

GIFT-64-128 / GIFT-128-128

- Lightweight Block Cipher -

Ji Yong-Hyeon

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

February 15, 2024

List of Symbols

$x_{n-1} \parallel x_{n-2} \parallel \cdots \parallel x_0$ n -bit plaintext (x_0 is LSB)
 $k_7 \parallel k_6 \parallel \cdots \parallel k_0$ 128-bit key state

Contents

- 1 Specifications 1**
 - 1.1 Key Schedule and Round Constants 2
 - 1.2 Round Function 3
 - 1.2.1 SubCells 3
 - 1.2.2 PermBits 3
 - 1.2.3 AddRoundKey 4
- A Generation of Tables 5**
 - A.1 Round Constants 5
 - A.2 GIFT S-BOX 6

Chapter 1

Specifications

Overview

Specification	GIFT-64-128	GIFT-128-128
Block Size (bits)	64	128
Key Size (bits)	128	128
Round Key Size (bits)	32	64
Number of Rounds	28	40
Design Strategy	Substitution-Permutation Network	Substitution-Permutation Network

Table 1.1: Specifications of GIFT-64-128 and GIFT-128-128

1.1 Key Schedule and Round Constants

Rounds	Constants
1 - 16	01, 03, 07, 0F, 1F, 3E, 3D, 3B, 37, 2F, 1E, 3C, 39, 33, 27, 0E
17 - 32	1D, 3A, 35, 2B, 16, 2C, 18, 30, 21, 02, 05, 0B, 17, 2E, 1C, 38
33 - 48	31, 23, 06, 0D, 1B, 36, 2D, 1A, 34, 29, 12, 24, 08, 11, 22, 04

Table 1.2: Round Constants generated by 6-bit affine LFSR

```

1  const u8 rCon[48] = {
2      0x01U, 0x03U, 0x07U, 0x0FU, 0x1FU, 0x3EU, 0x3DU, 0x3BU,
3      0x37U, 0x2FU, 0x1EU, 0x3CU, 0x39U, 0x33U, 0x27U, 0x0EU,
4      0x1DU, 0x3AU, 0x35U, 0x2BU, 0x16U, 0x2CU, 0x18U, 0x30U,
5      0x21U, 0x02U, 0x05U, 0x0BU, 0x17U, 0x2EU, 0x1CU, 0x38U,
6      0x31U, 0x23U, 0x06U, 0x0DU, 0x1BU, 0x36U, 0x2DU, 0x1AU,
7      0x34U, 0x29U, 0x12U, 0x24U, 0x08U, 0x11U, 0x22U, 0x04U
8  };

```

1.2 The Round Function

1.2.1 SubCells

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

Table 1.3: Specifications of GIFT Sbox GS

```

1  const u8 GS[16] = {
2      0x1U, 0xAU, 0x4U, 0xCU, 0x6U, 0xFU, 0x3U, 0x9U,
3      0x2U, 0xDU, 0xBU, 0x7U, 0x5U, 0x0U, 0x8U, 0xEU
4  };
5
6  const u8 invGS[16] = {
7      0xDU, 0x0U, 0x8U, 0x6U, 0x2U, 0xCU, 0x4U, 0xBU,
8      0xEU, 0x7U, 0x1U, 0xAU, 0x3U, 0x9U, 0xFU, 0x5U
9  };

```

1.2.2 PermBits

The permutation can be expressed as:

$$P_{64}(i) = 4 \cdot \left\lfloor \frac{i}{16} \right\rfloor + 16 \cdot \left[\left(3 \cdot \left\lfloor \frac{i \bmod 16}{4} \right\rfloor + (i \bmod 4) \right) \bmod 4 \right] + (i \bmod 4).$$

$$x_{P(i)} \leftarrow x_i$$

for $i \in \{0, \dots, n-1\}$.

```

1  for (u8 i = 0; i < 64; i++) {
2      permBits[i] =
3          4 * (i / 16) +
4          16 * ((3 * ((i % 16) / 4) + (i % 4)) % 4) +
5          (i % 4);
6  }

```

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	49	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

Table 1.4: Specifications of GIFT-64 Bit Permutation

```

1  const u8 permBits[64] = {
2      0x00U, 0x11U, 0x22U, 0x33U, 0x30U, 0x01U, 0x12U, 0x23U,
3      0x20U, 0x31U, 0x02U, 0x13U, 0x10U, 0x21U, 0x32U, 0x03U,
4      0x04U, 0x15U, 0x26U, 0x37U, 0x34U, 0x05U, 0x16U, 0x27U,
5      0x24U, 0x35U, 0x06U, 0x17U, 0x14U, 0x25U, 0x36U, 0x07U,
6      0x08U, 0x19U, 0x2AU, 0x3BU, 0x38U, 0x09U, 0x1AU, 0x2BU,
7      0x28U, 0x39U, 0x0AU, 0x1BU, 0x18U, 0x29U, 0x3AU, 0x0BU,
8      0x0CU, 0x1DU, 0x2EU, 0x3FU, 0x3CU, 0x0DU, 0x1EU, 0x2FU,
9      0x2CU, 0x3DU, 0x0EU, 0x1FU, 0x1CU, 0x2DU, 0x3EU, 0x0FU
10 };
11
12 const u8 invPermBits[64] = {
13     0x00U, 0x05U, 0x0AU, 0x0FU, 0x10U, 0x15U, 0x1AU, 0x1FU,
14     0x20U, 0x25U, 0x2AU, 0x2FU, 0x30U, 0x35U, 0x3AU, 0x3FU,
15     0x0CU, 0x01U, 0x06U, 0x0BU, 0x1CU, 0x11U, 0x16U, 0x1BU,
16     0x2CU, 0x21U, 0x26U, 0x2BU, 0x3CU, 0x31U, 0x36U, 0x3BU,
17     0x08U, 0x0DU, 0x02U, 0x07U, 0x18U, 0x1DU, 0x12U, 0x17U,
18     0x28U, 0x2DU, 0x22U, 0x27U, 0x38U, 0x3DU, 0x32U, 0x37U,
19     0x04U, 0x09U, 0x0EU, 0x03U, 0x14U, 0x19U, 0x1EU, 0x13U,
20     0x24U, 0x29U, 0x2EU, 0x23U, 0x34U, 0x39U, 0x3EU, 0x33U
21 };

```

1.2.3 AddRoundKey

Appendix A

Generation of Tables

A.1 Round Constants

Note (LFSR).

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

```
1 void generate_round_constants(u8 rCon[48]) {
2     u8 state = 0x01;
3     rCon[0] = state;
4
5     for (u8 i = 1; i < 48; i++) {
6         bool new_bit =
7             ((rCon[i-1] >> 5) & 0x01) ^
8             ((rCon[i-1] >> 4) & 0x01) ^ 0x01;
9         state <<= 1;
10        state |= new_bit;
11
12        rCon[i] = state & 0x3F; // 3F = 0011 1111
13    }
14 }
```

A.2 GIFT S-BOX

```
1 void generate_sbox(u8 S[16]) {
2     bool buffer[4], tmp;
3
4     for (u8 i = 0; i < 16; i++) {
5         buffer[0] = (i >> 0) & 1;
6         buffer[1] = (i >> 1) & 1;
7         buffer[2] = (i >> 2) & 1;
8         buffer[3] = (i >> 3) & 1;
9
10        buffer[1] = buffer[1] ^ (buffer[0] & buffer[2]);
11        tmp      = buffer[0] ^ (buffer[1] & buffer[3]);
12        buffer[2] = buffer[2] ^ (tmp      | buffer[1]);
13        buffer[0] = buffer[3] ^ buffer[2];
14        buffer[1] = buffer[1] ^ buffer[0];
15        buffer[0] = !(buffer[0]);
16        buffer[2] = buffer[2] ^ (tmp      & buffer[1]);
17        buffer[3] = tmp;
18
19        S[i] = (u8)buffer[3] << 3 |
20        (u8)buffer[2] << 2 |
21        (u8)buffer[1] << 1 |
22        (u8)buffer[0];
23    }
24 }
```

Bibliography

- [1] 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 (Full version)*. Temasek Laboratories, Nanyang Technological University, Singapore; School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore; School of Computer Science and Engineering, Nanyang Technological University, Singapore; NTT Secure Platform Laboratories, Japan; LASEC, École Polytechnique Fédérale de Lausanne, Switzerland. Emails: bsubhadeep@ntu.edu.sg, emailpandey@gmail.com, thomas.peyrin@ntu.edu.sg, SSIM011@e.ntu.edu.sg, Todo.Yosuke@lab.ntt.co.jp, Sasaki.Yu@lab.ntt.co.jp
- [2] Subhadeep Banik, Andrey Bogdanov, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, Elmar Tischhauser, and Yosuke Todo, “SUNDAE-GIFT v1.0,” *Nanyang Technological University, Singapore and Temasek Labs@NTU, Singapore and LASEC, Ecole Polytechnique Fédérale de Lausanne, Switzerland and Technical University of Denmark, Denmark and Cybercrypt A/S, Denmark and NTT Secure Platform Laboratories, Japan*, 2023. Note: Emails: thomas.peyrin@ntu.edu.sg, crypto.s.m.sim@gmail.com, subhadeep.banik@epfl.ch, anbog@dtu.dk, elmar@cybercrypt.com, Sasaki.Yu@lab.ntt.co.jp, yosuke.todo.xt@hco.ntt.co.jp.