SEED 양자회로 구현

https://www.youtube.com/watch?v=cNgDDOQaIlE

HANSUNG UNIVERSITY

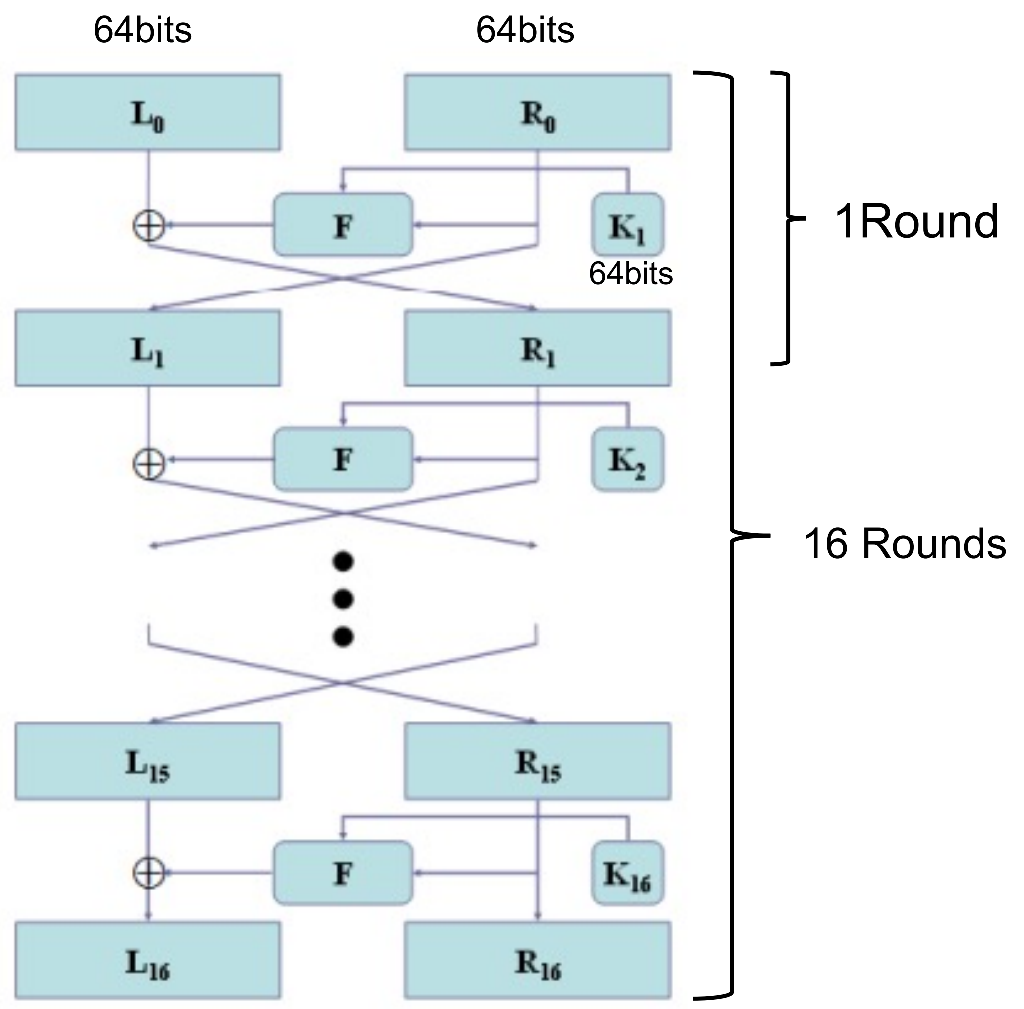
CryptoCraft LAB

SEED 구조

- ## SEED 구조

전체 구조는 Feistel 구조로 이루어짐.
128 비트 평문 블록과 128비트 키를 사용
128비트 평문을 각각 64비트씩 좌우로 나누어 연산
총 16라운드 진행

* 128비트 키를 사용한다는 것은 K_i가 128비트가 아닌
128비트키를 이용하여 K_i를 생성.
K_i 는 64비트



SEED - 키스케줄

- **라운드 키 생성**

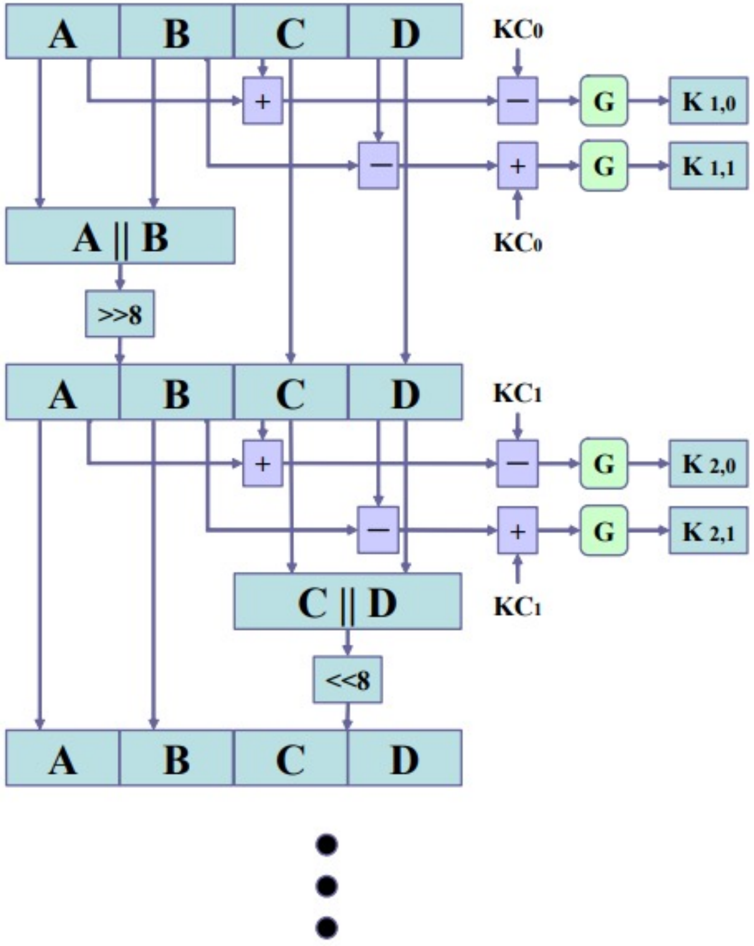
128비트 키 $K = A||B\ ||C||D$ $(A, B, C, D$ 32**비트)**

$K_{i,0}, K_{i,1}$ **= 32비트**

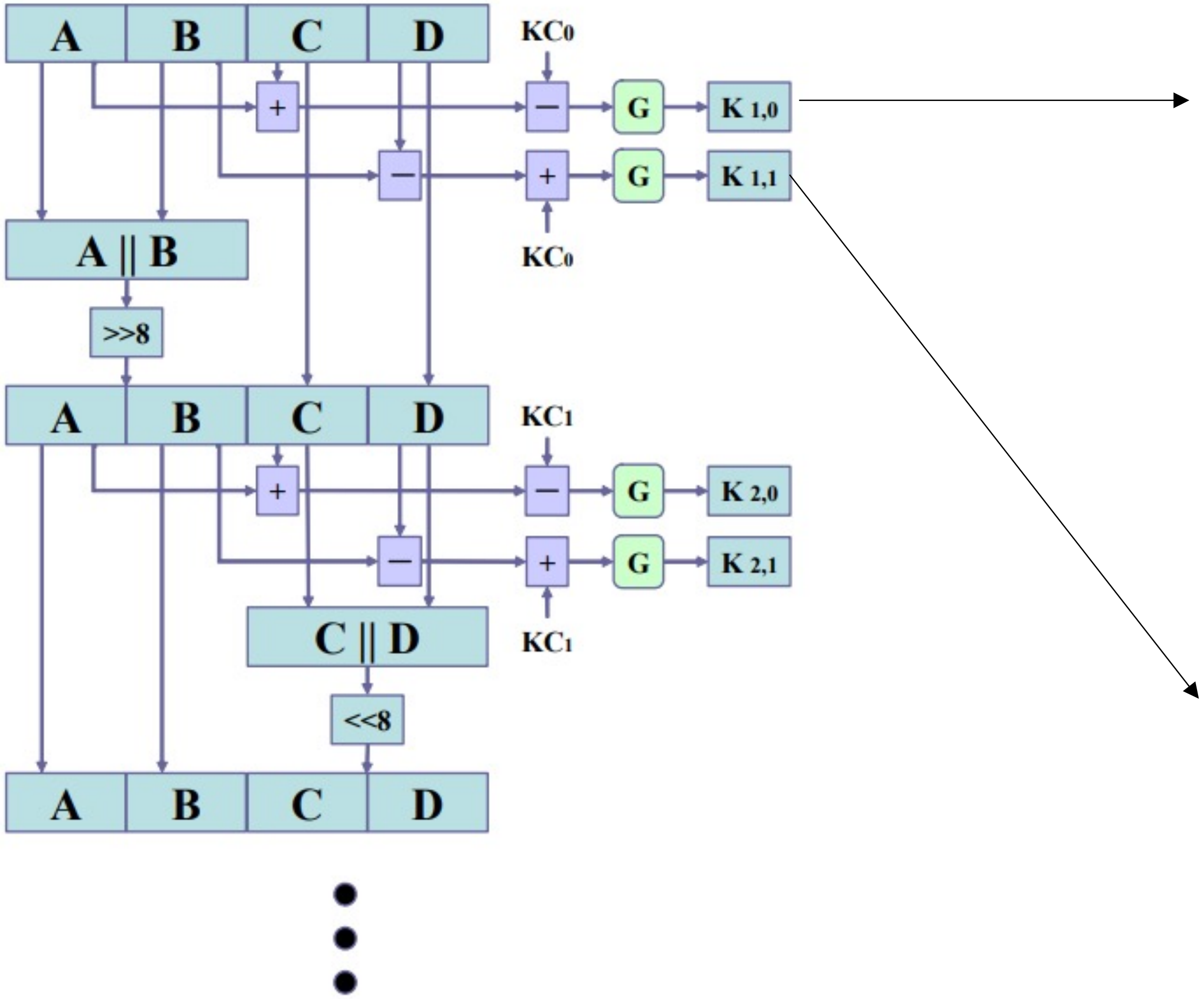
홀수 라운드일 경우 : $A||B$ **Right shift 8**

짝수 라운드일 경우 : $C||D$ **Left shift 8**

```
for i in range(16):
    Round_constant_XOR(eng, KC_q, KC[i], 32)
    Round_constant_XOR(eng, KC_q2, KC[i], 32)
    key0, C = KeySched0(eng, A, C, KC_q, ancilla)
    key1, D = KeySched1(eng, B, D, KC_q2, ancilla1)
    Round_constant_XOR(eng, KC_q, KC[i], 32)      #reverse, KC_q =0
    Round_constant_XOR(eng, KC_q2, KC[i], 32)
    L0,L1,R0,R1 = encrypt(eng, L0, L1, R0, R1, key0, key1, ancilla)

    if(i%2 ==0):
        A,B=  RightShift(eng, A, B)
    else :
        C,D = LeftShift(eng, C, D)
```



SEED - 키스케줄



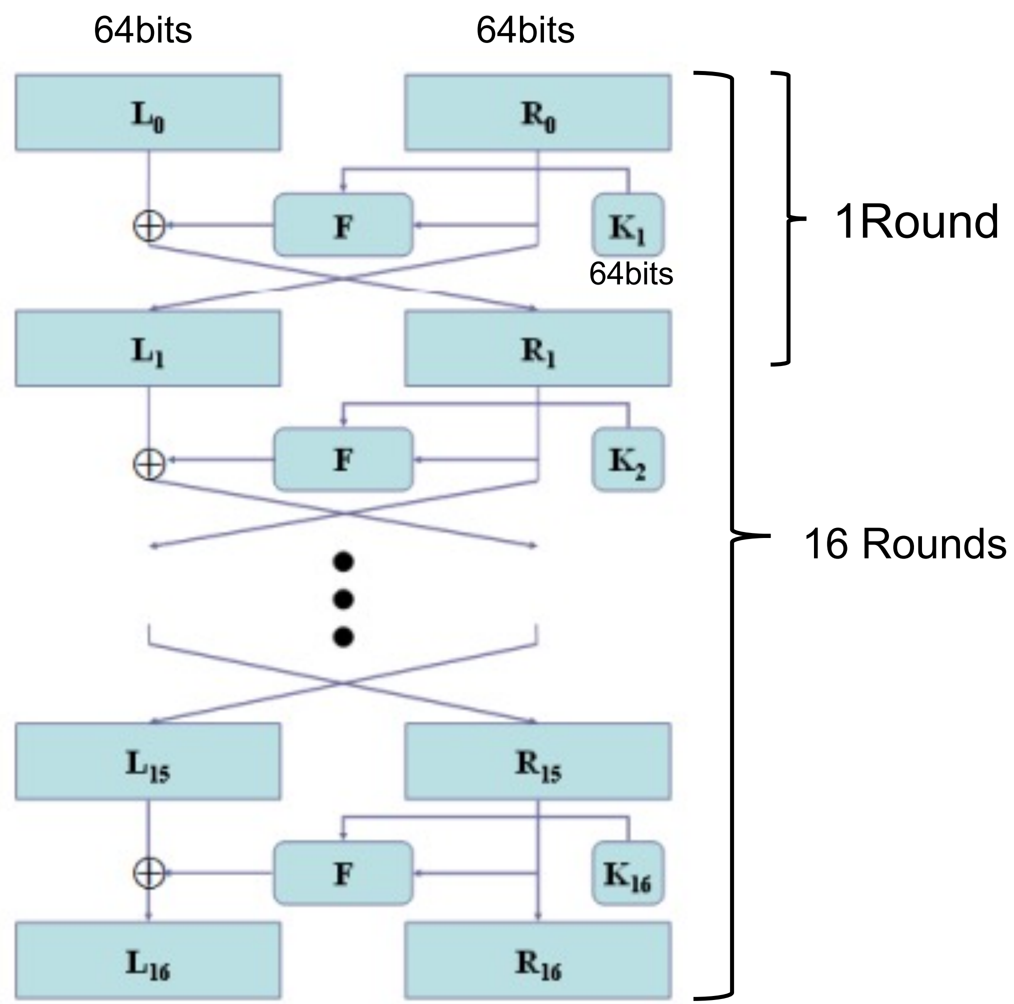

```python
def KeySched0(eng,A,C,KC,ancilla):
    c = eng.allocate_qubit()
    C2 = eng.allocate_qureg(32)
    copy(eng,C,C2,32)
    CDKM(eng,A,C,c,32)
    CDKM_minus(eng,KC,C,c,32)
    GFunction(eng,C,ancilla)
    return C, C2
```

```python
def KeySched1(eng,B,D,KC,ancilla):
    c = eng.allocate_qubit()
    D2 = eng.allocate_qureg(32)
    copy(eng,D,D2,32)
    CDKM_minus(eng,B,D,c,32)
    CDKM(eng,KC,D,c,32)
    GFunction(eng, D,ancilla)
    return D,D2
```

SEED - encrypt

```python
for i in range(16):
    Round_constant_XOR(eng, KC_q, KC[i], 32)
    Round_constant_XOR(eng, KC_q2, KC[i], 32)
    key0, C = KeySched0(eng, A, C, KC_q, ancilla)
    key1, D = KeySched1(eng, B, D, KC_q2, ancilla1)
    Round_constant_XOR(eng, KC_q, KC[i], 32)      #reverse, KC_q =0
    Round_constant_XOR(eng, KC_q2, KC[i], 32)
    L0,L1,R0,R1 = encrypt(eng, L0, L1, R0, R1, key0, key1, ancilla)

    if(i%2 ==0):
        A,B=  RightShift(eng, A, B)
    else :
        C,D = LeftShift(eng, C, D)
```

```python
def encrypt(eng,L0,L1,R0,R1,key0,key1,ancilla):
    FFunction(eng,R0,R1,key0,key1,ancilla)
    for i in range(32):
        CNOT | (key0[i],L0[i])
        CNOT | (key1[i],L1[i])

    return R0,R1,L0,L1
```

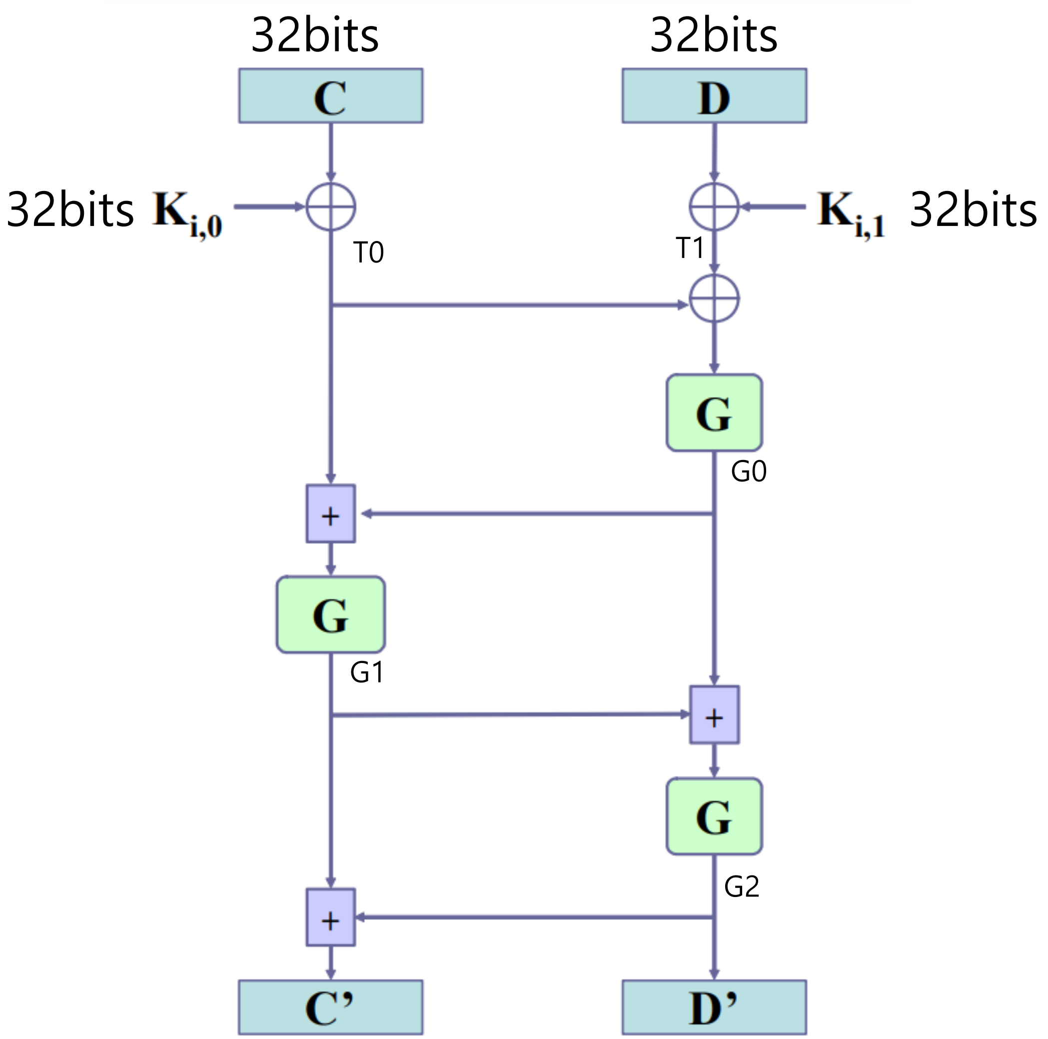


SEED - F Function

- **F Function**

  R_i 과 K(= $K_{i,0}$, $K_{i,1}$)를 입력으로 받음
  R_i 는 다시 32비트 C,D로 나뉨.
  XOR, Addition, G Function으로 이루어짐.
  C,D는 각각 $K_{i,0}$ 과 $K_{i,1}$ 과 XOR 연산

```
def FFunction(eng,R0,R1,key0,key1,ancilla):
    c = eng.allocate_qubit()

    CNOT32(eng,R0,key0)
    CNOT32(eng,R1,key1)
    CNOT32(eng,key0,key1)
    GFunction(eng,key1,ancilla)
    CDKM(eng,key1,key0,c,32)
    GFunction(eng, key0,ancilla)
    CDKM(eng,key0,key1,c,32)
    GFunction(eng, key1, ancilla)
    CDKM(eng, key1, key0, c, 32)
```

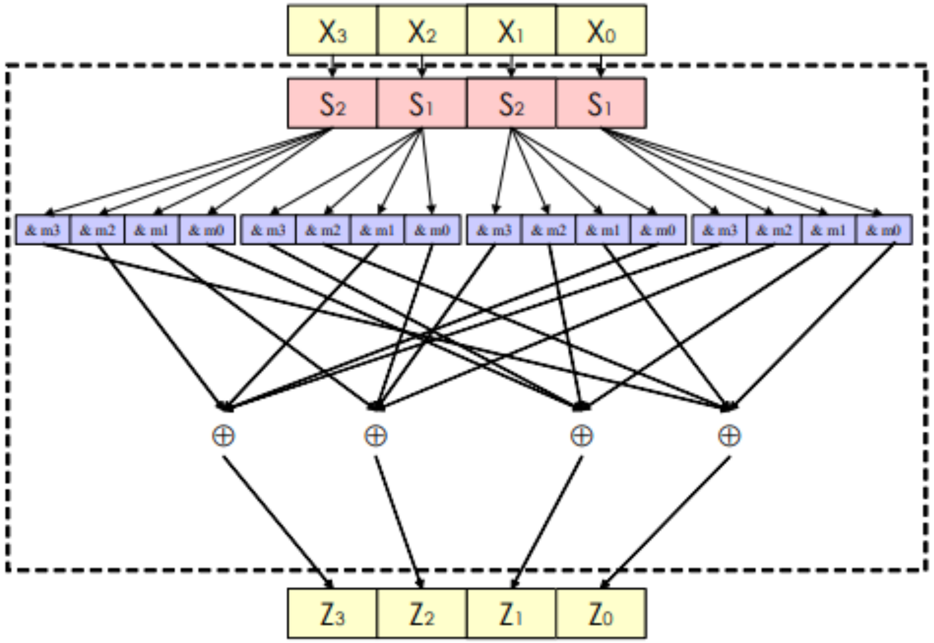


SEED - G Function

- **G Function**

  2개의 Sbox 사용

  - Sbox는 입력으로 8비트값을 받음
  - 상수 m0,m1,m2,m3와 &연산



$$Z = SS_3(X_3) \oplus SS_2(X_2) \oplus SS_1(X_1) \oplus SS_0(X_0)$$

$$Y_3 = S_2(X_3),\ \ Y_2 = S_1(X_2),\ \ Y_1 = S_2(X_1),\ \ Y_0 = S_1(X_0),$$

$$\begin{aligned}
Z_3 &= (Y_0 \& m_3) \oplus (Y_1 \& m_0) \oplus (Y_2 \& m_1) \oplus (Y_3 \& m_2) \\
Z_2 &= (Y_0 \& m_2) \oplus (Y_1 \& m_3) \oplus (Y_2 \& m_0) \oplus (Y_3 \& m_1) \\
Z_1 &= (Y_0 \& m_1) \oplus (Y_1 \& m_2) \oplus (Y_2 \& m_3) \oplus (Y_3 \& m_0) \\
Z_0 &= (Y_0 \& m_0) \oplus (Y_1 \& m_1) \oplus (Y_2 \& m_2) \oplus (Y_3 \& m_3)
\end{aligned}$$

$$(m_0 = 0\text{xfc},\ \ m_1 = 0\text{xf3},\ \ m_2 = 0\text{xcf},\ \ m_3 = 0\text{x3f})$$

```
def GFunction(eng,x,ancilla):
    c = eng.allocate_qureg(8)
    c1 = eng.allocate_qureg(8)
    c2 = eng.allocate_qureg(8)
    c3 = eng.allocate_qureg(8)
    Y0=[]
    Y1=[]
    Y2=[]
    Y3=[]
    Z0=[]
    Y0 = Sbox1(eng, x[0:8],8,ancilla[0:38]) # Y0
    Y1 = Sbox2(eng, x[8:16], 8,ancilla[38:76])  # Y1
    Y2 = Sbox1(eng, x[16:24], 8,ancilla[76:114])
    Y3 = Sbox2(eng, x[24:32], 8,ancilla[114:152])
```

SEED - Sbox

- **Sbox 구현**

양자 컴퓨터에서는 Look-up table사용 불가 -> 특정 상태로 결정 지을 수 없기 때문

$$S_i : Z_{2^8} \rightarrow Z_{2^8},$$
$$S_1(x) = A^{(1)} \cdot x^{247} \oplus 169 \quad S_2(x) = A^{(2)} \cdot x^{251} \oplus 56$$

```
def Sbox1(eng,x,n,ancilla):

    x= inversion(eng,x,ancilla)

    x = Squaring(eng, x, n)
    x = Squaring(eng, x, n)
    x = Squaring(eng, x, n)
```

GF(2^8)에서 계산

SEED GF(2^8)의 기약다항식 $p(x) = x^8 + x^6 + x^5 + x + 1$

$$x^{-1} \equiv x^{254} mod\ p(x)$$
$$(x^{-1})^8 \equiv x^{247} mod\ p(x)$$
$$(x^{-1})^4 \equiv x^{251} mod\ p(x)$$

$$x^{-1} = x^{254} = \left((a \cdot a^2) \cdot (a \cdot a^2)^4 \cdot (a \cdot a^2)^{16} \cdot a^{64}\right)^2$$

$$x^{247} = \left(\left((a \cdot a^2) \cdot (a \cdot a^2)^4 \cdot (a \cdot a^2)^{16} \cdot a^{64}\right)^2\right)^{2 \cdot 2 \cdot 2}$$

$$x^{251} = \left(\left((a \cdot a^2) \cdot (a \cdot a^2)^4 \cdot (a \cdot a^2)^{16} \cdot a^{64}\right)^2\right)^{2 \cdot 2}$$

$$A^{(1)} = \begin{array}{c c} & \begin{array}{cccccccc} x_7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x_0 \end{array} \\ \begin{array}{c} x_7 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_0 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}, \quad A^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

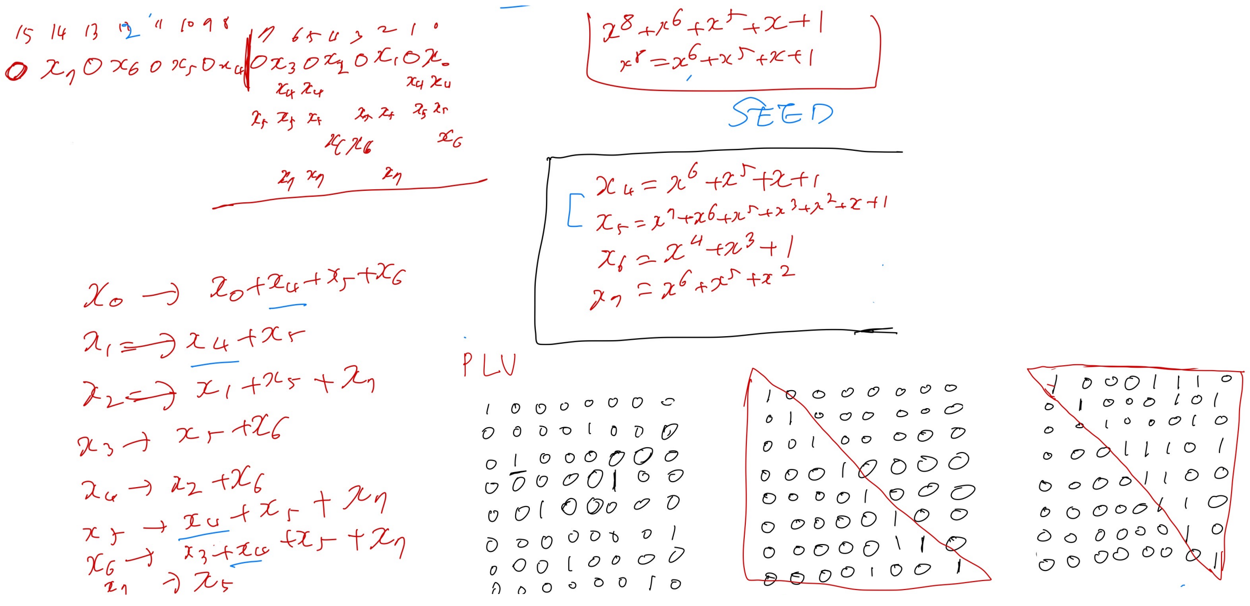
SEED - Sbox

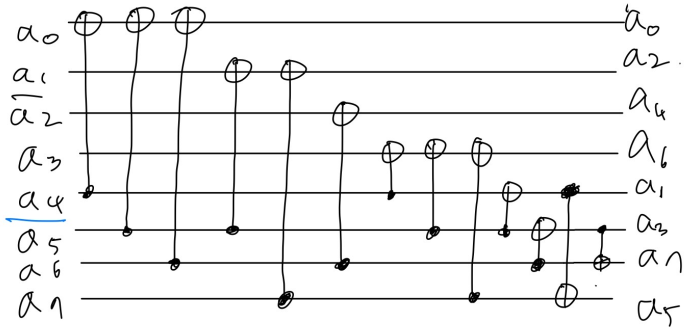
- **Sbox 구현**

$$x^{-1} = x^{254} = \left((a \cdot a^2) \cdot (a \cdot a^2)^4 \cdot (a \cdot a^2)^{16} \cdot a^{64}\right)^2$$

```python
def inversion(eng,a,ancilla):
    # x = (a*a^2)*(a^64)*((a*a^2)^4)*((a*a^2)^16)^2
    n = 8
    a1 = eng.allocate_qureg(n)

    count = 0

    copy(eng, a, a1, n) # a1 = a
    a1 = Squaring(eng, a1, n) # a1 = a^2

    a2 = []
    a2, count, ancilla = recursive_karatsuba(eng, a, a1, n, count, ancilla) #(a*a^2)
    a2 = Reduction(eng, a2)

    # (a * a^2)* a^64
    a = Squaring(eng, a, n)
    a = Squaring(eng, a, n)
    a = Squaring(eng, a, n)
    a = Squaring(eng, a, n)
    a = Squaring(eng, a, n)
    a = Squaring(eng, a, n) #a^64
```

```python
    # (a*a^2)*(a^64)*((a*a^2)^4)*((a*a^2)^16)^2
    a5 = Squaring(eng, a5, n)

    return a5
```



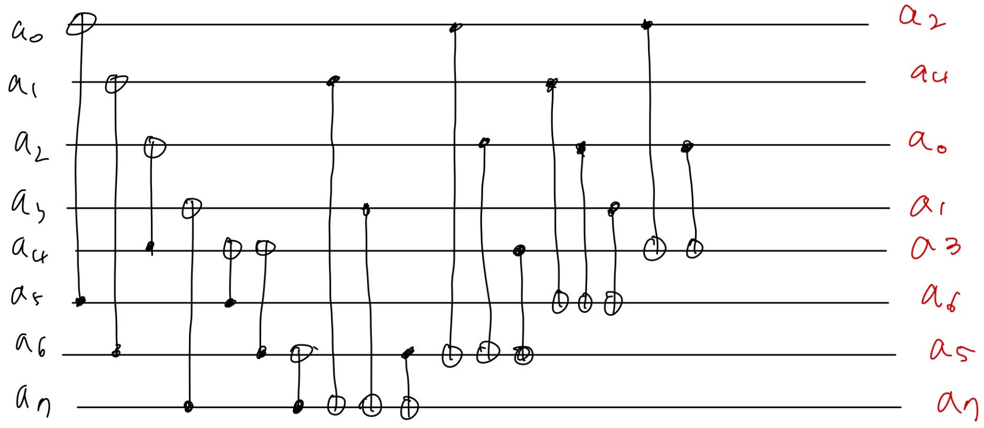


SEED - Sbox

- **Sbox 구현**

$$S_i : Z_{2^8} \rightarrow Z_{2^8} ,$$
$$S_1(x) = A^{(1)} \cdot x^{247} \oplus 169 \quad S_2(x) = A^{(2)} \cdot x^{251} \oplus 56$$

$$A^{(1)} = \begin{array}{c} \\ x_7 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_0 \end{array} \begin{array}{c} x_7 \cdots\cdots x_0 \\ \begin{pmatrix} 1&0&0&0&1&0&1&0 \\ 1&1&1&1&1&1&1&0 \\ 1&0&0&0&0&1&0&1 \\ 0&1&0&0&0&0&1&0 \\ 0&1&0&0&0&1&0&1 \\ 0&0&1&0&0&0&0&1 \\ 1&0&0&0&1&0&0&0 \\ 0&0&0&1&0&1&0&0 \end{pmatrix} \end{array} \Rightarrow \begin{array}{c} \\ x_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_7 \end{array} \begin{array}{c} x_0 \cdots\cdots x_7 \\ \begin{bmatrix} 0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&0&0&1 \\ 1&0&0&0&0&1&0&0 \\ 1&0&1&0&0&0&1&0 \\ 0&1&0&0&0&0&1&0 \\ 1&0&1&0&0&0&0&1 \\ 0&1&1&1&1&1&1&1 \\ 0&1&0&1&0&0&0&1 \end{bmatrix} \end{array} = \begin{array}{c} \text{Permutation} \\ \begin{bmatrix} 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \end{array} \begin{array}{c} \text{Upper} \\ \begin{bmatrix} 1&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&1&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \end{array} \begin{array}{c} \text{Lower} \\ \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&1&0&1&0&0&0 \\ 0&1&1&1&0&1&0&0 \\ 1&0&1&0&1&0&1&0 \\ 0&1&0&1&0&0&1&1 \end{bmatrix} \end{array}$$



SEED - 결과

```
Key        : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plaintext  : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Encryption....
Ciphertext : 5E BA C6 E0 05 4E 16 68 19 AF F1 CC 6D 34 6C DB
```

```
Generate Ciphertext...
Result : 11100000110001101011101001011110

Result : 01101000000101100100111000000101

Result : 11001100111100011010111100011001

Result : 11011011011011000011010001101101
```

0xE0C6BA5E. $_{(16)}$ = 5E BA C6 E0

0x68164E05 $_{(16)}$ = 05 4E 16 68

0xCCF1AF19 $_{(16)}$ = 19 AF F1 CC

0xDB6C346D $_{(16)}$ = 6D 34 6C DB

Q & A