GIFT

SPL Encrypted

[1] V Shyam, 12041700 vellurushyam@iitbhilai.ac.in
[2] Prathamesh R, 12041170 rprathamesh@iitbhilai.ac.in
[3] Lohit Daksha, 12040800 lohitdaksha@iitbhilai.ac.in

Abstract. In this document, we have given an introduction on the working of the GIFT cipher. Section 1 explains how GIFT cipher improves upon PRESENT cipher. In Section 2, we explain the working of GIFT cipher in detail. Section 3 analyses the Sbox of GIFT cipher and shows the Difference Distribution Table and the Linear Approximation Table of the Sbox. In Section 4, we have presented how GIFT cipher can be cryptanalysed by Differential and Linear Cryptanalysis. Section 5 presents a way to model the Sbox and the permutation layer of GIFT using Mixed Integer Linear Programming.

Keywords: GIFT · PRESENT · SKINNY · Mixed Integer Linear Programming · Differential Cryptanalysis · Linear Cryptanalysis · Differential Branch Number · Differential Uniformity

1 Introduction

GIFT is a relatively new lightweight block cipher that improves upon one of the most famous lightweight block ciphers invented, *PRESENT*. The *PRESENT* block cipher was first presented 15 years ago at the Cryptographic Hardware and Embedded Systems - CHES 2007 - workshop. Now, *PRESENT* cipher is an ISO standard.

One of the reasons for *PRESENT* cipher's popularity is that it is compact, around 2.5 times smaller than *AES*. 10 years after CHES 2007, the *GIFT* cipher was presented at CHES 2017, with the claim of having some advantages over *PRESENT*. Some of them are listed below.

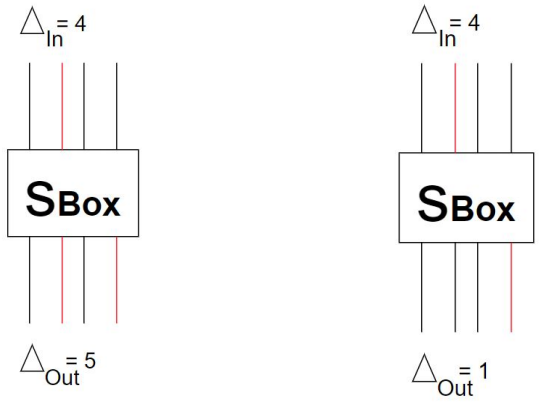
1. **Smaller area** owing to a smaller Sbox and lesser subkey additions
2. **Better resistance against Linear Cryptanalysis** thanks to the choice of the Sbox and the bit permutations.
3. Lesser rounds and **higher throughput**

4. Simpler and **faster key schedule**

All the above points are easily understood, except that it has a **smaller area**. In order to understand this, we will need to know a few concepts about the Sbox itself.

1.1 Differential Branching Number x (BNx)

The differential branching number of an Sbox is the least total hamming weight of the input and output diffferences. Here, hamming weight refers to the number of 1's in the bit string. So, among all possible input differences and their corresponding output differences, the minimum value of the sum of hamming weights of input difference and output difference is the differential branching number. For example, if the minimum case for input and output difference's hamming weights are as shown, then the Sbox on the left has a differential branching number of 3 while the Sbox on the right has a differential branching number of 2.



The differential branching number of *PRESENT*'s Sbox is 3. In general, BN3 Sboxes are costly with respect to area. However, *GIFT*'s Sbox is a BN2 Sbox, thus it will take up lesser area.

1.2 Hardware Performance comparison

The following table [BPP+17] shows a comparison of the *GIFT* cipher on some hardware performance metrics with other common lightweight block ciphers.

We can see that *SKINNY* and *GIFT* both take up lesser area than *PRESENT*, as both *SKINNY* and *GIFT* have BN2 Sboxes. The unit for measuring area, GE, stands for Gate Equivalent. Gate Eequivalent is defined

	Area (GE)	Delay (ns)	Cycles	TP_{MAX} (MBit/s)	Power (μW) (@10MHz)	Energy (pJ)
GIFT-64-128	1345	1.83	29	1249.0	74.8	216.9
SKINNY-64-128	1477	1.84	37	966.2	80.3	297.0
PRESENT 64/128	1560	1.63	33	1227.0	71.1	234.6
SIMON 64/128	1458	1.83	45	794.8	72.7	327.3
GIFT-128-128	1997	1.85	41	1729.7	116.6	478.1
SKINNY-128-128	2104	1.85	41	1729.7	132.5	543.3
SIMON 128/128	2064	1.87	69	1006.6	105.6	728.6

as the physical area of a single NAND gate. Thus, *GIFT* performs better than *PRESENT* on many metrics.

2 Specifications

The original authors of the *GIFT* cipher proposed two versions of the cipher, *GIFT-64-128* and *GIFT-128-128*. Both versions have a key of size 128 bits, hence, we will refer to them as *GIFT-64* and *GIFT-128* from now onwards. *GIFT-64* has a block size of 64 bits and goes through 28 rounds. *GIFT-128* has a block size of 128 bits and goes through 40 rounds. We represent the number of rounds as N.

2.1 Round Function

Each round of the *GIFT* cipher has 3 steps: SubCells, PermBits and AddRoundKey. These steps are repeated N-1 times and for the final round, the PermBits operation is excluded as it does not provide any cryptographic value. This is because an attacker can easily invert the permutation at the last layer. So, a permutation operation is useful only if it is followed by an Sbox operation. If not, it only adds to the computation time of encryption and decryption making the cipher slightly slower.

2.1.1 SubCells

The SubCells operation is just a simple Sbox operation. *GIFT*'s Sbox is a 4-bit Sbox, so every 4 bits of the plaintext will be passed through *GIFT*'s Sbox. The hexadecimal form of the input and output of the Sbox is shown in the table below.

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

2.1.2 PermBits

The PermBits operation is a permutation of the bits of the *GIFT* state. A bit at position i of the *GIFT* state is mapped to the position P(i) given by these formulae.

$$P_{64}(i) = 4\left\lfloor \frac{i}{16} \right\rfloor + 16\left(\left(3\left\lfloor \frac{i\, mod 16}{4} \right\rfloor + (i\, mod 4)\right) mod 4\right) + (i\, mod 4)$$
$$P_{128}(i) = 4\left\lfloor \frac{i}{16} \right\rfloor + 32\left(\left(3\left\lfloor \frac{i\, mod 16}{4} \right\rfloor + (i\, mod 4)\right) mod 4\right) + (i\, mod 4)$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P_{64}(i)$	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
$P_{64}(i)$	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
$P_{64}(i)$	8	25	42	59	56	9	26	43	40	57	10	27	24	41	58	11
i	48	48	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P_{64}(i)$	12	29	46	63	60	13	30	47	44	61	14	31	28	45	62	15

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P_{128}(i)$	0	33	66	99	96	1	34	67	64	97	2	35	32	65	98	3
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P_{128}(i)$	4	37	70	103	100	5	38	71	68	101	6	39	36	69	102	7
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P_{128}(i)$	8	41	74	107	104	9	42	75	72	105	10	43	40	73	106	11
i	48	48	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P_{128}(i)$	12	45	78	111	108	13	46	79	76	109	14	47	44	77	110	15
i	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$P_{128}(i)$	0	33	66	99	96	1	34	67	64	97	2	35	32	65	98	3
i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$P_{128}(i)$	4	37	70	103	100	5	38	71	68	101	6	39	36	69	102	7
i	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
$P_{128}(i)$	24	57	90	123	120	25	58	91	88	121	26	59	56	89	122	27
i	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
$P_{128}(i)$	28	61	94	127	124	29	62	95	92	125	30	63	60	93	126	31

2.1.3 AddRoundKey

The AddRoundKey operation consists of adding the round key and the round constant to the *GIFT* state. Both the round key and the round constants are XORed with the state at specific bit positions. However, we make the distinction between round keys and round constants because the round key depends on the initial key chosen and hence, could be different for every encryption, whereas the round constants are same for any encryption because there is a fixed initial state for them.

Round Key Addition

The size of the round key is half of the block size. So, for *GIFT-64*, we have a 32-bit round key RK and for *GIFT-128*, we have a 64-bit

round key. The round key is further partitioned into two strings $U||V = u_{s-1}\dots u_0||v_{s-1}\dots v_0$ where $s = 16, 32$ *GIFT-64* and *GIFT-128* respectively. This round key is XORed at specific positions of the *GIFT* state as given below for *GIFT-64* and *GIFT-128* respectively.

$$b_{4i+1} \leftarrow b_{4i+1} \oplus u_i \text{ and } b_{4i} \leftarrow b_{4i} \oplus v_i \quad \forall i \in \{0, \dots, 15\}$$
$$b_{4i+2} \leftarrow b_{4i+2} \oplus u_i \text{ and } b_{4i+1} \leftarrow b_{4i+1} \oplus v_i \quad \forall i \in \{0, \dots, 31\}$$

Round Constant Addition

The round constant for both versions of the cipher consists of a 6-bit round constant $C = c_5c_4c_3c_2c_1c_0$ and a single bit '1'. This round constant is XORed at specific positions of the *GIFT* state as given below for *GIFT-64* and *GIFT-128* respectively.

$$b_{63} \leftarrow b_{63} \oplus 1, \quad b_{23} \leftarrow b_{23} \oplus c_5, \quad b_{19} \leftarrow b_{19} \oplus c_4, \quad b_{15} \leftarrow b_{15} \oplus c_3$$
$$b_{11} \leftarrow b_{11} \oplus c_2, \quad b_7 \leftarrow b_7 \oplus c_1, \quad b_3 \leftarrow b_3 \oplus c_0$$

$$b_{127} \leftarrow b_{63} \oplus 1, \quad b_{23} \leftarrow b_{23} \oplus c_5, \quad b_{19} \leftarrow b_{19} \oplus c_4, \quad b_{15} \leftarrow b_{15} \oplus c_3$$
$$b_{11} \leftarrow b_{11} \oplus c_2, \quad b_7 \leftarrow b_7 \oplus c_1, \quad b_3 \leftarrow b_3 \oplus c_0$$

Round Key Extraction and Key State Update

We get the round key for each round by extracting a set of bits from the key state. Then, the key schedule updates the key state to a new state from which the round key will be extracted for the next round. The 128-bit key is represented by $k_7||k_6||k_5||k_4||k_3||k_2||k_1||k_0$ (where k_i is a 16-bit string) which we call the key state.

For *GIFT-64*, two 16-bit strings RK = U||V form the round key, while for *GIFT-128*, two 32-bit strings RK = U||V form the round key. They are extracted as follows

$$U \leftarrow k_1, \quad V \leftarrow k_0$$
$$U \leftarrow k_5||k_4, \quad V \leftarrow k_1||k_0$$

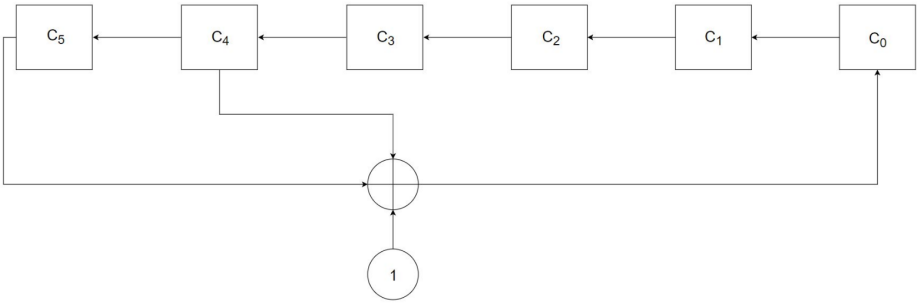
The key state is then updated by right rotating the key state by 2 (with fixed blocks) and then right rotating the overflowed blocks by 2 and 12 respectively.

$$k_7||k_6||k_5||k_4||k_3||k_2||k_1||k_0 \leftarrow k_1 \gg 2||k_0 \gg 12||k_7||k_6||k_5||k_4||k_3||k_2$$

Round Constant LFSR

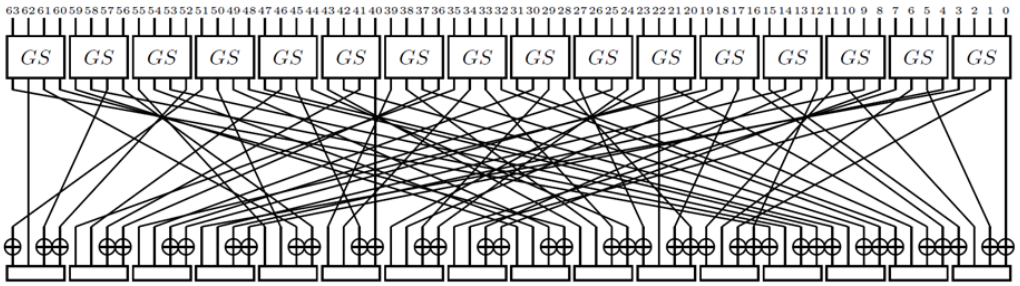
The round constant is taken from an LFSR (Linear Feedback Shift Register). This LFSR is the exact same one used by the *SKINNY* cipher. The six bits of the LFSR are initialized to 000001 and the following update function is used to get the round constants for the second round onwards.

$$(c_5, c_4, c_3, c_2, c_1, c_0) \leftarrow (c_4, c_3, c_2, c_1, c_0, c_5 \oplus c_4 \oplus 1)$$



2.2 Complete Picture

The three operations SubCells, PermBits and AddRoundKey together form one round of the *GIFT* cipher. One full round looks like this



3 SBox Analysis of GIFT Cipher

Substitution boxes are the non linear component of any cipher they provide confusion. This is GIFT Sbox:

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

3.1 Difference Distribution Table

- **Differential Branch Number = 2 :** The differential branch number of an S-Box S is defined as $\min_{v,w \neq v} \{wt(v \oplus w) + wt(S[v] \oplus S[w])\}$, wt here refers to the hamming weight.

- **Differential Unifromity = 6 :** This is equivalent to the highest value occurring in the DDT for any non zero input difference.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	2	0	2	2	2	2	2	0	0	2
2	0	0	0	0	0	4	4	0	0	2	2	0	0	2	2	0
3	0	0	0	0	0	2	2	0	2	0	0	2	2	2	2	2
4	0	0	0	2	0	4	0	6	0	2	0	0	0	2	0	0
5	0	0	2	0	0	2	0	0	2	0	0	0	2	2	2	4
6	0	0	4	6	0	0	0	2	0	0	2	0	0	0	2	2
7	0	0	2	0	0	2	0	0	2	2	2	4	2	0	0	0
8	0	0	0	4	0	0	0	4	0	0	4	0	0	0	0	4
9	0	2	0	2	0	0	2	2	2	0	2	0	2	2	0	0
a	0	4	0	0	0	0	4	0	0	2	2	0	0	2	2	0
b	0	2	0	2	0	0	2	2	2	2	0	0	2	0	2	0
c	0	0	4	0	4	0	0	0	2	0	2	0	2	0	2	0
d	0	2	2	0	4	0	0	0	0	0	2	2	0	2	0	2
e	0	4	0	0	4	0	0	2	2	0	0	2	2	2	0	0
f	0	2	2	0	4	0	0	0	0	2	0	2	0	0	2	2

3.2 Linear Approximation Table

The following table shows the LAT of the GIFT Sbox

Table a.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	2	-2	-2	2	4	0	0	-4	-2	-2	-2	-2
2	0	0	0	-4	0	4	0	0	2	2	2	-2	2	-2	2	2
3	0	0	0	-4	-2	-2	2	-2	2	-2	2	-2	-2	-2	0	0
4	0	0	0	0	0	0	0	-4	-4	0	0	0	0	0	4	-4
5	0	0	0	0	2	-2	2	-2	0	4	4	0	2	2	-2	-2
6	0	0	0	-4	4	0	0	0	-2	-2	-2	2	2	-2	-2	-2
7	0	0	0	4	2	2	2	-2	2	-2	-2	-2	4	0	0	0
8	0	0	0	0	0	-4	0	-4	0	0	0	0	0	-4	0	4
9	0	0	0	0	-2	-2	2	2	4	0	0	4	2	-2	2	-2
a	0	0	-4	0	0	0	4	0	-2	-2	2	-2	-2	-2	2	-2
b	0	0	4	0	2	-2	2	2	-2	2	-2	-2	0	0	4	0
c	0	4	4	0	0	0	0	0	0	-4	4	0	0	0	0	0
d	0	-4	4	0	-2	2	2	-2	0	0	0	0	-2	-2	-2	-2
e	0	-4	0	0	4	0	0	0	2	-2	2	2	-2	2	2	2
f	0	-4	0	0	-2	-2	-2	2	-2	-2	2	-2	4	0	0	0

4 Security Analysis of GIFT

4.1 Differential & Linear Cryptanalysis

Differential cryptanalysis and Linear cryptanalysis are amongst the strongest analysis methods for block ciphers like GIFT. One way to measure the resistance of GIFT is to study the lower bound of number of *active* SBoxes in a characteristic.

MILP (Mixed Integer Linear Programming) based approaches can be used to identify lower bounds in DC and LC for various rounds. MILP solution produces actual differential or linear characteristic, from which we can compute differential probablity from DDT of GIFT Sbox. The following table **(b.)** shows the number of active SBoxes upto 9 rounds.

Table. Lower-bound number of *active* SBoxes for Rounds 1 to 9

Table b.	1	2	3	4	5	6	7	8	9
DC	1	2	3	5	7	10	13	16	18
LC	1	2	3	5	7	9	12	15	18

The main goal of GIFT creators is to match the differential bounds of PRESENT, which averages to 2 active SBox per round, but with a lighter Sbox and without a constraint on differential branching number of 3. We will be discussing DC and LC on 9 rounds of GIFT.

Differential Cryptanalysis. To compute the 9 round differential probability of GIFT we first have to find a characteristic with least number of *active* SBoxes. Then we identify the next best characteristic keeping the input and output differences fixed. The search for other characteristics of the differential terminates when the contribution of them is insignificant towards the total differential probability.

For GIFT-64, it has a 9-round differential probability of $2^{-44.415}$, when compared to PRESENT which has a 9-round differential probability of $2^{-40.702}$. Considering an average probability of $2^{-4.935}$ and move forward it is expected that a 14-round version will result in a differential probability of 2^{-69}. Henceforth we can conclude the full 28-round version of GIFT-64 is resistant against DC.

For GIFT-128, it has a 9-round differential probability of $2^{-46.99}$ which translates to 2^{-127} in 26 rounds making the full 40-round version of GIFT-128 resistant against DC.

For both versions of GIFT the optimal characteristic had significantly lesser number of *active* SBoxes which makes the differential probablity approximately equal to the probability of optimal characteristic. This is not the case with PRESENT which has several characteristics that contribute significantly towards the differential probability due to its symmetric structure.

Linear Cryptanalysis. Given a linear characteristic of bias ϵ the square of the correlation contribution, called *correlation potential* is defined as $4\epsilon^2$. For an adversary to mount LC on an n-bit block cipher, they would require the correlation potential to be larger than 2^{-n}. To compute the r-round linear hull effect, which is the compile effect of multiple r-round linear characteristics under the same input and output masking, we use the following theorem. **Table a.** is the LAT of GIFT Sbox.

Theorem. *The average correlation potential (linear hull effect) between an input and an output selection pattern is the sum of the correlation potentials of all linear trails between the input and output selection patterns.*

Similar to differential, we first find an optimal linear characteristic, fix the input and output masking to find subsequent best possible linear characteristics and take the summation of the correlation potentials. The search is terminated when subsequent linear characteristic has insignificant contribution to the linear hull effect.

For GIFT-64, it has a 9-round linear hull effect of $2^{-49.99}$, which is expected to go down below 2^{-64} for 13-rounds. Therefore we can say that 28-round version of GIFT-64 is resistant to LC. This is a much stronger when compared to PRESENT which has a 9-round linear hull effect of $2^{-27.86}$.

For GIFT-128, it has a 9-round linear hull effect of $2^{-45.99}$, hence requires 27 rounds to achieve a correlation potential lower than 2^{-128}. Therefore the full version i.e., 40-rounds of GIFT-128 is resistant to LC.

5 Automated Cryptanalysis Modelling of GIFT

MILP (Mixed Integer Linear Programming) is often used to used to search for a lower bound on number of active SBoxes in differentials. Our main target is modelling of a differential hence we have to come with a set of linear inequalities that is able to capture the valid and invalid transitions of a DDT. For equation generator code visit https://github.com/LohitDaksha/GIFT-CS553

5.1 Substitution Layer Modelling

Now when it comes to modelling a DDT using boolean inequalities there are two major constraints one must overcome.

Constraint 1. How to generate a (possibly large) set of inequalities on variables $(x_0, ..., x_{n-1})$ that can capture the DDT?

Constraint2 . How to arrive at a smaller set of inequalities(to improve time efficiency of MILP solvers) that is still able to capture the DDT?

One approach to solve **Constraint 1** is called logical condition modelling and is based on the idea each impossible transition can be modelled by a sing linear inequality given below:

$$\sum_{i=0}^{m-1}(1-a_i)x_i + a_i(1-x_i) \geq 1$$

For any transition $(x_0, ..., x_{n-1}) \rightarrow (y_0, ..., y_{n-1})$ *or* $(x_n, ..., x_{m-1})$ *in DDT we represent a from impossible transitions, for example if input difference=1 can never go to output difference=6, we represent* $(100) \nrightarrow (011)$ *as* $a = (100011)$ *which gives an inequality* $-x_0 + x_1 + x_2 + y_0 - y_1 - y_2 \geq -2$.

However this will give us an extremely large number of equations for GIFT SBox which has 99 possible transition we will get 157 equations!

To reduce the number of equations and arrive at smaller set optimization algorithms like Espresso and Quine-McCluskey are used which we will not be discussing here.

5.2 Permutation Layer modelling

Permutation layer of gift is a well defined rearrangement of bits to model this aspect of the cipher we simply introduce new variables and equate them to appropriate positons of SBox output.

For example in the k^{th} round bit position $i = 2$ after substitution layer maps to position $i^{'} = 34$ after permutation layer, and to capture this behaviour we will have an equation $x_{34}^{k+1} = x_2^k$.

Hence to model 1 round of GIFT-64 we will have $16N + 64$ linear relations, where N represents the optimized number of equations capturing transitions of DDT.

6 Software Implementation

The encryption and decryption of GIFT-64 is present in the Github repo https://github.com/LohitDaksha/GIFT-CS553

7 Software Application

A software application using the GIFT cipher is present in the Github repo https://github.com/LohitDaksha/GIFT-CS553

References

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, Heidelberg, September 2017.