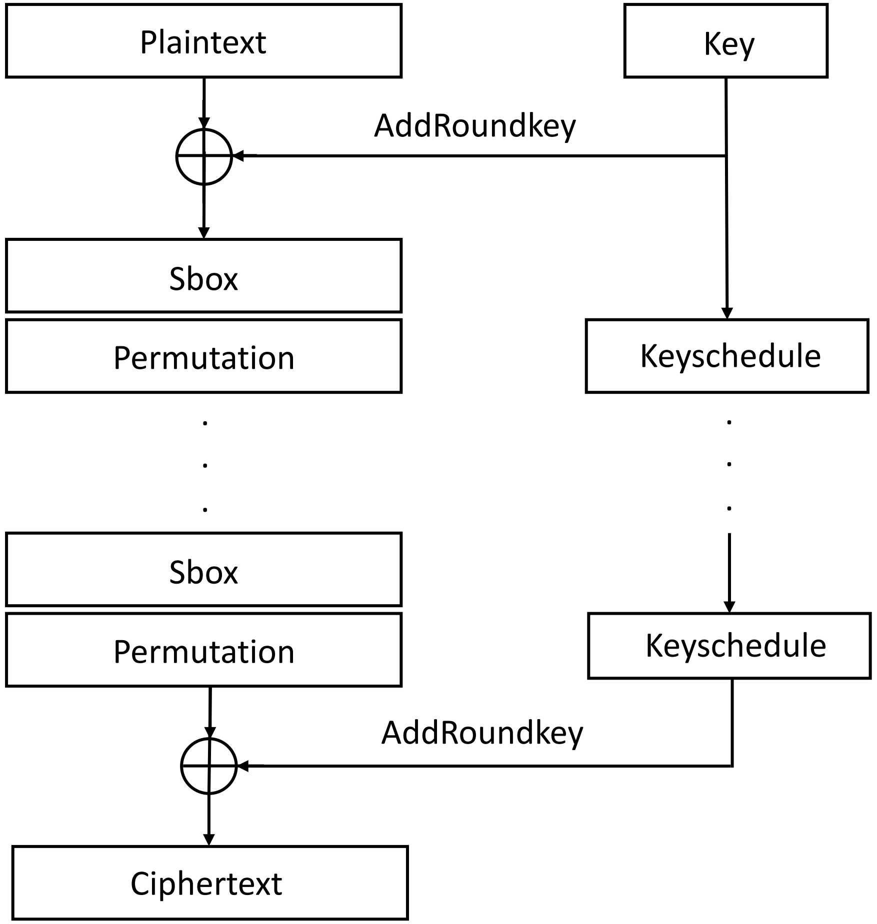
# PRESENT 양자 구현

https://youtu.be/_KgJ4qEyMGE

한성대학교
HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

# PRESENT




PRESENT 암호화 구조

- 64-bit 평문, (80, 128)-bit 키
- 31 Round
- SPN 구조

# AddRoundkey

**addRoundKey.** Given round key $K_i = \kappa_{63}^i \dots \kappa_0^i$ for $1 \leq i \leq 32$ and current STATE $b_{63} \dots b_0$, addRoundKey consists of the operation for $0 \leq j \leq 63$,

$$b_j \rightarrow b_j \oplus \kappa_j^i.$$

- 모든 Round가 끝나고 AddRoundkey를 마지막으로 수행 ( 총 32번 수행)
- 80 bit, 128bit 키 중, 가장 왼쪽 64-bit를 추출하여 라운드 키로 사용

```
def Add_RK(eng, b, k):
    for i in range(64):
        CNOT | (k[i], b[i])
```

매개변수만 조정

# Sbox

**sBoxlayer.** The S-box used in PRESENT is a 4-bit to 4-bit S-box $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. The action of this box in hexadecimal notation is given by the following table.

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | C | 5 | 6 | B | 9 | 0 | A | D | 3 | E | F | 8 | 4 | 7 | 1 | 2 |

- PRESENT 논문에서 확인 못하였음

**4.3 The S-box.**

We use a single 4-bit to 4-bit S-box $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ in PRESENT. This is a direct consequence of our pursuit of hardware efficiency, with the implementation of such an S-box typically being much more compact than that of an 8-bit S-box. Since we use a bit permutation for the linear diffusion layer, AES-like diffusion techniques [12] are not an option for PRESENT. Therefore we place some additional conditions on the S-boxes to improve the so-called *avalanche of change*. More precisely, the S-box for PRESENT fullfils the following conditions, where we denote the Fourier coefficient of $S$ by

$$S_b^W(a) = \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle b, S(x) \rangle + \langle a, x \rangle}.$$

1. For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed non-zero output difference $\Delta_O \in \mathbb{F}_2^4$ we require
$$\#\{x \in \mathbb{F}_2^4 \,|\, S(x) + S(x + \Delta_I) = \Delta_O\} \leq 4.$$
2. For any fixed non-zero input difference $\Delta_I \in \mathbb{F}_2^4$ and any fixed output difference $\Delta_O \in \mathbb{F}_2^4$ such that $\mathrm{wt}(\Delta_I) = \mathrm{wt}(\Delta_O) = 1$ we have
$$\{x \in \mathbb{F}_2^4 \,|\, S(x) + S(x + \Delta_I) = \Delta_O\} = \emptyset.$$
3. For all non-zero $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_4$ it holds that $|S_b^W(a)| \leq 8$.
4. For all $a \in \mathbb{F}_2^4$ and all non-zero $b \in \mathbb{F}_4$ such that $\mathrm{wt}(a) = \mathrm{wt}(b) = 1$ it holds that $S_b^W(a) = \pm 4$.

# Sbox

- PRESENT 관련 몇몇 논문에서 아래 Sbox 연산 수식을 언급

$$f0 = a\,b'c + a\,b'd + a'c'd' + a'b\,c + a'c\,d \quad (1)$$

$$f1 = a\,b\,c' + b'c\,d' + a'b\,c\,d + a'b'c' + b'c'd \quad (2)$$

$$f2 = a'b'c + a\,b\,d + a'c\,d' + a\,b'c' + a\,b'd' \quad (3)$$

$$f3 = a'b'd + a'c\,d + a\,b'd' + a\,c\,d' + a\,b\,c'd + a'b\,c'd' \quad (4)$$

- AND 연산, NOT 연산, XOR 연산
- Sbox( a , b, c, d ) → (f0, f1, f2, f3)   Example : Sbox(0) = C (1100)
- 비효율적인 구현 예상 ( 추가 큐비트 ?)

# Sbox

**#BAKSI ANUBHAB#** <ANUBHAB001@e.ntu.edu.sg> 11월 16일 (월) 오전 2:11

나, khj930704@gmail.com, shuraatum@gmail.com, hwajeong84@gmail.com에게

영어 > 한국어 메일 번역 영어 번역 안함

Dear Kyoungbae, Hyunjun, Siwoo and Hwajeong,

I have just come across your eprint paper on "Grover on GIFT". I haven't gone through the details yet though; it looks promising. Given the emerging threat of quantum computers, this type of research will become more important in the coming future.

I just would like to give my two cents. We have done a short research on reversible implementation of 4x4 SBoxes [paper, source codes: https://github.com/vdasu/lighter-r]. It can take any 4x4 SBox and give its reversible implementation w. r. t. some predefined cost for the gates. I am hoping you'll find it worth your time.

With regards,
Anubhab

- 모든 4-bit Sbox 의 input, output 매칭만으로 양자 회로로 바꿀 수 있음
- 기존 컴퓨터에서 사용하던 LIGHTER 라는 툴을 양자 컴퓨터에서도 사용할 수 있게끔 확장한 논문

**LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes**

Vishnu Asutosh Dasu*, Anubhab Baksi†, Sumanta Sarkar‡ and Anupam Chattopadhyay§
*Manipal Institute of Technology, Manipal, India
†§School of Computer Science & Engineering, Nanyang Technological University, Singapore
‡TCS Innovation Labs, Hyderabad, India
*vishnu.asutosh@learner.manipal.edu, †anubhab001@e.ntu.edu.sg,
‡sumanta.sarkar1@tcs.com, §anupam@ntu.edu.sg

# Sbox

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

**TABLE II:** LIGHTER encoding: Example with PRINCE SBox (BF32AC916780E5D4)

| Look-up Based SBox → | B | F | 3 | 2 | A | C | 9 | 1 | 6 | 7 | 8 | 0 | E | 5 | D | 4 | Encoded SBox ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $z_3$ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | CE2A |
| $z_2$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 44CF |
| $z_1$ | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | F8C8 |
| $z_0$ | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | E346 |
| $z_0 \oplus z_1$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1B8E |
| $z_0 \wedge z_1$ | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | E040 |

*C. Results*

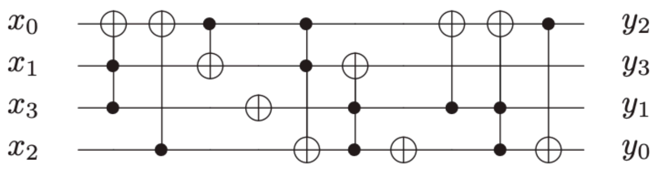



**FIG. 4:** SKINNY SBox (C6901A2B385D4E7F) with NCT (optimized for quantum cost)

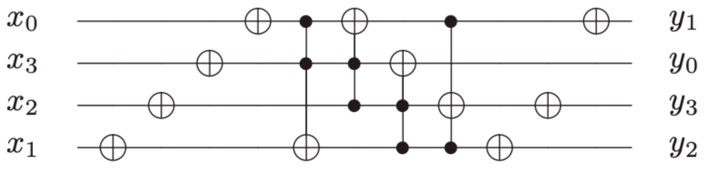



**FIG. 5:** PICCOLO SBox (E4B238091A7F6C5D) with MCT (optimized for two-qubit cost)

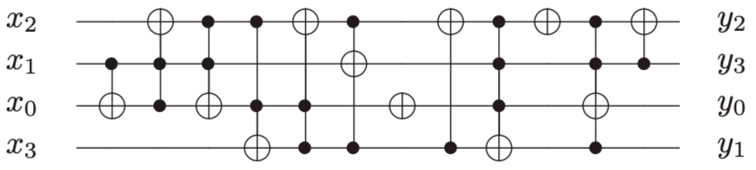



**FIG. 6:** PRINCE SBox (BF32AC916780E5D4) with MCT (optimized for gate count)

- input, output 매칭이 안됨
- 이해 어려움

# Sbox

경배 <starj1023@gmail.com>
ANUBHAB001에게

Dear Baksi,

I am looking at your Sbox implementation paper to expand our research.

Is the figure 6 below a completed PRINCE sbox quantum circuit?

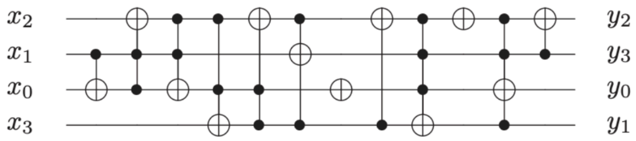



**FIG. 6:** PRINCE SBox (BF32AC916780E5D4) with MCT (optimized for gate count)

How does the implementation of the figure 6 match the input value to the output value? (0123456789ABCDEF -> BF32AC916780E5D4)

Is something omitted?

Thank you!

**#BAKSI ANUBHAB#**
나에게

영어 > 한국어 메일 번역

Hi 경배,

Please find our code to crosscheck the implementation of the PRINCE SBox (Python file). The output from the LIGHTER-R tool is given in the attached C file "BF32AC916780E5D4" -f "libraries/MCT_gc.conf" -s "PRINCE").

For example, if you set (x0, x1, x2, x3) = (0, 0, 0, 0) = 0x0,
y2 is flipped from x2, so y2 = 1
y3 is flipped from x1, so y3 = 1
y0 is flipped from x0, so y0 = 1
y1 is same as x3, so y1 = 0
so, (y0, y1, y2, y3) = (1, 0, 1, 1) = 0xb
So the input 0x0 is mapped to 0xb (this is the leftmost look-up value in the PRINCE SBox: BF32AC916780E5D4)
In this way, all the input values are mapped to the corresponding look-up values.

I am not sure if I understood your question, so please let me know if this answers it.

Thanks and regards,
Anubhab Baksi

...

...

...

# Sbox

```
a@a-NUC8i5BEH:~/바탕화면/lighter-r-master$ ./non-lin-search -qvw -o "1a4c6f392db7508e" -f "libraries/MCT_gc.conf" -s "GIFT"

From : 00FF 0F0F 3333 5555
To   : 5563 3C59 4EB1 8778
0
100
200
Generating implementation 0
(cost : 400 + 400 -> 8GE)
300
400
```

GIFT Sbox

```
a@a-NUC8i5BEH:~/바탕화면/lighter-r-master$  ./non-lin-search -qvw -o "c56b90ad3ef84712" -f "libraries/MCT_gc.conf" -s "PRESENT_sbox"

From : 00FF 0F0F 3333 5555
To   : 9B70 E16C 32E5 59A6
0
100
200
300
400
Generating implementation 0
(cost : 600 + 500 -> 11GE)
500
```

PRESENT Sbox

```
LICENSE          bool_op.cpp   f1_list_100.txt  f1_list_500.txt  f2_list_100.txt  f2_list_500.txt  implementation_GIFT_0.c          main.o     non-lin-search     string_bool_op.o
Makefile         bool_op.o     f1_list_200.txt  f1_list_600.txt  f2_list_200.txt  f2_list_600.txt  implementation_PRESENT_sbox_0.c  mitm.cpp   reversible-helper  test.c
README-lighter   f1_infos.txt  f1_list_300.txt  f2_infos.txt     f2_list_300.txt  impl_info.cpp    libraries                        mitm.h     static_sort.h      utils.cpp
README.md        f1_list_0.txt f1_list_400.txt  f2_list_0.txt    f2_list_400.txt  impl_info.o      main.cpp                         mitm.o     string_bool_op.cpp utils.o
```

# Sbox

```
// from : 00FF 0F0F 3333 5555

F[0] = X[0];
F[1] = X[2];
F[2] = X[1];
F[3] = X[3];

F[1] = CCNOT2(F[3], F[2], F[1]);
F[3] = CCNOT2(F[1], F[0], F[3]);
F[0] = CNOT1(F[0], F[2]);
F[2] = CNOT1(F[2], F[1]);
F[1] = RNOT1(F[1]);
F[0] = CCNOT2(F[3], F[1], F[0]);
F[1] = CNOT1(F[1], F[0]);
F[2] = CCNOT2(F[3], F[1], F[2]);

X[0] = F[3];
X[1] = F[2];
X[2] = F[0];
X[3] = F[1];

//to : 5563 3C59 4EB1 8778
// Cost : 8
// Logic Library : MCT_gc
```

GIFT Sbox

```
// from : 00FF 0F0F 3333 5555

F[0] = X[0];
F[1] = X[2];
F[2] = X[3];
F[3] = X[1];

F[2] = RNOT1(F[2]);
F[3] = CCNOT2(F[2], F[1], F[3]);
F[0] = CCNOT2(F[3], F[1], F[0]);
F[2] = CNOT1(F[2], F[0]);
F[1] = CCCNOT2(F[0], F[2], F[3], F[1]);
F[3] = RNOT1(F[3]);
F[1] = CCNOT2(F[3], F[0], F[1]);
F[0] = CNOT1(F[0], F[3]);
F[3] = CNOT1(F[3], F[2]);
F[2] = CCCNOT2(F[0], F[1], F[3], F[2]);
F[0] = CCNOT2(F[2], F[1], F[0]);

X[0] = F[2];
X[1] = F[0];
X[2] = F[1];
X[3] = F[3];

//to : 9B70 E16C 32E5 59A6
// Cost : 11
// Logic Library : MCT_gc
```

PRESENT Sbox

# Sbox

- 라이브러리 내 다양한 버전 존재

```
a@a-NUC8i5BEH:~/바탕화면/lighter-r-master/libraries$ ls
CPU.conf  MCT_2qbc.conf  MCT_gc.conf  MCT_qc.conf  NCT_2qbc.conf  NCT_gc.conf  NCT_qc.conf  TSMC65nm.conf  UMC180nm.conf
```

```
// from : 00FF 0F0F 3333 5555

F[0] = X[0];
F[1] = X[2];
F[2] = X[3];
F[3] = X[1];

F[2] = RNOT1(F[2]);
F[3] = CCNOT2(F[2], F[1], F[3]);
F[0] = CCNOT2(F[3], F[1], F[0]);
F[2] = CNOT1(F[2], F[0]);
F[1] = CCCNOT2(F[0], F[2], F[3], F[1]);
F[3] = RNOT1(F[3]);
F[1] = CCNOT2(F[3], F[0], F[1]);
F[0] = CNOT1(F[0], F[3]);
F[3] = CNOT1(F[3], F[2]);
F[2] = CCCNOT2(F[0], F[1], F[3], F[2]);
F[0] = CCNOT2(F[2], F[1], F[0]);

X[0] = F[2];
X[1] = F[0];
X[2] = F[1];
X[3] = F[3];

//to : 9B70 E16C 32E5 59A6
// Cost : 11
// Logic Library : MCT_gc
```

PRESENT MCT 버전

```
// from : 00FF 0F0F 3333 5555

F[0] = X[1];
F[1] = X[2];
F[2] = X[3];
F[3] = X[0];

F[1] = CNOT1(F[1], F[0]);
F[3] = CCNOT2(F[1], F[0], F[3]);
F[0] = CCNOT2(F[3], F[1], F[0]);
F[1] = CCNOT2(F[2], F[0], F[1]);
F[0] = CNOT1(F[0], F[3]);
F[3] = RNOT1(F[3]);
F[0] = CNOT1(F[0], F[1]);
F[2] = CNOT1(F[2], F[3]);
F[1] = CNOT1(F[1], F[2]);
F[2] = RNOT1(F[2]);
F[3] = CCNOT2(F[1], F[0], F[3]);

X[0] = F[1];
X[1] = F[3];
X[2] = F[0];
X[3] = F[2];

//to : 9B70 E16C 32E5 59A6
// Cost : 11
// Logic Library : NCT_gc
```

PRESENT NCT 버전

더 좋음

# Sbox

```
def p_sbox(eng, b):

    X | b[0]
    Toffoli | (b[0], b[1], b[2])
    Toffoli | (b[2], b[1], b[3])

    CNOT | (b[3], b[0])

    with Control(eng, b[3]):
        Toffoli | (b[0], b[2], b[1])

    X| b[2]

    Toffoli | (b[2], b[3], b[1])
    CNOT | (b[2], b[3])
    CNOT | (b[0], b[2])

    with Control(eng, b[3]):
        Toffoli | (b[1], b[2], b[0])

    Toffoli | (b[0], b[1], b[3])

    Swap | (b[3], b[0])
    Swap | (b[0], b[2])
```

- 현재 Sbox 는 MCT 버전 → 제일 좋은 버전으로 바꿀 예정

# Permutation

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

20 x 3 = 60

```
def Permutation(eng, b):

    Swap|(b[1], b[4]) # b[1] = 4
    Swap|(b[16], b[4]) # b[16] = 1 b[4] = 16
```

.

.

.

# Keyshcedule

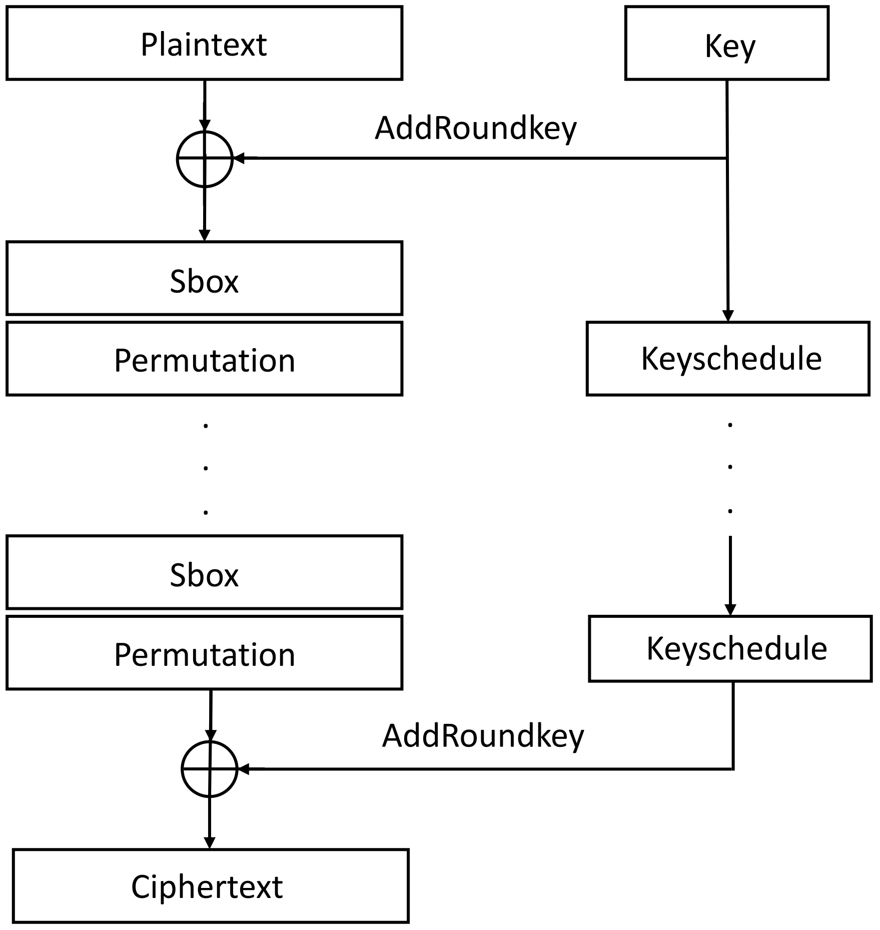
1. $[\mathtt{k}_{79}\mathtt{k}_{78}\ldots\mathtt{k}_1\mathtt{k}_0] = [\mathtt{k}_{18}\mathtt{k}_{17}\ldots\mathtt{k}_{20}\mathtt{k}_{19}]$ Rotation
2. $[\mathtt{k}_{79}\mathtt{k}_{78}\mathtt{k}_{77}\mathtt{k}_{76}] = S[\mathtt{k}_{79}\mathtt{k}_{78}\mathtt{k}_{77}\mathtt{k}_{76}]$ Sbox
3. $[\mathtt{k}_{19}\mathtt{k}_{18}\mathtt{k}_{17}\mathtt{k}_{16}\mathtt{k}_{15}] = [\mathtt{k}_{19}\mathtt{k}_{18}\mathtt{k}_{17}\mathtt{k}_{16}\mathtt{k}_{15}] \oplus \mathtt{round\_counter}$ Constant XOR

```
def keySchedule(eng, k, ctr):
    ctr = ctr+1

    for j in range(19):
        for i in range(79):
            Swap | (k[i], k[i+1])

    p_sbox(eng, k[76:80])
    CTR_XOR(eng, k[15:20], ctr)

    return ctr
```

```
def CTR_XOR(eng, k, ctr):

    if (ctr >= 16):
        X | (k[4])
        ctr = ctr - 16
    if (ctr >= 8):
        X | (k[3])
        ctr = ctr - 8
    if (ctr >= 4):
        X | (k[2])
        ctr = ctr - 4
    if (ctr >= 2):
        X | (k[1])
        ctr = ctr - 2
    if (ctr >= 1):
        X | (k[0])
        ctr = ctr - 1
```

# Encryption

```python
def Enc(eng):

    b = eng.allocate_qureg(64)
    k = eng.allocate_qureg(80)

    #All(X) | b
    #All(X) | k
    ctr = 0

    for j in range(31):

        Add_RK(eng, b, k[16:80]) #Addroundeky

        ctr = keySchedule(eng, k, ctr)  # Keyschedule

        for i in range(16):      #Sbox
            p_sbox(eng, b[4*i : 4*(i+1)])

        Permutation(eng, b)      #Permutation

    Add_RK(eng, b, k[16:80])  # Addroundeky

    print(ctr)
    All(Measure) | b

    for i in range (64):
        print(int(b[63-i]), end=' ')
```



# Test Vector

## Appendix I

Test vectors for PRESENT with an 80-bit key are shown in hexadecimal notation.

| *plaintext* | *key* | *ciphertext* |
|---|---|---|
| 00000000 00000000 | 00000000 00000000 0000 | 5579C138 7B228445 |
| 00000000 00000000 | FFFFFFFF FFFFFFFF FFFF | E72C46C0 F5945049 |
| FFFFFFFF FFFFFFFF | 00000000 00000000 0000 | A112FFC7 2F68417B |
| FFFFFFFF FFFFFFFF | FFFFFFFF FFFFFFFF FFFF | 3333DCD3 213210D2 |

PRESENT ×

```
/Users/kb/PycharmProjects/projectq/venv/bin/python /Users/kb/PycharmProjects/projectq/Lightweight_crypto/PRESENT_80.py
31
```

#All(X) | b
#All(X) | k

```
0 1 0 1 0 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1
```

5 5 4 5

#All(X) | b
All(X) | k

```
1 1 1 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1
```

All(X) | b
#All(X) | k

```
1 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1
```

All(X) | b
All(X) | k

```
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 0
```

# Resource ( Sbox 는 변경 예정 )

- PRESENT (80-bit 키)

```
Gate counts:
    Allocate : 144
    CCCX : 1054
    CCX : 2108
    CX : 3629
    Deallocate : 144
    Measure : 64
    Swap : 48825
    X : 1134

Depth : 1824.
```

```
Gate counts:
    Allocate : 128
    CCCX : 1054
    CCX : 2108
    CX : 3629
    Deallocate : 128
    Measure : 64
    X : 1118

Depth : 374.
```

Swap 게이트 무시

- PRESENT (128 - bit 키)

```
Gate counts:
    Allocate : 128
    CCCX : 1116
    CCX : 2232
    CX : 3722
    Deallocate : 128
    Measure : 64
    X : 1164

Depth : 374.
```

Swap 게이트 무시

# 감사합니다