PIPO 양자 구현

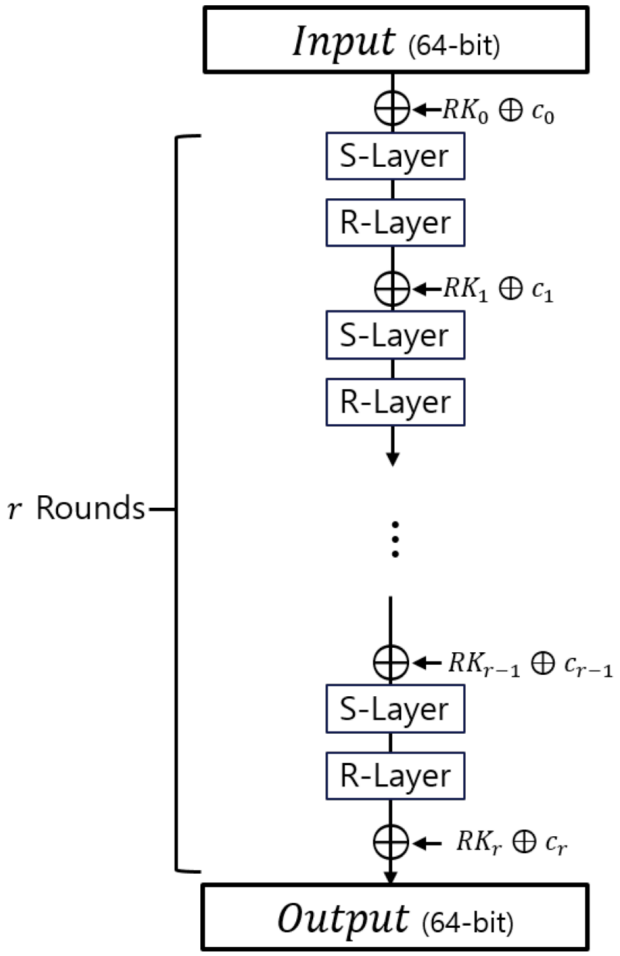
장경배

https://youtu.be/cEPS92_vaeo

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

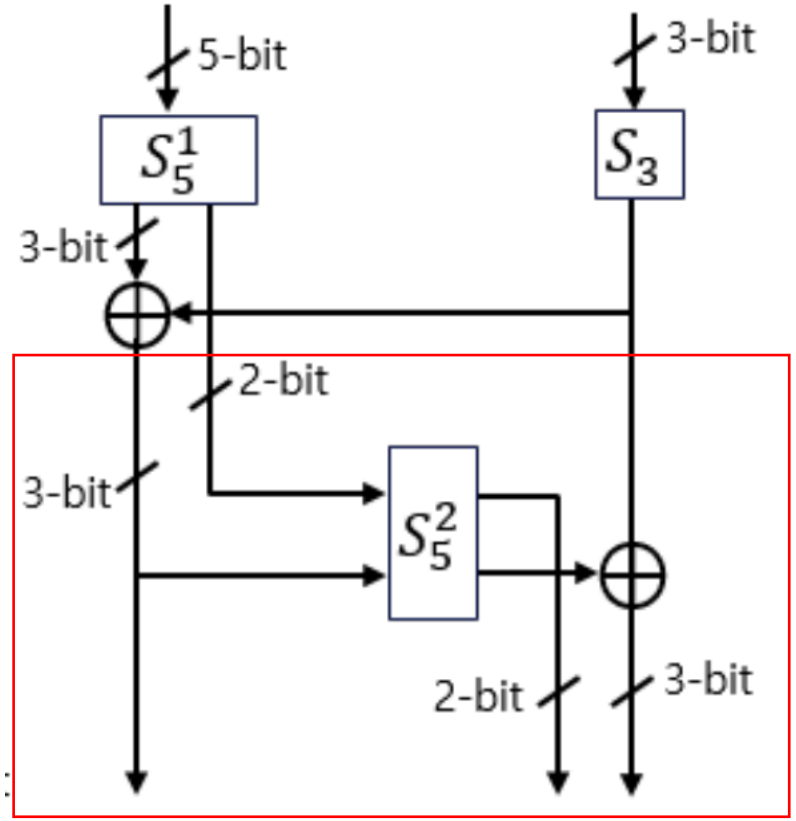
PIPO



PIPO 암호화 구조

- 64-bit 평문, (128, 256)-bit 키
- (13, 17) 라운드
- 라운드 함수
  - AddRoundkey, Sbox, Permutation
- 키 스케줄
  - 라운드 Constant XOR

PIPO Sbox



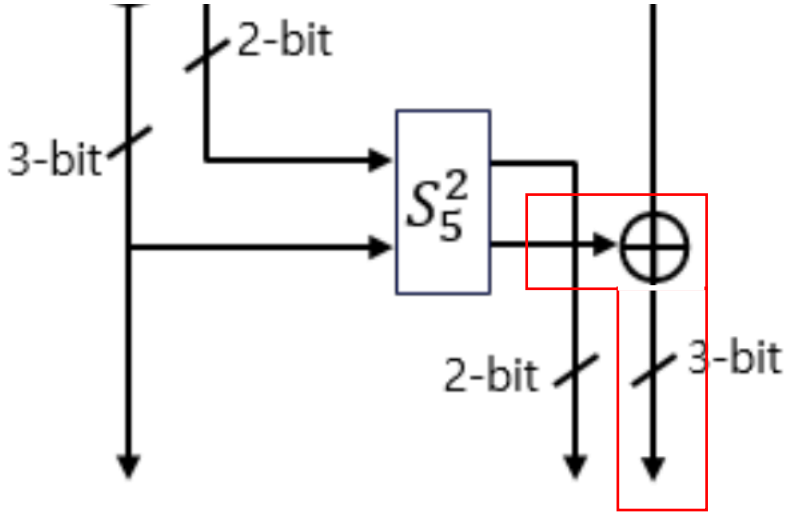
(D) Unbalanced-Bridge

- 문제점

  추가 큐빗이 사용 되어야 하는데 심지어

  매 라운드마다 계속 할당되어야 함

PIPO Sbox 최적화 (1/7)



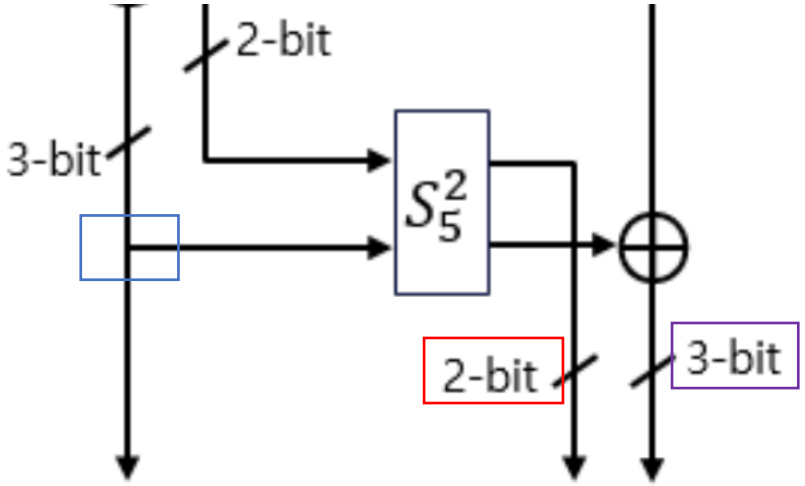
1. $S_5{}^2$ 를 수행하고 3-bit 결과 값을 XOR 해준 뒤, Reverse 연산

→ 결과값(3-bit)은 활용만 되고 유지 될 필요 없으니 괜찮음

→ 남은 건 입력되는 3-bit 와 결과 값의 2-bit

유지되어야 함 변경됨

PIPO Sbox 최적화 (2/7)



3-bit 는 변경되었다면

2-bit 중 (x[5]) 를 유지할 수 없음

2. 남은 건 $S_5{}^2$ 에 입력되는 3-bit 와 결과 값의 2-bit

→ 3-bit 는 유지 되어야함

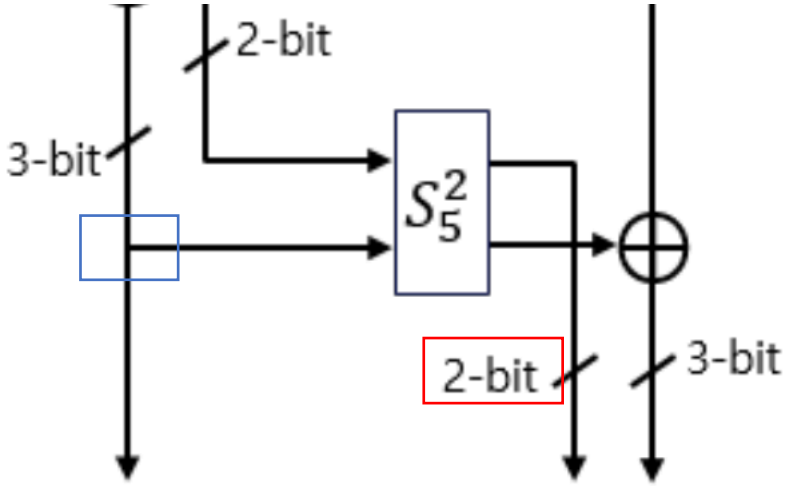
→ 2-bit는 변경 됨

* 앞 단계의 3-bit 에서는 고려 사항이 아니였음

```
889  //S5_2
890  t[0] = x[7]; t[1] = x[3]; t[2] = x[4];
891  x[6] ^= (t[0] & x[5]);
892  t[0] ^= x[6];
893  x[6] ^= (t[2] | t[1]);
894  t[1] ^= x[5];
895  x[5] ^= (x[6] | t[2]);
896  t[2] ^= (t[1] & t[0]);
897  // truncate XOR and swap
898  x[2] ^= t[0]; t[0] = x[1] ^ t[2]; x[1] = x[0]^t[1];
899  x[0] = x[7]; x[7] = t[0];
900  t[1] = x[3]; x[3] = x[6]; x[6] = t[1];
901  t[2] = x[4]; x[4] = x[5]; x[5] = t[2];
902  // Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
903
```

← 때문

PIPO Sbox 최적화 (3/7)



2-bit (x[5], x[6]) 는 변경되었지만

3-bit 를 (x[7], x[3] ,x[4]) 를 유지할 수 있음

2. 남은 건 $S_5{}^2$ 에 입력되는 3-bit 와 결과 값의 2-bit

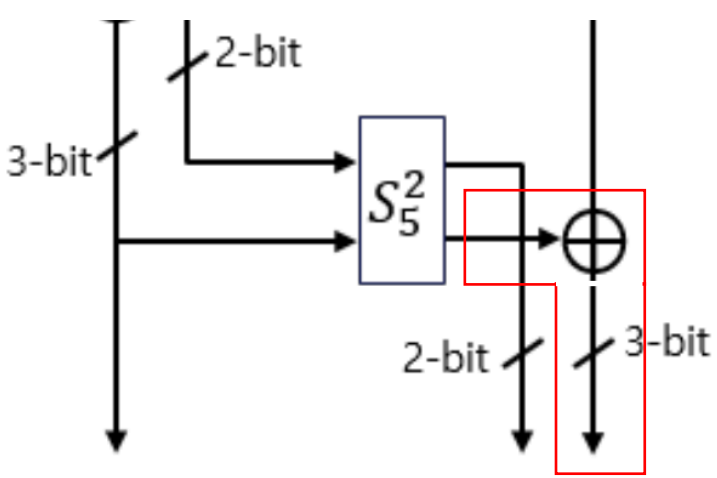
→ 3-bit 는 유지 되어야함

* 앞 단계의 3-bit 에서는 고려 사항이 아니였음

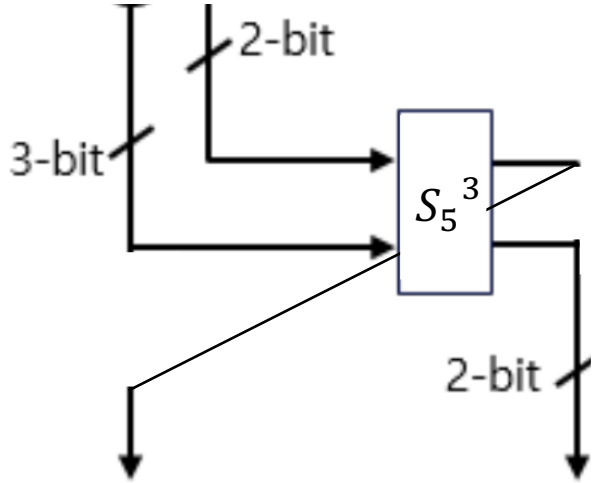
→ 2-bit는 변경 됨

```
889 //S5_2
890 t[0] = x[7]; t[1] = x[3]; t[2] = x[4];
891 x[6] ^= (t[0] & x[5]);
892 t[0] ^= x[6];
893 x[6] ^= (t[2] | t[1]);
894 t[1] ^= x[5];
895 x[5] ^= (x[6] | t[2]);
896 t[2] ^= (t[1] & t[0]);
897 // truncate XOR and swap
898 x[2] ^= t[0]; t[0] = x[1] ^ t[2]; x[1] = x[0]^t[1];
899 x[0] = x[7]; x[7] = t[0];
900 t[1] = x[3]; x[3] = x[6]; x[6] = t[1];
901 t[2] = x[4]; x[4] = x[5]; x[5] = t[2];
902 // Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
903
```

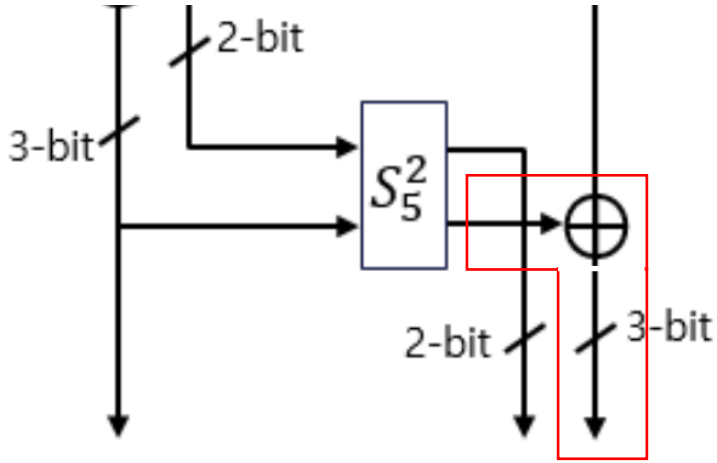
PIPO Sbox 최적화 (4/7)



Reverse



PIPO Sbox 최적화 (5/7)

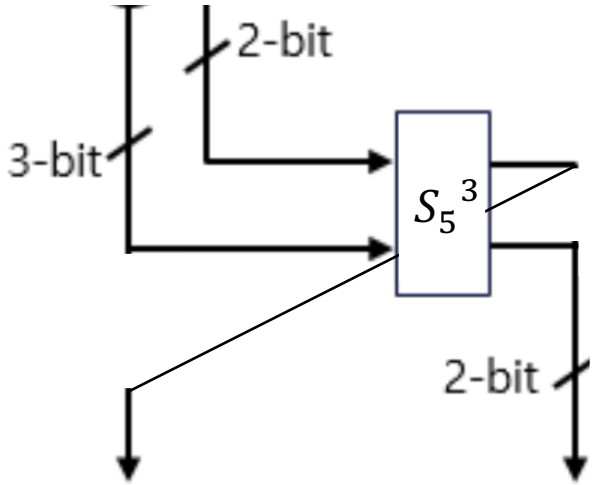



```
889  //S5_2
890  t[0] = x[7]; t[1] = x[3]; t[2] = x[4];
891  x[6] ^= (t[0] & x[5]);
892  t[0] ^= x[6];
893  x[6] ^= (t[2] | t[1]);
894  t[1] ^= x[5];
895  x[5] ^= (x[6] | t[2]);
896  t[2] ^= (t[1] & t[0]);
897  // truncate XOR and swap
898  x[2] ^= t[0]; t[0] = x[1] ^ t[2]; x[1] = x[0]^t[1];
899  x[0] = x[7]; x[7] = t[0];
900  t[1] = x[3]; x[3] = x[6]; x[6] = t[1];
901  t[2] = x[4]; x[4] = x[5]; x[5] = t[2];
902  // Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
903
```

```
def Sbox5_2(eng, x):

    Toffoli | (x[4], x[2], x[3])
    CNOT | (x[3], x[4])

    CNOT | (x[2], x[0])

    Toffoli | (x[0], x[4], x[1])

    # 3-bit : x0, x1, x4 (XOR to 3-bit)
    # 2-bit : x2, x3     (store)
```

t0, t1, t2 가 3-bit

PIPO Sbox 최적화 (6/7)

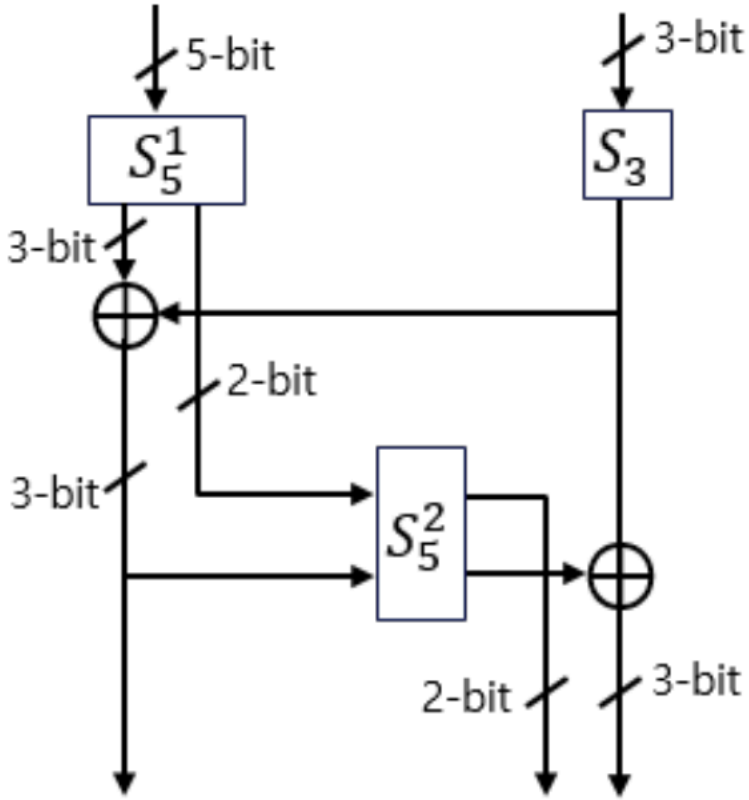



```
889  //S5_2
890  t[0] = x[7]; t[1] = x[3]; t[2] = x[4];
891  x[6] ^= (t[0] & x[5]);
892  t[0] ^= x[6];
893  x[6] ^= (t[2] | t[1]);
894  t[1] ^= x[5];
895  x[5] ^= (x[6] | t[2]);
896  t[2] ^= (t[1] & t[0]);
897  // truncate XOR and swap
898  x[2] ^= t[0]; t[0] = x[1] ^ t[2]; x[1] = x[0]^t[1];
899  x[0] = x[7]; x[7] = t[0];
900  t[1] = x[3]; x[3] = x[6]; x[6] = t[1];
901  t[2] = x[4]; x[4] = x[5]; x[5] = t[2];
902  // Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
903
```

```
def Sbox5_2_new(eng, x):

    Toffoli | (x[4], x[2], x[3])

    X | x[1]
    X | x[0]
    Toffoli | (x[1], x[0], x[3])
    X | x[3]
    X | x[1]
    X | x[0]

    X | x[3]
    X | x[1]
    Toffoli | (x[3], x[1], x[2])
    X | x[2]
    X | x[3]
    X | x[1]
```

x[5], x[6] 이 2-bit

PIPO Sbox 최적화 (7/7)




```
def New_Sbox(eng, x):
    Sbox5_1(eng, x[3:8])
    Sbox3(eng, x[0:3])

    #Extend XOR
    CNOT | (x[1], x[7])
    CNOT | (x[2], x[3])
    CNOT | (x[0], x[4])

    with Compute(eng):
        Sbox5_2(eng, x[3:8]) # t0,1,2 --> x7, 3, 4

    CNOT | (x[7], x[2]) #x[2]^t[0]
    CNOT | (x[4], x[1]) #x[1]^t[2]
    CNOT | (x[3] ,x[0]) #x[0]^t[1]

    Uncompute(eng)
    Sbox5_2_new(eng, x[3:8])

    Swap | (x[7], x[0]) #New x[0](x[7]), x[7] = x[0]^t[1]
    Swap | (x[7], x[1]) #New x[7](x[1]^t[2], x[1] = x[0]^t[1]
    Swap | (x[3], x[6])
    Swap | (x[4], x[5])
```

R-layer, S-layer , AddRoundkey

```
def R_layer(eng, x):

    for j in range(1):  # Rotate X[1] right by 1
        for i in range(7):
            Swap | (x[i+8], x[i + 9])

    for j in range(4):  # Rotate X[2] right by 4
        for i in range(7):
            Swap | (x[i+16], x[i + 17])

    for j in range(5):  # Rotate X[3] right by 5
        for i in range(7):
            Swap | (x[i+24], x[i + 25])

    for j in range(2):  # Rotate X[4] right by 2
        for i in range(7):
            Swap | (x[i+32], x[i + 33])

    for j in range(3):  # Rotate X[5] right by 3
        for i in range(7):
            Swap | (x[i+40], x[i + 41])

    for j in range(7):  # Rotate X[6] right by 7
        for i in range(7):
            Swap | (x[i+48], x[i + 49])

    for j in range(6):  # Rotate X[7] right by 6
        for i in range(7):
            Swap | (x[i+56], x[i + 57])
```

```
def Round(eng, x, k):

    # Sbox
    S0 = [x[0],x[8],x[16],x[24],x[32],x[40],x[48],x[56]]
    S1 = [x[1], x[9], x[17], x[25], x[33], x[41], x[49], x[57]]
    S2 = [x[2], x[10], x[18], x[26], x[34], x[42], x[50], x[58]]
    S3 = [x[3], x[11], x[19], x[27], x[35], x[43], x[51], x[59]]
    S4 = [x[4], x[12], x[20], x[28], x[36], x[44], x[52], x[60]]
    S5 = [x[5], x[13], x[21], x[29], x[37], x[45], x[53], x[61]]
    S6 = [x[6], x[14], x[22], x[30], x[38], x[46], x[54], x[62]]
    S7 = [x[7], x[15], x[23], x[31], x[39], x[47], x[55], x[63]]

    New_Sbox(eng, S0)
    New_Sbox(eng, S1)
    New_Sbox(eng, S2)
    New_Sbox(eng, S3)
    New_Sbox(eng, S4)
    New_Sbox(eng, S5)
    New_Sbox(eng, S6)
    New_Sbox(eng, S7)

    R_layer(eng, x)

    # AddRoundkey
    for i in range(64):
        CNOT | (k[i], x[i])
```

Keyschedlue

```
def PIPO(eng):

    x = eng.allocate_qureg(64)
    k0 = eng.allocate_qureg(64)
    k1 = eng.allocate_qureg(64)

    # Plaintext : 0x098552f6_1e270026
    # Key : 0x6DC416DD_779428D2_7E1D20AD_2E152297
    plaintext_key_init(eng, x, k0, k1)

    # Add whitening key
    for i in range(64):
        CNOT | (k0[i], x[i])

    # Round function (13 rounds),
    # Key generation performed with X gate (Round Constant XOR)

    X | k1[0]
    Round(eng, x, k1) #1
    #X | k1[0]


    X | k0[1]
    Round(eng, x, k0) #2
    X | k0[1]


    #X | k1[0]
    X | k1[1]
    Round(eng, x, k1) #3
    #X | k1[0]
    X | k1[1]

    X | k0[2]
    Round(eng, x, k0) #4
    #X | k0[2]
```

라운드 Constant XOR 하는 것 중 Reverse 연산 부분 생략 가능

Result

- 주간 보고 결과 보다 최적화

PIPO-64/128

```
Gate counts:
    Allocate : 192
    CCX : 1248
    CX : 2248
    Deallocate : 192
    X : 1685

Depth : 248.
```

PIPO-64/256

```
Gate counts:
    Allocate : 320
    CCX : 1632
    CX : 2920
    Deallocate : 320
    X : 2202

Depth : 324.
```

감사합니다