional.

The MILP model describing the rounds extended at the top is almost the same as above. We don't repeat the detailed description. We just point out the only difference when extending backward at the top of the distinguisher: the number of active bits in the last added round should be should be less than 128, to avoid a full-codebook attack.

Adding the involved key bits. We already know how many rounds can be extended both at the top and the bottom of the differential and the extended round number is denoted by R. Next we try to find out which input and output difference values can be extended by R rounds, meanwhile, the amount of involved key bits in the R rounds is small. These constraints are very easy to construct.

For each extended round, we add 64 boolean variables $k[i], 0 \leq i \leq 63$ denoting the 64-bit round key RK for `GIFT-128`. $k[i] = 0$ represents $RK[i]$ is not involved and $k[i] = 1$ otherwise. For example, in the encryption direction, $RK[i]$ will be Xored into the jth Sbox. Suppose the input-output difference bits is $(x[3], x[2], x[1], x[0], y[3], y[2], y[1], y[0])$, then if the jth Sbox is active, we have to guess $RK[i]$. Hence, we have $k[i] - x[l] \geq 0$ for $0 \leq l \leq 3$.

In the decryption direction, the round keys are Xored into the state after `PermBits`. However, in the key-recovery process, the round keys are transformed as RK' (only key bit positions are change from RK) and Xored into the state after Sbox layer. Hence, the constraints on RK' are similar to those in encryption phase. Note that there are dependencies between the round keys, two round key bits share a same variable if they are derived from the same master key bit.

The objective function Since this new MILP model is to find output difference values that lead to a small amount of involved key bits. The objective function is as follows:

$$Minimize \sum_{j=r_1}^{r_R} \sum_{i=0}^{63} k_j[i].$$

Using this model, we will know the least amount of involved key bits and get all corresponding input and output difference values. We store all these difference values in the `InitialSet`.

3.2 A Revisit of Matsui's Algorithm Searching for Advantageous Differentials

In this step, we try to find out the longest differentials with a high probability, whose input and output difference from the `InitialSet`. The search process as shown in Algorithm 1 is a revisit of Matsui's branch-and-bound algorithm which adopts a depth-first with pruning strategy. This method guarantees to return all best trails for any initial value. As we already get the advantageous initial values in the first step, which are stored in the `InitialSet`, we are confident to search out some advantageous differentials.

We start the second step with the optimal elements of the `InitialSet`, *i.e.*, the greatest number of rounds can be extended and the involved key bits are the least, and set them as the input and output value of the potential differential trails. However, when given an input and output value, we can not determine the existence of valid distinguishers in advance before the searching process of the second step. Our strategy in this step is that, we start searching process with the optimal choices from the `InitialSet`, if we can not get advantageous distinguishers, we go on searching with the sub-optimal ones until we find out some valid distinguishers. This strategy ensures that the distinguishers searched out can lead to the best key-recovery attacks.

For `GIFT-128`, every round function has 32 Sboxes, the searching space will be very large when no constraints about the number of active S-boxes are set, as the algorithm needs to traverse all possible differentials. Due to this, we set up an upper bound of active S-boxes in each round function to be 4. And also, we set a lower bound of the probability of the differential characteristics that will be recorded to be 2^{-128}. When the searching process covers r rounds, it outputs the qualified results. Note that r is decided somewhat

Algorithm 1: The Search Algorithm

Procedure 1: Initialization

1. Initialize t as the upper bound of the number of active Sboxes in each round, r as the number of searched rounds and *prob* as the lower bound of probability of qualified differential trails.
2. Choose the input difference, ΔX_0, from the `InitialSet`, set the initial probability as $p_0 = 1$.

Procedure 2: Recursive Search - *search round* $i, i \geq 1$

4. For each $(i-1)$-round differential trail, we get the output difference ΔX_{i-1} and the corresponding probability p_{i-1}.

For each Sbox in round i, try all of its possible output differences.

Check the overall propagation probability p and the number of active SBoxes t_e.

Continue to the next Sbox only when $p \geq prob$ and $t_e \leq t$.

5. Get qualified ΔX_i and $p_i = p_{i-1} \cdot p_e$.

p_e is the propagation probability from ΔX_{i-1} to ΔX_i.

6. If $i < r$, go to *search round* $i + 1$.
7. If $i = r$ and $p_i \geq prob$, go to Procedure 3.

Procedure 3: Record Qualified Results

8. If ΔX_r is also in `InitialSet`, record it.

The record format is $(\Delta X_0, p_0), (\Delta X_1, p_1), \cdots, (\Delta X_r, p_r)$.

9. Summarize all possible differentials and compute their overall probability.

The differential record format is $(\Delta X_0, \Delta X_r, p_r)$.

experimentally, for example, we already know the longest known `GIFT-128` differential tails are less than 22 rounds, then we run the model for r=21,20,19... until we find a valid trail to launch the attack.

Remarks. There are other options to perform the second stage, such as MILP-based or SAT-based differential search. However, with the differential searching experiences [ZDY18, JZD19, JZZD20] on `GIFT-128`, when searching for longer rounds (e.g., 19, 20, 21 rounds etc.) with the same constrained number of active S-boxes in each round, the branch and bound method is more efficient than those automatic-tool based method. Hence, for the attack on `GIFT-128`, we mainly use Matsui's method to perform the second stage.

For `GIFT-COFB` [BCI+] and `SUNDAE-GIFT` [BBP+], due the data limitation, only short trials with high probabilities can be used to perform key-recovery attack. The search space is small, the tool-based model is also efficient. Hence, to search those trails we can use tool-based method to replace Matsui's method.

On the balance between the first stage and the second stage. The first stage is very fast. It only outputs some candidate initial values for the second stage. In the first stage, we collect those initial values that do not fully activate the plaintext and ciphertext when appending 4 rounds and 3 rounds at the top and bottom of the distinguisher. Then, we sort those initial values by the number of key bits involved in the extended rounds and store them in `InitialSet`. In the second stage, we use large computing resources (Dell PowerEdge with about 64 cores in our platform) to enumerate the initial values in `InitialSet` from the best one, until a distinguisher is found. We start the second step with the best elements from `InitialSet`, if failed, we continue searching with some second-best elements. But for other ciphers, the situation may be different.

On the output of the second stage. The second stage outputs some distinguishers conforming with certain initial values from the `InitialSet`, with probability larger than

2^{-n}. However, we can not guarantee those distinguishers can be used to perform key-recovery attacks by extending 4 rounds and 3 rounds at the top and the bottom of the distinguisher. Look at the constraints in the first stage, we only restrict that the difference bits of plaintext and ciphertext are not fully activated. Then, use the number of key bits involved in the extended rounds to sort those candidate initial values in `InitialSet`. Hence, the output distinguishers by the second stage may not lead a valid attack because of other factors, for example, the probability of the distinguisher may be too low to work, or the number of active bits after appending 4 rounds and 3 rounds are too many to use. If these cases happen, we may either tweak the distinguishers by peeling off one round to increase the probability, or extend fewer rounds on the top or the bottom to enjoy fewer active bits in plaintext or ciphertext. Therefore, our model may fail its target in certain situations. However, the distinguishers given in the second stage under the guide from the first stage are likely to work, for example the 27-round key-recovery attack on `GIFT-128` in Section 5. Even though for certain cases, some tweaks have to be applied to the given distinguisher, it still preserves some advantages in the key-recover attack. For example, if we have to extend fewer rounds at the top and bottom (e.g., 3 and 2 rounds), the number of active bits are likely to be small, since when extending 4 rounds and 3 round at the top and bottom, the input and output states are not fully activated yet.

4 The Strategy for Searching Linear Trails

The two-step strategy can also be used to search for advantageous linear trails. As in a linear key-recovery attack, the process is executed by first searching out some good linear trails as the distinguisher and then adding some rounds at the top and the bottom of it, which is very similar with that in the differential key-covery attack. The overall procedure is very similar with that used for searching differentials in Section 3.

Due to the fact that the interplay of the Sbox and the linear layer in `GIFT` is well-crafted to resist linear cryptanalysis, we can not search out long linear trails when t (the upper bound of linear active Sboxes in each round in the Matsui's branch and bound algorithm) is not larger than 4. We set t as 5 and the lower bound of the probability of qualified linear trails as 2^{-128}.

5 Differential Cryptanalysis of `GIFT-128`

Following the strategy in Section 3, we first get the InitialSet and find that at most 4 rounds can be extended at the top and at most 3 rounds can be extended at the bottom of a distinguisher. The least amount of the involved key bits is 62, we find 2436 optimal elements of the InitialSet. For the best solutions, there are only one or two active bits in the input or the output. However, we can not search out long valid differentials with them when executing further in the second step. Instead, we find out some differentials with the second-best solutions. We list two of them in Table 6.

Table 6: Solutions of the first step

	Best Solution	Second-best Solution
Input	0000000000000004000000200000000	0000000000000000000000000000000a0
Output	0000000000000000000000000000000c	0000000000000000000000002200000011
Involved Key Bits	62 $k_0, k_1^{15,14,5-0}, k_2^{6,4-0}, k_4$ $k_5^{12-4}, k_6^{3-0}, k_7^{14,12,2,0}$	80 $k_0^{11-0}, k_1, k_2^{6,4,2,0}, k_3^{15-12,10,8-4}, k_4^{7-0}$ $k_5, k_6^{14,12,10,8}, k_7^{15-12,7-4,2,0}$
Top	4 rounds	4 rounds
Bottom	3 rounds	3 rounds

In addition, we also have some other observations. When the length of the potential distinguisher and the number of the extended rounds are fixed, the involved key bits are the same, it is not influenced by the starting round index of the distinguisher. For example, the least involved key bits of a potential 27-round attack based on a 20-round distinguisher by extending 4 rounds and 3 rounds at the top and the bottom is 62, it is the same when the attack occupies round 0 27, 1 28, 2 29, 3 30, 4 31. What's more, we also find that three rounds can be extended at the bottom as long as the output is only inactive in the first half or the second half bits.

We finally use some 20-round advantageous differentials with probability $2^{-121.83}$ to attack 27-round GIFT-128. The time complexity is $2^{124.83}$, the data complexity is $2^{123.53}$, the memory complexity is 2^{80}-bit space.

5.1 The 20-round Differentials

We search out 8 valid differentials and list them in Table 7. For these differentials, the involved key bits are the same and each contains 8 valid differential trails. For example, there are 8 differential trails in the first differential: 2 trails are with probability of 2^{-124}; 4 trails are with probability of 2^{-125}; 2 trails are with probability of 2^{-126}. We also list two of them in Table 8, they are both with probability of 2^{-124}.

Table 7: The 20-round differentials of GIFT-128

	Input Difference	Output Difference	$\Pr(\log_2)$
1	0000000000000000000000000000a0	00000000000000000000002200000011	-121.83
2	0000000000000000000000000000a0	00000000000000002000000210000001	-121.83
3	0000000000000000000000000000d0	00000000000000002000000210000001	-122.83
4	0000000000000000000000000000d0	00000000000000000000002200000011	-121.83
5	000000000000000000000000000a00	00000000000000002000002001000010	-121.83
6	000000000000000000000000000a00	00000000000000000000002200000011	-121.83
7	000000000000000000000000000d00	00000000000000002000002001000010	-122.83
8	000000000000000000000000000d00	00000000000000000000002200000011	-122.83

Table 8: Two 20-round differential trails with probability 2^{-124}

r	ΔX_r	Pr	ΔX_r	Pr
0	0000000000000000000000000000a0	0	0000000000000000000000000000a0	0
1	00000001000000000000000000000000	-2	00000001000000000000000000000000	-2
2	08000000000000000000000000000000	-5	08000000000000000000000000000000	-5
3	20000001000000000000000000000000	-7	20000001000000000000000000000000	-7
4	40400000202000000000000000000000	-12	40400000202000000000000000000000	-12
5	50500000000000005050000000000000	-20	50500000000000005050000000000000	-20
6	0000000000000000000000a000a000	-32	0000000000000000000000a000a000	-32
7	00000000000000000000001100000000	-36	00000000000000000000001100000000	-36
8	00000800000008000000000000000000	-42	00000800000008000000000000000000	-42
9	02020000010100000000000000000000	-46	02020000010100000000000000000000	-46
10	00000000505000000000000050500000	-56	00000000505000000000000050500000	-56
11	000000000000000000000000a000a0	-68	000000000000000000000000a000a0	-68
12	00000011000000000000000000000000	-72	00000011000000000000000000000000	-72
13	08000000080000000000000000000000	-78	08000000080000000000000000000000	-78
14	20200000101000000000000000000000	-82	20200000101000000000000000000000	-82
15	50500000000000005050000000000000	-92	50500000000000005050000000000000	-92
16	00000000a000a000000000000000000	-104	00000000a000a000000000000000000	-104
17	00000000000000000011000000000000	-108	00000000000000000011000000000000	-108
18	000000000000c000000000000001000	-114	000000000000c000000000000003000	-114
19	00000000000000000000000000080008	-120	00000000000000000000000000080008	-120
20	00000000000000000000002200000011	-124	00000000000000000000002200000011	-124

Table 9: The differential attack on 27-round GIFT-128

ΔP	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
ΔX_1^S	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--
	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--
ΔX_1^P	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
RK_1	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
ΔX_2^S	-**-	--**	*--*	**--	-**-	--**	1--*	*1--	-**-	--**	*--*	**--	-**-	--**	*--*	**--
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
ΔX_2^P	****	****	****	****	----	----	----	----	****	11**	****	****	----	----	----	----
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
RK_2	11	11	11	11	--	--	--	--	11	11	11	11	--	--	--	--
	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
ΔX_3^S	*---	-*--	--*-	---*	----	----	----	----	*---	-1--	--*-	---*	----	----	----	----
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
ΔX_3^P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	****	----	*1**	----	----	----	----	----
RK_3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
	--	--	--	--	--	--	--	--	11	--	11	--	--	--	--	--
ΔX_4^S	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	1---	----	--1-	----	----	----	----	----
ΔX_4^P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	1-1-	----
RK_4	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
Input	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	----	----	----	----	----	1-1-	----	
Output	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	--1-	--1-	----	----	----	----	----	----	---1	---1
RK'_{25}	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
	--	--	--	--	--	--	11	11	--	--	--	--	--	--	11	11
$\Delta X'^P_{25}$	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	****	****	----	----	----	----	----	----	****	****
ΔX_{26}	----	----	----	----	----	*--*	----	*--*	----	----	----	----	----	**--	----	**--
	----	----	----	----	----	-**-	----	-**-	----	----	----	----	----	--**	----	--**
RK'_{26}	--	--	--	--	--	11	--	11	--	--	--	--	--	11	--	11
	--	--	--	--	--	11	--	11	--	--	--	--	--	11	--	11
$\Delta X'^P_{26}$	----	----	----	----	----	****	----	****	----	----	----	----	----	****	----	****
	----	----	----	----	----	****	----	****	----	----	----	----	----	****	----	****
ΔX_{27}	----	*-*-	----	*-*-	----	*-*-	----	*-*-	----	-*-*	----	-*-*	----	-*-*	----	-*-*
	----	*-*-	----	*-*-	----	*-*-	----	*-*-	----	-*-*	----	-*-*	----	-*-*	----	-*-*
RK'_{27}	--	11	--	11	--	11	--	11	--	11	--	11	--	11	--	11
	--	11	--	11	--	11	--	11	--	11	--	11	--	11	--	11
$\Delta X'^K_{27}$	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****
	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****
ΔC	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*
	--	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*

Table 10: Key bits involved of 27-round differential cryptanalysis of GIFT-128

1	$k_5^{15}k_1^{15}$	$k_5^{14}k_1^{14}$	$k_5^{13}k_1^{13}$	$k_5^{12}k_1^{12}$	$k_5^{11}k_1^{11}$	$k_5^{10}k_1^{10}$	$k_5^{9}k_1^{9}$	$k_5^{8}k_1^{8}$	$k_5^{7}k_1^{7}$	$k_5^{6}k_1^{6}$	$k_5^{5}k_1^{5}$	$k_5^{4}k_1^{4}$	$k_5^{3}k_1^{3}$	$k_5^{2}k_1^{2}$	$k_5^{1}k_1^{1}$	$k_5^{0}k_1^{0}$
2	$k_7^{15}k_3^{15}$	$k_7^{14}k_3^{14}$	$k_7^{13}k_3^{13}$	$k_7^{12}k_3^{12}$					$k_7^{7}k_3^{7}$	$k_7^{6}k_3^{6}$	$k_7^{5}k_3^{5}$	$k_7^{4}k_3^{4}$				
3									$k_0^{3}k_4^{7}$		$k_0^{1}k_4^{5}$					
4																
25							$k_4^{2}k_0^{10}$	$k_5^{14}k_0^{2}$							$k_4^{0}k_0^{8}$	$k_5^{12}k_0^{0}$
26						$k_6^{14}k_3^{10}$ $k_6^{10}k_3^{6}$		$k_7^{2}k_2^{6}$ $k_7^{14}k_2^{2}$						$k_6^{12}k_3^{8}$ $k_6^{8}k_3^{4}$		$k_7^{0}k_2^{4}$ $k_7^{12}k_2^{0}$
27		$k_0^{11}k_5^{11}$ $k_0^{7}k_5^{7}$		$k_1^{5}k_4^{7}$ $k_1^{1}k_4^{3}$		$k_0^{10}k_5^{10}$ $k_0^{6}k_5^{6}$		$k_1^{4}k_4^{6}$ $k_1^{0}k_4^{2}$		$k_0^{9}k_5^{9}$ $k_0^{5}k_5^{5}$		$k_1^{3}k_4^{5}$ $k_1^{15}k_4^{1}$		$k_0^{8}k_5^{8}$ $k_0^{4}k_5^{4}$		$k_1^{2}k_4^{4}$ $k_1^{14}k_4^{0}$

5.2 The 27-round Differential Key-recovery Attack

Based on the first 20-round differential in Table 7:

$$(0000000000000000000000000000000a0 \rightarrow 00000000000000000000002200000011),$$

we attack 27-round GIFT-128 by extending 4 rounds at the top and 3 rounds in the bottom. The whole attack details are shown in Table 9. For better readability, we use - denoting the inactive bits in the state. The attack procedure is as follows.

Data collection.

1. In GIFT, there is no whitening key at the beginning, we can construct structures before the first round key involved at X_1^P. We set $X_1^P[63 \sim 0]$ as constant and traverse all values of $X_1^P[127 \sim 64]$ as one structure. There are 2^{64} elements in each structure, providing $2^{64\times 2-1} = 2^{127}$ pairs.

2. Construct 2^t structures, we get 2^{127+t} message pairs.

3. For each message, we can get the plaintext P by applying the $\texttt{PermBits}^{-1}$ and $\texttt{SubCells}^{-1}$ operations. After that, we can get the corresponding ciphertext by encrypting the plaintext.

Key Recovery. In the key-recovery process, we only care for which the ciphertext difference conforms to the difference pattern of ΔC as shown in Table 9. ΔC has 64 inactive bits, this will perform a 64-bit filter, about $2^{t+127-64} = 2^{t+63}$ pairs remain. The involved key bits during the key recovery process is given in Table 10. To simplify the description of the key guessing procedure, we move the subkeys involved in the last two rounds to the place between the SubCells and PermBits operation. Next we give the detailed procedure of counting right pairs using a guess and filter approach.

(a) Guess the involved key bits in RK_1 and RK_{27}.

Guess the value of $RK_1[51, 50]$, *i.e.*, k_5^9 and k_1^9. As the value of $X_1^P[103 \sim 100]$ is known, we can make an Sbox operation and get the value of $X_2^S[103 \sim 100]$. Discard the pairs that don't satisfy $\Delta X_2^S[103, 102, 101] = 100$. This performs a 3-bit filter, about 2^{t+60} pairs remain.

According to the key schedule, when $i \equiv j \ mod\ 2$, RK_i and RK_j consist of the same 64 bits. Thus, RK_{27} contains the same 64 master key bits as RK_1. As shown in Table 10, k_5^9 is also involved in the 27-th round. The value of $X_{26}'^P$ can be deduced by a $\texttt{PermBits}^{-1}$ operation on the ciphertext. By guessing the value of k_0^9, we can make an Sbox^{-1} operation can get the value of $X_{27}[91 \sim 88]$. Discard the pairs that don't satisfy $\Delta X_{27}[91, 89] = 00$. This also performs a 2-bit filter, about 2^{t+58} pairs remain.

Overall, 3 key bits are guessed and the filter probability is 2^{-5}.

Similarly, another 5-bit filter can be got by guessing the value of k_5^8,k_1^8 and k_0^8.

After this, the other 14 active Sboxes are all 4-to-2 transformations. Similar as above, we make a Sbox operation by guessing two key bits in the first round and then make a Sbox^{-1} operation by guessing one key bit in the 27-th round. Each step performs a 4-bit filter. We repeat the process until all involved key bit in RK_{27} and RK_1 are guessed.

After this step, about $2^{t+63-2\times5-14\times4} = 2^{t-3}$ pairs remain.

(b) Guess the value of $RK_2[61, 60]$, *i.e.*, k_7^{14} and k_3^{14} make an Sbox operation to deduce the value of $X_3^S[123 \sim 120]$. Discard the pairs that do not satisfy $\Delta X_3^S[123, 121, 120] = 000$. This performs a 3-bit filter.

As shown in Table 10, k_7^{14} is also involved in RK'_{26}. By guessing the value k_2^2, we make an Sbox^{-1} operation to deduce $\Delta X_{26}[35 \sim 32]$. Discard the pairs that don't satisfy $\Delta X_{26}[35, 32] = 00$. This performs a 2-bit filter.

There are four same bits in RK_2 and RK'_{26}, $k_7^{14}, k_7^{12}, k_3^6, k_3^4$. We repeat a similar process four times. Notice that the pairs go through the 22-th Sbox in the second round with probability 2^{-2}. About $2^{t-3-5*3-4} = 2^{t-22}$ pairs remain.

For the other involved key bits in RK_2 , the pairs will be filtered with probability 2^{-3} after guessing every two key bits. For the other involved key bits in RK'_{26} , the pairs will be filtered with probability 2^{-2} after guessing every two key bits. This procedure is operated 8 times until all involved bits in RK'_2 and RK'_{26} are guessed. The number of candidate pairs will be $2^{t-22-5*4} = 2^{t-42}$.

(c) Guess the value of $RK_3[15]$, notice that $RK_3[14] = k_4^7$ is already guessed in RK'_{27}. Encrypt and discard the pairs that the difference pattern of $\Delta X_4^S[31 \sim 28]$ does not conform to 1000. This performs a 4-bit filter.

Guess $RK_3[11]$, notice that $RK_3[10] = k_4^5$ is already guessed in RK'_{27}. Encrypt and discard the pairs that the difference pattern of $\Delta X_4^S[23 \sim 20]$ does not conform to 0100. This performs a 3-bit filter.

About $2^{t-42-7} = 2^{t-49}$ pairs remain.

(d) As shown in Table 10, $k_4^2, k_4^0, k_0^{10}, k_0^8$ are already guessed in RK'_{27}, k_5^{14}, k_5^{12} are already guessed in RK_1. We only need to guess the value of k_0^2 and k_0^0. Deduce the value of $X_{25}^S[7 \sim 4]$. After a Sbox^{-1} operation, discard the pairs that the difference pattern of $\Delta X_{25}[7 \sim 4]$ do not conform to 0001. We repeat a similar process four times and discard the pairs that can not conform to the output difference of the differential. Each process performs a 4-bit filter, about $2^{t-49-4\times4} = 2^{t-65}$ pairs remain.

Table 11: Time complexity in each step. Note that after guessing the keys involved in an S-box, we have to use 2 S-box operations to compute the partial internal values for P and P', which is about $\frac{2}{32} = \frac{1}{16}$ one round operation.

Step	**#Pairs**	**#Keys**	**Time**($\frac{1}{16}$ Round)	**Probability**
a	2^{t+63}	2^{32+16}	$2^{t+66.7}$	$2^{-5\times2-4\times14}$
b	2^{t-3}	2^{16+12}	$2^{t+1.72}$	$2^{-4-7\times5}$
c	2^{t-42}	2^2	2^{t-40}	2^{-4-3}
d	2^{t-49}	2^2	2^{t-55}	$2^{-4\times4}$

Complexity. The detailed time complexity estimation is shown in Table 11. For the wrong key guesses, $2^{t-49-4\times4} = 2^{t-65}$ pairs remain. While for the right key guesses, there are about $2^{t+127-64-121.83} = 2^{t-58.83}$ pairs remaining. We set $t \approx 60.83$, about 2^2 pairs remain for the right key guesses, and $2^{-4.17}$ pairs remain for each wrong key guess. The data complexity is $2^{60.83+64} = 2^{124.83}$, and the time complexity of the key recovery process is dominated by step (a), which is equal to $\frac{1}{16} \times 2^{60.83+66.7} \approx 2^{123.53}$. The memory complexity for storing the guessed key bits is 2^{80}-bit space.

6 Linear Cryptanalysis of GIFT-128

For GIFT-128, different from differential cryptanalysis, in our two-stage model to search linear hulls, the second stage that applies Matsui's algorithm is very time consuming due to the fast diffusion of the linear mask. In our computing resource (Dell PowerEdge with about 64 cores), we have to restrict the round number of the second stage as 15 rounds. In total, we need about 45 days run the second stage to output a sound 15-round linear hull. However, the time consumption of first stage is negligible. In the InitialSet, the best initial values only involve 56-bit key in the rounds extended at the top and bottom of the linear hull. The second-best solutions involve 76-bit keys. Finally, with the second-best initial values, we find some good 15-round linear hulls as shown in Table 12.

However, based on the 15-round linear hulls, we can not directly launch an attack by extending 4 rounds and 3 rounds at the top and bottom. Note that in our two-stage model, according to the constraints of the first stage, we only restrict that the state of plaintext and ciphertext is not fully activated and then sort the corresponding initial values (input-output of the distinguisher) with the number of involved key bits in the InitialSet. Hence, those distinguishers are only chosen towards a better key-recovery attack, but can not be guarantee to achieve an valid attack. For example, when extending 4-round and 3-round at the top and the bottom of the 15-round distinguishers, the bits of plaintext and ciphertext are not fully activated, but the number of active bits is too large to perform a valid linear attack (the time complexity is higher than exhaustive search). Therefore, our two-stage model is finding distinguishers towards better attacks, but can not guarantee it. When extending 4-round and 3-round fail the attack, we have to peel off several rounds the top or the bottom to enjoy more inactive bits until a valid attack is obtained. Since the initial values in InitialSet do not reach full active bits with 4 rounds (top) and 3 rounds (bottom) extended, they are likely to have fewer active bits when extending fewer rounds (e.g., 3 rounds at top and 2 rounds at bottom) than those not in InitialSet. Hence, the InitialSet is actually a guide to key-recovery-attack-friendly distinguishers.

In addition, due to the 15-round limitation in the second stage, the probability of the linear hulls obtained are usually high. Hence, we extend the 15-round linear hull by one round at the top and bottom to get a 17-round linear hull with probability of 2^{-115}:

$$(0000000000000000000000000a010000) \mapsto (00001000000000000000400000000000).$$

The 17-round linear hull includes 2 trails given in Table 13. Of course, one can search the 17-round linear hull directly in the second stage if the computing resource is large enough.

6.1 The 22-round Linear Hull Attack

Based on the 17-round linear hull, we mount a 22-round linear hull attack on GIFT-128. The whole attack details are shown in Table 14. The attack procedure is as follows.

The attack procedure.

1. Denote the number of the needed plaintext-ciphertext pairs as N. For each of the plaintext-ciphertext pair, do the following steps.

Table 12: 15-round linear hulls of `GIFT-128` with probability 2^{-109}

	Input Mask	Output Mask	#Trails
1	000000a0000000000000000000000000	00000000000000800000000000000000	2
2	000000b0000000000000000000000000	00000000000000800000000000000000	2
3	000000c0000000000000000000000000	00000000000000800000000000000000	2
4	000000d0000000000000000000000000	00000000000000800000000000000000	2
5	0000a000000000000000000000000000	00000000800000000000000000000000	2
6	0000b000000000000000000000000000	00000000800000000000000000000000	2
7	0000c000000000000000000000000000	00000000800000000000000000000000	2
8	0000d000000000000000000000000000	00000000800000000000000000000000	2

Table 13: Two 17-round linear trails with probability 2^{-116}

r	ΔX_r	Pr	ΔX_r	Pr
0	00000000000000000000000a010000	0	00000000000000000000000a010000	0
1	000000a0000000000000000000000000	-4	000000a0000000000000000000000000	-4
2	00000000000000000000000002000000	-6	00000000000000000000000002000000	-6
3	00000020000000100000000000000000	-8	00000020000000100000000000000000	-8
4	01000000080800000000000000000000	-14	01000000080800000000000000000000	-14
5	00000000105000000000000040500000	-22	00000000105000000000000040500000	-22
6	0050005000a000a00000000000000000	-32	0050005000a000a00000000000000000	-32
7	00000000000000004400000000220000	-44	00000000000000004400000000220000	-44
8	00006000000020c0000000000000c010	-54	00006000000030c0000000000000c010	-54
9	00000000200000a01000000000a0000	-66	00000000200000a01000000000a0000	-66
10	00000000000000000082802000000000	-76	00000000000000000082802000000000	-76
11	00009500000000000000650000000000	-86	00009500000000000000650000000000	-86
12	00000000000000000000000000c000c00	-100	00000000000000000000000000c000c00	-100
13	00000000000001100000000000000000	-104	00000000000001100000000000000000	-10
14	00000000000c0000000000000000000	-110	00000000000c0000000000000000000	-106
15	00000000000000000000000000100000	-112	00000000000000000000000000100000	-112
16	00000000000008000000000000000000	-114	00000000000008000000000000000000	-114
17	00001000000000000000400000000000	-116	00001000000000000000400000000000	-116

2. The corresponding value of X_1^P and $X_{22}'^K$ can be deduced from each plaintext-ciphertext pair. Compress the N samples into 2^{78} counters according to the value of $(X_P^1[63 \sim 48], X_P^1[31 \sim 0], X_{22}'^K[127 \sim 124, 111 \sim 108, 95 \sim 92, 79, 76, 63 \sim 60, 47 \sim 44, 31 \sim 28, 15 \sim 12])$. Since when decrypting the messages, $X_{21}'^P[78]$ is linear with $X_{22}'^S[77]$ and $X_{22}'^S[78]$, there is no need to store them. If $X_{22}'^S[78] \oplus X_{22}'^S[77] = 0$, increase the corresponding counter by one and otherwise, decrease it by one. There are 2^{78} counters remaining.

3. Guess the value of $RK_1[0, 1]$ and make one Sbox operation. We can get the value of $X_2^S[2]$. Similar to the above step, we can compress the states into the counters of $(X_1^P[63 \sim 48], X_1^P[31 \sim 4], X_{22}'^K[127 \sim 124, 111 \sim 108, 95 \sim 92, 79, 76, 63 \sim 60, 47 \sim 44, 31 \sim 28, 15 \sim 12])$. This requires $2^{78+2} = 2^{80}$ Sbox operations and 2^{74} counters remain.

4. Repeat guessing every two bits of RK_1 and making the single Sbox operation. After guessing all 24 involved key bits in RK_1, according to the value of $(X_2^P[111 \sim 108, 102 \sim 100, 65 \sim 64], X_{22}'^K[127 \sim 124, 111 \sim 108, 95 \sim 92, 79, 76, 63 \sim 60, 47 \sim 44, 31 \sim 28, 15 \sim 12])$, the messages can be compressed into 2^{39} counters. The total complexity of guessing RK^1 is $2^{80} + 2^{78} + 2^{76} + 2^{75} + 2^{74} + 2^{73} + \cdots + 2^{67} \approx 2^{80}$ Sbox operations.

5. Guess the value of $RK_{22}'[6, 7]$ and make the inverse operation of a single Sbox, we can get the value of $X_{21}'^P[12]$. The messages can be compressed into 2^{36} counters according to the value of $X_2^P[111 \sim 108, 102 \sim 100, 65, 64], X_{22}'^K[127 \sim 124, 111 \sim 108, 95 \sim 92, 79, 76, 63 \sim 60, 47 \sim 44, 31 \sim 28], X_{21}'^P[12]$. This requires $2^{24} \times 2^{39+2} = 2^{65}$ Sbox operations.

Table 14: The linear hull attack on 22-round GIFT-128

P	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
X_1^S	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	
	--	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	
X_1^P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	****	****	****	****	----	----	----	----	****	****	****	****	****	****	****	****	****
RK_1	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
	11	11	11	11	--	--	--	--	11	11	11	11	11	11	11	11	
X_2^S	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	-*--	--*-	---*	*---	----	----	----	----	-*--	--*-	---*	1---	--*-	---*	1---	-1--	
X_2^P	----	----	----	----	****	----	1***	----	----	----	----	----	----	----	----	----	11**
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
RK_2	--	--	--	--	11	--	11	--	--	--	--	--	--	--	--	11	
	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
X_3^S	----	----	----	----	1---	----	--1-	----	----	----	----	----	----	----	----	----	---1
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
X_3^P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	----	1-1-	----	---1	----	----	----	----	----
RK_3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
	--	--	--	--	--	--	--	--	--	-1	--	--	--	--	--	--	--
$Input$	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	----	----	----	----	----	----	1-1-	----	---1	----	----	----	----	----
$Output$	----	----	----	---1	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	-1--	----	----	----	----	----	----	----	----	----	----	----	----	----
X_{21}^S	----	----	----	****	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	*11*	----	----	----	----	----	----	----	----	----	----	----	----	----
RK'_{21}	--	--	--	11	--	--	--	--	--	--	--	--	--	--	--	--	--
	--	--	--	11	--	--	--	--	--	--	--	--	--	--	--	--	--
$X_{21}^{'K}$	----	----	----	****	----	----	----	----	----	----	----	----	----	----	----	----	----
	----	----	----	*11*	----	----	----	----	----	----	----	----	----	----	----	----	----
$X_{21}^{'P}$	*---	----	----	----	*---	----	----	----	-*--	----	----	----	-1--	----	----	----	----
	--*-	----	----	----	--1-	----	----	----	---*	----	----	----	---*	----	----	----	----
X_{22}^S	****	----	----	----	****	----	----	----	****	----	----	----	*11*	----	----	----	----
	****	----	----	----	****	----	----	----	****	----	----	----	****	----	----	----	----
RK'_{22}	11	--	--	--	11	--	--	--	11	--	--	--	11	--	--	--	
	11	--	--	--	11	--	--	--	11	--	--	--	11	--	--	--	
$X_{22}^{'K}$	****	----	----	----	****	----	----	----	****	----	----	----	*11*	----	----	----	----
	****	----	----	----	****	----	----	----	****	----	----	----	****	----	----	----	----
C	-*--	-*--	-*--	-1--	-*--	-*--	-*--	-*--	--*-	--*-	--*-	--1-	--*-	--*-	--*-	--*-	
	---*	---*	---*	---*	---*	---*	---*	---*	*---	*---	*---	*---	*---	*---	*---	*---	

Table 15: Key bits involved of 22-round linear cryptanalysis of GIFT-128

1	$k_4^{15}k_0^{15}$ $k_4^{14}k_0^{14}$ $k_4^{13}k_0^{13}$ $k_4^{12}k_0^{12}$ $k_4^7k_0^7$ $k_4^6k_0^6$ $k_4^5k_0^5$ $k_4^4k_0^4$ $k_4^3k_0^3$ $k_4^2k_0^2$ $k_4^1k_0^1$ $k_4^0k_0^0$
2	$k_7^{11}k_3^{11}$ $k_7^9k_3^9$ $k_7^0k_3^0$
3	k_4^6
21	$k_5^1k_0^{11}$ $k_5^{13}k_0^7$
22	$k_7^9k_3^1$ $k_7^5k_3^{13}$ $k_7^8k_3^0$ $k_7^4k_3^{12}$ $k_7^7k_3^{15}$ $k_7^3k_3^{11}$ $k_7^6k_3^{14}$ $k_7^2k_3^{10}$

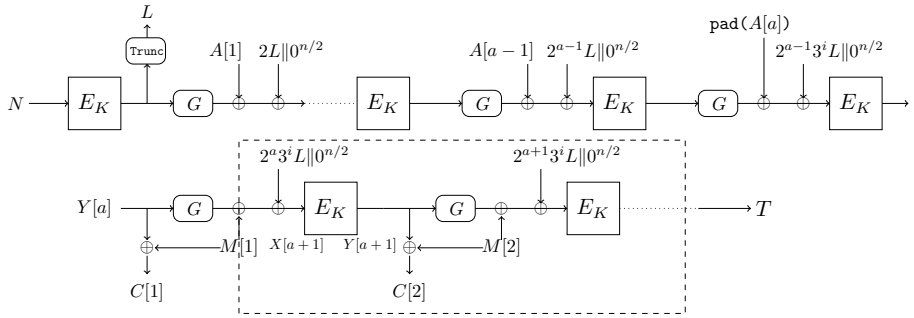

Figure 1: Attack target of GIFT-COFB

6. Repeat guessing the other 12 bits in RK'_{22} (the key bits with symbol *) and compressed the messages into 2^{15} counters according to the value of $X_2^P[111 \sim 108, 102 \sim 100, 65, 64], X_{21}'^K[115 \sim 112, 51, 48]$. The total complexity of guessing RK'_{22} is about 2^{41} Sbox operations.

7. Similarly, we can guess two key bits and compress the states in each step until covering all key bits involved in the attack. This leads to a significant less time compared to the time in step 4. We omit the detailed procedure in this paper.

8. We store the number of messages that satisfy $\{Input[27] \oplus Input[25] \oplus Input[16] = Output[112] \oplus Output[50]\}$ in the remaining counter as λ and choose the corresponding key as the right key that $|\lambda - N/2|$ is the largest.

Complexity. To deduce the right key value, we collect $N = c \cdot \frac{1}{\varepsilon^2} = 4 \cdot \frac{1}{2^{-115}} = 2^{117}$ plaintext-ciphertext pairs. The time complexity of the attack procedure for guessing keys is dominated by Step(4) and it is about 2^{80} Sbox operations, thus the time complexity is 2^{117} for encrypting the messages. The memory complexity is about 2^{78}-bit for implementing the counters.

7 Linear Cryptanalysis on GIFT-COFB

The attacking target is the part in dashed box in Figure 1. The security claim of GIFT-COFB is up to birthday bound, thus, the valid data complexity of cryptanalysis on GIFT-COFB should be less than $2^{n/2} = 2^{64}$. In addition, since the input of E_K has the form as shown in Figure 1, taking the first block as example, the input of the E_K is $G(Y[a]) \oplus M[1] \oplus (2^a3^iL\|0^{n/2})$, where L is unknown (depending on Nonce and key). Hence, there should be no active mask bits in the most significant 64 bits of the input. The largest number of extended rounds at the top/bottom is different from GIFT-128, i.e., 3 rounds both on the top and the bottom for GIFT-COFB. Based on a 9-round linear hull:

$$0000000000005050000000000005050 \rightarrow 00000000000000000000040000000200$$

Table 16: The 9-round linear trail

r	X_r	Pr
0	00000000000050500000000000005050	0
1	00000000000a000a00000000000a000a	-8
2	00000000000000000022002200000000	-16
3	00009900000000000000660000000000	-28
4	0000000000000000000000000c000c00	-40
5	00000000000000110000000000000000	-44
6	00000000000c0000000000000000000	-50
7	00000000000000000000000000100000	-52
8	00000000000000000000004000000000	-56
9	00000000000000000000040000000200	-58

with probability 2^{-58}, we analyze the security of its 15-round GIFT-128-based version. The linear trail is shown in Table 16.

Suppose we get a set of plaintext-ciphertext pairs of GIFT-COFB, *i.e.,* $(M[1], C[1])$, $(M[2], C[2]), \ldots, (M[m-1], C[m-1])$ under one nonce N. Then we can get the input-output pairs of the cipher GIFT-128, where the input is $G(M[i] \oplus C[i]) \oplus M[i]$, the corresponding output is $M[i+1] \oplus C[i+1]$. So just according to each $(G(M[i] \oplus C[i]) \oplus M[i])[63-0]$ and $M[i+1] \oplus C[i+1]$, we can get one useful plaintext-ciphertext pair. In fact, these samples may also be collected from the plaintext-ciphertext pairs under different nonces. The attack process is shown in Table 18 and the involved key bits are listed in Table 19 in Appendix C

The attack procedure.

1. Denote the number of the needed plaintext-ciphertext pairs as N. For each of the plaintext-ciphertext pair, do the following steps.

2. The corresponding value of X_1^P and $X_{15}'^K$ can be deduced from each plaintext-ciphertext pair. Compress the N samples into 2^{96} counters according to the value of $(X_P^1[111 \sim 96], X_P^1[79 \sim 64], X_{15}'^K[123 \sim 120, 107 \sim 104, 91 \sim 88, 83 \sim 80, 75 \sim 72, 67 \sim 64, 59 \sim 56, 51 \sim 48, 43 \sim 40, 35 \sim 32, 27 \sim 24, 19 \sim 16, 11 \sim 8, 3 \sim 0])$.

3. Guess the value of $RK_1[33, 32]$, *i.e.,* $k_5^0 k_1^0$. Make one Sbox operation to calculate the value of $X_2^S[65, 64]$.

   k_5^0 is also involved in the 15-th round as $RK_{15}'[41]$. By guessing the value of $RK_{15}'[42]$, we can make one Sbox^{-1} operation and get the value of $X_{15}[43, 41]$.

   Similar to the above step, we can compress the states into the counters of $(X_P^1[111 \sim 96], X_P^1[79 \sim 68], X_2^S[65, 64], X_{15}'^K[123 \sim 120, 107 \sim 104, 91 \sim 88, 83 \sim 80, 75 \sim 72, 67 \sim 64, 59 \sim 56, 51 \sim 48, 35 \sim 32, 27 \sim 24, 19 \sim 16, 11 \sim 8, 3 \sim 0], X_{15}[43, 41])$.

   This requires about $2^{96} \cdot 2^2 + 2^{94} \cdot 2^3 = 2^{98} + 2^{97}$ Sboxes and 2^{92} counters remain.

4. There are 8 same key bits in RK_1 and RK_{15}', *i.e.,* $k_5^{3\sim 0}, k_1^{11\sim 8}$. Each key bit corresponds to one active Sbox in the first round. Repeat a similar process as in Step 3 for 8 times to guess all involved key bits in RK_1 and 16 key bits in RK_{15}'.

   After that, there are still 16 unknown key bits in RK_{15}', repeat guessing each two key bits which corresponds to the same Sbox and make one Sbox^{-1} operation until all key bits in RK_{15}' are known.

   Till now, we get the value of all involved key bit in RK_1 and RK_{15}'. The states can be compressed into 2^{46} counters, which corresponds to the value of $(X_2^S[111, 108, 107, 106, 102, 101, 97, 96, 79, 76, 75, 74, 70, 69, 65, 64], X_{14}'^K[43 \sim 40, 35 \sim 32, 11, 8, 3 \sim 0])$. Since when decrypting the messages, $X_{14}'^K[10, 9]$ is linear with $X_{14}[10]$, there is no need to

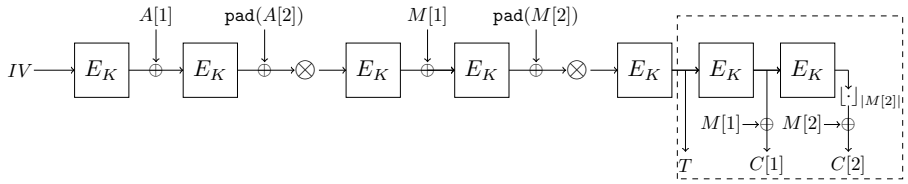

Figure 2: Attack target of SUNDAE-GIFT

store them. If $X_{14}'^{K}[10] \oplus X_{14}'^{K}[9] = 0$, increase the corresponding counter by one and otherwise, decrease it by one.

The total complexity of guessing RK_1 and RK_{15}' is $2^{98} + 2 \cdot 2^{97} + \cdots + 2 \cdot 2^{90} + 7 \cdot 2^{89} \approx 2^{99.6}$ Sbox operations.

5. Since the time complexity is dominated by guessing RK_1, RK_{15}' and the process for guessing the involved key bits in $RK_2, RK_3, RK_{13}', RK_{14}'$ and compressing the states is similar to the process as describe in above steps, we don't repeat the details.

6. We store the number of messages that satisfy $\{Input[78] \oplus Input[76] \oplus Input[70] \oplus Input[68] \oplus Input[14] \oplus Input[12] \oplus Input[6] \oplus Input[4] = Output[9] \oplus Output[42]\}$ in the remaining counter as λ and choose the corresponding key as the right key that $|\lambda - N/2|$ is the largest.

Complexity. The attack need a $\frac{c}{\epsilon^2} = 2^{62}$ data when $c = 4$, the time complexity is about $2^{99.6}$ Sbox operations, which is equivalent to $2^{99.6} \cdot \frac{1}{32} \cdot \frac{1}{15} \approx 2^{90.7}$ 15-round encryptions, the memory complexity is about 2^{96}-bit space to implement the counters.

8 Linear Cryptanalysis on SUNDAE-GIFT

In our cryptanalysis on SUNDAE-GIFT, we focus on the Encryption message step shown in dashed box in Figure 2. Due to the security claim up to birthday bound, the data complexity of cryptanalysis on SUNDAE-GIFT should be also less than $2^{n/2} = 2^{64}$. Hence, the analysis result in Section 6 is invalid. In the data collection phase, we collect T, $(M[1], C[1])$,...,$(M[l], C[l])$. The input and output pairs of GIFT-128 are $(T, M[1] \oplus C[1])$,..., $(M[i] \oplus C[i], M[i+1] \oplus C[i+1])$.

Key recovery attack. Based on the a 10-round linear trail

$$0000000000000000000000a008a002 \rightarrow 00000000000000000044000000220000$$

with probability 2^{-56}, we analyze the security of the 16-round GIFT-128-based version of SUNDAE-GIFT. Due to the data usage limit, we can only add 4 rounds at the top and 2 rounds at the bottom. The situation is similar to the 22-round linear attack on GIFT-128. The trail is shown in Table 20 in Appendix C.

During the key-recovery process, the linear mask of X_1^P has 64 active bits and the linear mask of the ciphertext has 32 active bits. The memory complexity is dominated by implementing the $2^{64+32} = 2^{96}$ counters with about 2^{96}-bit space. Similarly with the process in Section 6, by guessing key bits involved in each Sbox operation and compressing messages into new counters, we choose the corresponding key as the right key with the largest bias.

Complexity. The data complexity is $\frac{c}{\epsilon^2} = 2^{60}$ when c is set as 4. The time complexity is about $2^{100.2}$ Sbox operations, which is equivalent to $2^{91.2}$ 16-round encryptions. The memory complexity is 2^{96}-bit space.

Comment. Due to the data usage limitation of GIFT-COFB and SUNDAE-GIFT, we can not mount long differential attack on these two proposals. In addition, for GIFT-COFB, there are varied secret $2^a \cdot 3^i \cdot L$ for different blocks. While for linear attack, we can ignore this value with zero linear mask for this part.

9 Conclusion

In this article, we propose a two-step strategy for searching advantageous distinguishers which can lead to long attacks with small involved key bits. The first step is to reduce the searching space and give advantageous initial values of the second step. It utilizes the advantages of the MILP based method and Matsui's branch-and-bound algorithm and can be used to search for differential trails and linear trails. As a first application, the strategy is used to analyze GIFT-128 and a 27-round differential attack and a 22-round linear hull attack on GIFT-128. The differential attack covers one more round than the previous result. What's more, we give evaluation results on two GIFT-128-based proposals: SUNDAE-GIFT and GIFT-COFB.

Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704701), the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008), the National Natural Science Foundation of China (No. 61902207), and the Chinese Major Program of National Cryptography Development Foundation (No. MMJJ20180101). Yiyuan Luo is supported by National Natural Science Foundation of China (No. 62072207), Natural Science Foundation of Shanghai (No. 19ZR1420000), and Open Foundation of Network and Data Security Key Laboratory of Sichuan Province (University of Electronic Science and Technology of China).

References

[AK18] Ralph Ankele and Stefan Kölbl. Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 163–190. Springer, 2018. [page 157]

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017. [page 157]

[BBLT18] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. SUNDAE: small universal deterministic authenticated encryption for the internet of things. *IACR Trans. Symmetric Cryptol.*, 2018(3):1–35, 2018. [page 161]

[BBP+] Subhadeep Banik, Andrey Bogdanov, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, Elmar Tischhauser, and Yosuke Todo. Sundae-gift. Submission to Round 1 of the NIST Lightweight Cryptography Standardization process (2019). [pages 157, 161, 164, and 182]

[BCI+] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift-cofb. Submission to Round 1 of the NIST Lightweight Cryptography Standardization process (2019). [pages 157, 161, 164, 180, and 181]

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016. [page 157]

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 450–466, 2007. [page 157]

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345, 2017. [pages 157, 159, 160, and 161]

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO 90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1991. [page 156]

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 175:1–175:6, 2015. [page 157]

[CDJ+19] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. Elastic-tweak: A framework for short tweak tweakable block cipher. Cryptology ePrint Archive, Report 2019/440, 2019. https://eprint.iacr.org/2019/440. [page 157]

[CIMN17] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 277–298, 2017. [page 161]

[CWZ] Lele Chen, Gaoli Wang, and Guoyan Zhang. MILP-based related-key rectangle attack and its application to GIFT, Khudra, MIBS. Accepted by The Computer Journal. [page 158]

[FWG+16] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. 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*, pages 268–288, 2016. [page 157]

[Gur] Gurobi - the fastest solver. http://www.gurobi.com/. [page 157]

[HV18] Mathias Hall-Andersen and Philip S. Vejre. Generating graphs packed with paths estimation of linear approximations and differentials. *IACR Trans. Symmetric Cryptol.*, 2018(3):265–289, 2018. [page 158]

[IKM+] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Ling Sun. Thank goodness it's friday (TGIF). Submission to Round 1 of the NIST Lightweight Cryptography Standardization process (2019). [page 157]

[JZD19] Fulei Ji, Wentao Zhang, and Tianyou Ding. Improving matsui's search algorithm for the best differential/linear trails and its applications for des, DESL and GIFT. *IACR Cryptology ePrint Archive*, 2019:1190, 2019. [pages 157, 159, and 164]

[JZZD20] Fulei Ji, Wentao Zhang, Chunning Zhou, and Tianyou Ding. Improved (related-key) differential cryptanalysis on gift. To appear at SAC 2020, 2020. https://eprint.iacr.org/2020/1242. [pages 157, 158, 159, and 164]

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015. [page 157]

[LLL+19] Yu Liu, Huicong Liang, Muzhou Li, Luning Huang, Kai Hu, Chenhe Yang, and Meiqin Wang. Stp models of optimal differential and linear trail for s-box based ciphers. Cryptology ePrint Archive, Report 2019/025, 2019. https://eprint.iacr.org/2019/025. [pages 158 and 159]

[LS19] Yunwen Liu and Yu Sasaki. Related-key boomerang attacks on gift with automated trail search including bct effect. Cryptology ePrint Archive, Report 2019/669, 2019. https://eprint.iacr.org/2019/669. [pages 158 and 159]

[LWZZ19] Lingchen Li, Wenling Wu, Yafei Zheng, and Lei Zhang. The relationship between the construction and solution of the MILP models and applications. *IACR Cryptology ePrint Archive*, 2019:49, 2019. [pages 157, 158, and 159]

[Mat94] Mitsuru Matsui. Linear Cryptoanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1994. [page 156]

[NIS] NIST. Lightweight cryptography (lwc) standardization process, 2019. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates. [page 157]

[Sas18] Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai,*

Japan, September 3-5, 2018, Proceedings, pages 227–243, 2018. [pages 157 and 158]

[SS14] Meiqin Wang Peng Wang Kexin Qiao Xiaoshuang Ma Danping Shi Ling Song Kai Fu Siwei Sun, Lei Hu. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747, 2014. http://eprint.iacr.org/2014/747. [page 157]

[STP] Stp : The simple theorem prover. http://www.http://stp.github.io/. [page 157]

[ZDM+19] Boxin Zhao, Xiaoyang Dong, Willi Meier, Keting Jia, and Gaoli Wang. Generalized related-key rectangle attacks on block ciphers with linear key schedule. Cryptology ePrint Archive, Report 2019/714, 2019. https://eprint.iacr.org/2019/714. [page 158]

[ZDY18] Baoyu Zhu, Xiaoyang Dong, and Hongbo Yu. Milp-based differential attack on round-reduced gift. Cryptology ePrint Archive, Report 2018/390, 2018. https://eprint.iacr.org/2018/390. [pages 157, 159, and 164]

[ZSCH18] Yingjie Zhang, Siwei Sun, Jiahao Cai, and Lei Hu. Speeding up MILP aided differential characteristic search with matsui's strategy. In Liqun Chen, Mark Manulis, and Steve A. Schneider, editors, *Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings*, volume 11060 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 2018. [page 159]

[ZZDX19] Chunning Zhou, Wentao Zhang, Tianyou Ding, and Zejun Xiang. Improving the milp-based security evaluation algorithm against differential/linear cryptanalysis using A divide-and-conquer approach. *IACR Trans. Symmetric Cryptol.*, 2019(4):438–469, 2019. [page 157]

A GIFT-COFB

GIFT-COFB instantiates the COFB (Combined FeedBack) block cipher based AEAD mode with the GIFT-128 block cipher. This proposal primarily focuses on the hardware implementation size.

Recommended Parameter Choice. In GIFT-COFB, the underlying block cipher is the only parameter. The block cipher can be chosen by the following recommendation.

1. n: Length of the block cipher state in bits. The recommended choice is $n = 128$.
2. τ : Length of the tag in bits. The recommended choice is $\tau = 128$.
3. EK : The recommended choice of EK is the block cipher GIFT-128.

Input and Output Data. To encrypt a message M with associated data A and nonce N, one needs to provide the information given below.

The encryption algorithm takes as input:

- An encryption key $K \in \{0,1\}^{128}$.
- A nonce $N \in \{0,1\}^{128}$. This can include the counter to make the nonce non-repeating.
- Associated data and message $A, M \in \{0,1\}^*$.

It generates the following output data:

- Ciphertext $C \in \{0,1\}^{|M|}$.
- Tag $T \in \{0,1\}^{128}$.

To decrypt (with verification) a ciphertext-tag pair (C, T) with associated data A and nonce N, one needs to provide the information given below.

- An encryption key $K \in \{0,1\}^{128}$.
- A nonce $N \in \{0,1\}^{128}$.
- Associated data and ciphertext $A, C \in \{0,1\}^*$.
- Tag $T\{0,1\}^{128}$.

It generates the following output data:

- Message $M\{0,1\}^{|C|} \bigcup \{\perp\}$. $\perp$ is a special symbol denoting rejection.

The specifications of GIFT-COFB are shown in Algorithm 2 and Figure 3. For more details, please refer to [BCI+].

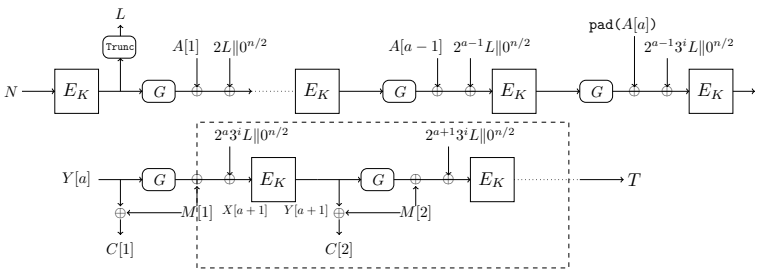

Figure 3: GIFT-COFB encryption [BCI+].

Algorithm 2: The encryption of COFB [BCI+]

Input: N, A, M
Output: C, T
$Y[0] \leftarrow E_K(N)$
$L \leftarrow \texttt{Trunc}_{n/2}(Y[0])$
$(A[1], ..., A[a]) \leftarrow \texttt{Pad}(A)$
if $M \neq \epsilon$ **then**
    $(M[1], ..., M[a]) \leftarrow \texttt{Pad}(M)$
for i from 1 to $a-1$ **do**
    $L \leftarrow 2 \cdot L$
    $X[i] \leftarrow A[i] \oplus G \cdot Y[i-1] \oplus L \| 0^{n/2}$
    $Y[i] \leftarrow E_K(X[i])$
if $|A| \text{mod} n = 0$ and $A \neq \epsilon$ **then**
    $L \leftarrow 3 \cdot L$
else
    $L \leftarrow 3^2 \cdot L$
if $M \neq \epsilon$ **then**
    $L \leftarrow 3^2 \cdot L$,
$X[a] \leftarrow A[a] \oplus G \cdot Y[a-1] \oplus L \| 0^{n/2}$
$Y[a] \leftarrow E_K(X[a])$
for i from 1 to $m-1$ **do**
    $L \leftarrow 2 \cdot L$
    $C[i] \leftarrow M[i] \oplus Y[i+a-1]$
    $X[i+a] \leftarrow M[i] \oplus G \cdot Y[i+a-1] \oplus L \| 0^{n/2}$
    $Y[i+a] \leftarrow E_K(X[i+a])$
if $M \neq \epsilon$ **then**
    if $|M| \text{mod} n = 0$ **then**
        $L \leftarrow 3 \cdot L$
    else
        $L \leftarrow 3^2 \cdot L$
    $C[m] \leftarrow M[m] \oplus Y[a+m-1]$
    $X[a+m] \leftarrow M[m] \oplus G \cdot Y[a+m-1] \oplus L \| 0^{n/2}$
    $Y[a+m] \leftarrow E_K(X[a+m])$
    $C \leftarrow \texttt{Trunc}_{|M|}(C[1] \| ... \| C[m])$
    $T \leftarrow \texttt{Trunc}_{\tau}(Y[a+m])$
else
    $C \leftarrow \epsilon$
    $T \leftarrow \texttt{Trunc}_{\tau}(Y[a])$
return (C, T)

B SUNDAE-GIFT

The encryption of SUNDAE takes a 128-bit key K, an associated data $A \in \{0,1\}^*$ and a message $M \in \{0,1\}^*$ as input. The designers define four variants of SUNDAE-GIFT depending on 4 different length of nonce N as shown in Table 17. For simplicity, the nonce N and associated data A are regarded as $A \leftarrow N \| A$. SUNDAE outputs a ciphertext $C \in \{0,1\}^{n+|M|}$, where the first n bits are the *tag* T. The encryption algorithm shown in Figure 4 is composed of five steps:

1. **Initialization.** An initial block $B = b_{127}b_{126}b_{125}b_{124} \| 0$ is defined as follows:

$$
\begin{aligned}
b_{127} &= \begin{cases} 0 & \text{if } |A| = 0 \\ 1 & \text{otherwise.} \end{cases} \\
b_{126} &= \begin{cases} 0 & \text{if } |M| = 0 \\ 1 & \text{otherwise.} \end{cases} \\
b_{125}b_{124} &= \begin{cases} 00 & \text{if } |N| = 0 \\ 01 & \text{if } |N| = 64 \\ 10 & \text{if } |N| = 96 \\ 11 & \text{if } |N| = 128. \end{cases}
\end{aligned}
$$

Apply GIFT-128 block cipher E_K to the initial block B to produce a chaining value V.

2. **Processing associated data.** If the associated data A is empty, skip this step. Otherwise, partition A into 128-bit blocks (the last block may be a partial block). The blocks are processed as shown in Figure 4 and the last padded block is processed differently by multiplying m before applying GIFT-128 block cipher.
3. **Processing message.** The message M is processed in a similar way to the step of **Processing associated data**.
4. **Extracting tag.** As shown in Figure 4, the chaining value is outputed as the tag T.
5. **Encrypting message.** The message blocks are encrypted as shown in Figure 4.

For more detials about SUNDAE-GIFT, please refer to [BBP+].

Table 17: Four members in the SUNDAE-GIFT family [BBP+]

Member	Name	Nonce length	Key length	Tag length
1	SUNDAE-GIFT-96	96	128	128
2	SUNDAE-GIFT-0	0	128	128
3	SUNDAE-GIFT-128	128	128	128
4	SUNDAE-GIFT-64	64	128	128

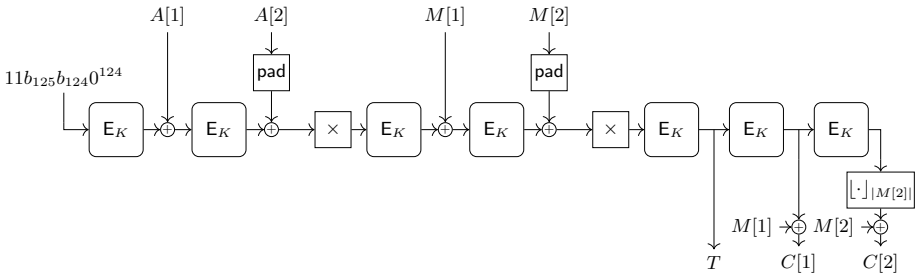

Figure 4: SUNDAE-GIFT encryption with association and plaintext data [BBP+].

C Tables of the attack on GIFT-COFB and the 10-round linear trail of SUNDAE-GIFT

Table 18: The linear hull attack on 15-round-GIFT-128-based GIFT-COFB

P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****	****
X_1^S	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	-**-	--**	*--*	**--	
X_1^P	----	----	----	----	****	****	****	****	----	----	----	----	****	****	****	****	
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
RK_1	--	--	--	--	11	11	11	11	--	--	--	--	11	11	11	11	
	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
X_2^S	----	----	----	----	*--*	**--	-**-	--**	----	----	----	----	*--*	**--	-**-	--**	
	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
X_2^P	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
	----	****	----	****	----	----	----	----	----	****	----	****	----	----	----	----	
RK_2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
	--	11	--	11	--	--	--	--	--	11	--	11	--	--	--	--	
X_3^S	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
	----	-1-1	----	-1-1	----	----	----	----	----	-1-1	----	-1-1	----	----	----	----	
X_3^P	----	----	----	----	----	----	----	----	----	----	----	----	----	-1-1	----	-1-1	
	----	----	----	----	----	----	----	----	----	----	----	----	----	-1-1	----	-1-1	
RK_3	--	--	--	--	--	--	--	--	--	--	--	--	1-	--	1-	--	
	--	--	--	--	--	--	--	--	--	--	--	--	1-	--	1-	--	
Input	----	----	----	----	----	----	----	----	----	----	----	----	-1-1	----	-1-1	----	
	----	----	----	----	----	----	----	----	----	----	----	----	-1-1	----	-1-1	----	
Output	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
	----	----	----	----	----	-1--	----	----	----	----	----	----	----	--1-	----	----	
X_{13}^S	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
	----	----	----	----	----	*11*	----	----	----	----	----	----	----	****	----	----	
RK'_{13}	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
	--	--	--	--	--	11	--	--	--	--	--	--	--	11	--	--	
X'^K_{13}	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	
	----	----	----	----	----	*11*	----	----	----	----	----	----	----	****	----	----	
X_{14}	----	----	----	----	----	--1-	----	--*-	----	----	----	----	----	---*	----	---*	
	----	----	----	----	----	*---	----	*---	----	----	----	----	----	-1--	----	-*--	
X_{14}^S	----	----	----	----	----	****	----	****	----	----	----	----	----	****	----	****	
	----	----	----	----	----	****	----	****	----	----	----	----	----	*11*	----	****	
RK'_{14}	--	--	--	--	--	11	--	11	--	--	--	--	--	11	--	11	
	--	--	--	--	--	11	--	11	--	--	--	--	--	11	--	11	
X'^K_{14}	----	----	----	----	----	****	----	****	----	----	----	----	----	****	----	****	
	----	----	----	----	----	****	----	****	----	----	----	----	----	*11*	----	****	
X_{15}	----	*-*-	----	*-*-	----	*-*-	----	*-1-	----	-*-*	----	-*-*	----	-*-*	----	-*-*	
	----	*-*-	----	*-*-	----	*-*-	----	*-*-	----	-*-*	----	-*-*	----	-*-*	----	-1-*	
X_{15}^S	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****	
	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****	
RK'_{15}	--	11	--	11	--	11	--	11	--	11	--	11	--	11	--	11	
	--	11	--	11	--	11	--	11	--	11	--	11	--	11	--	11	
X'^K_{15}	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****	
	----	****	----	****	----	****	----	****	----	****	----	****	----	****	----	****	
C	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	
	--	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	*-*-	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	-*-*	

Table 19: Involved key bits of linear cryptanalysis on GIFT-COFB

1					$k_5^{11}k_1^{11}$	$k_5^{10}k_1^{10}$	$k_5^{9}k_1^{9}$	$k_5^{8}k_1^{8}$				$k_5^{3}k_1^{3}$	$k_5^{2}k_1^{2}$	$k_5^{1}k_1^{1}$	$k_5^{0}k_1^{0}$	
2		$k_6^{14}k_2^{14}$		$k_6^{12}k_2^{12}$						$k_6^{6}k_2^{6}$	$k_6^{4}k_2^{4}$					
3												k_{15}^{5} k_0^{5}		k_{13}^{3} k_0^{3}		
13					$k_4^{6}k_1^{0}$								$k_4^{4}k_1^{14}$			
14					$k_6^{10}k_3^{4}$ $k_6^{6}k_3^{0}$			$k_7^{12}k_2^{2}$ $k_7^{8}k_2^{14}$					$k_6^{8}k_3^{2}$ $k_6^{4}k_3^{14}$		$k_7^{10}k_2^{0}$ $k_7^{6}k_2^{12}$	
15	$k_0^{7}k_0^{5}$ $k_0^{3}k_0^{1}$		$k_1^{15}k_1^{3}$ $k_1^{11}k_1^{7}$		$k_0^{6}k_0^{4}$ $k_0^{2}k_0^{0}$			$k_1^{14}k_1^{2}$ $k_1^{10}k_1^{6}$		$k_0^{5}k_0^{3}$ $k_0^{1}k_0^{15}$	$k_1^{13}k_1^{1}$ $k_1^{9}k_1^{5}$		$k_0^{4}k_0^{2}$ $k_0^{0}k_0^{14}$		$k_1^{12}k_1^{0}$ $k_1^{8}k_1^{4}$	

Table 20: The 10-round linear trail

r	X_r	Pr
0	0000000000000000000000000a008a002	0
1	00000000000000660000000000000011	-8
2	00000000000c000c0000000000000000	-20
3	00000000000000000000000000110000	-24
4	00000000000000c0000000000000000	-30
5	00000000000000000000000000020000	-32
6	00000000000000000000002000000010	-34
7	00000000000008080000000000000000	-40
8	00000000005000000000000050000	-44
9	00000000040004000000000000000000	-52
10	00000000000000000044000000220000	-56