PIPO 구현 코드 분석

https://youtu.be/Q6R0wn9Dlts

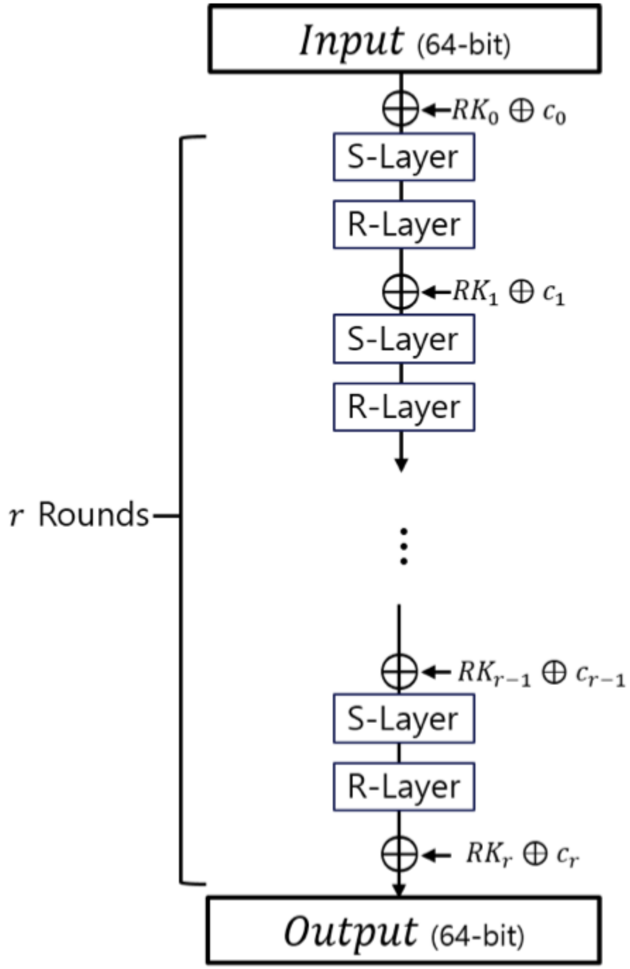
HSU 한성대학교
HANSUNG UNIVERSITY

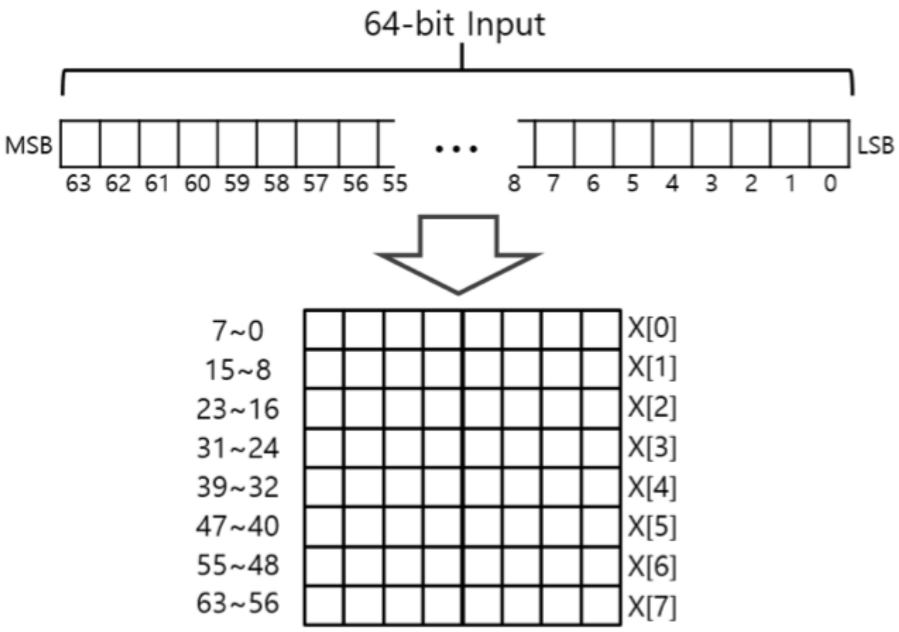
CryptoCraft LAB
https://crypto.modoo.at

PIPO

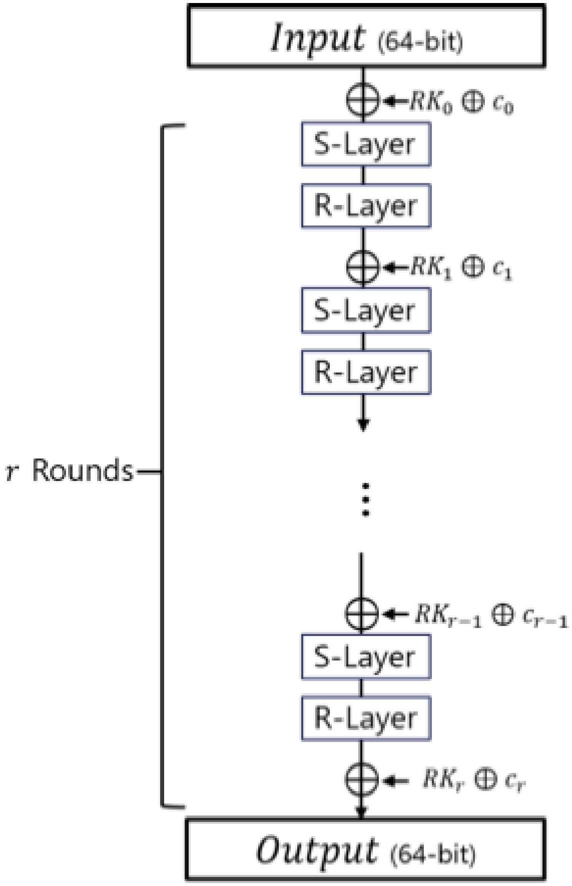
PIPO - 64/128 13ROUND

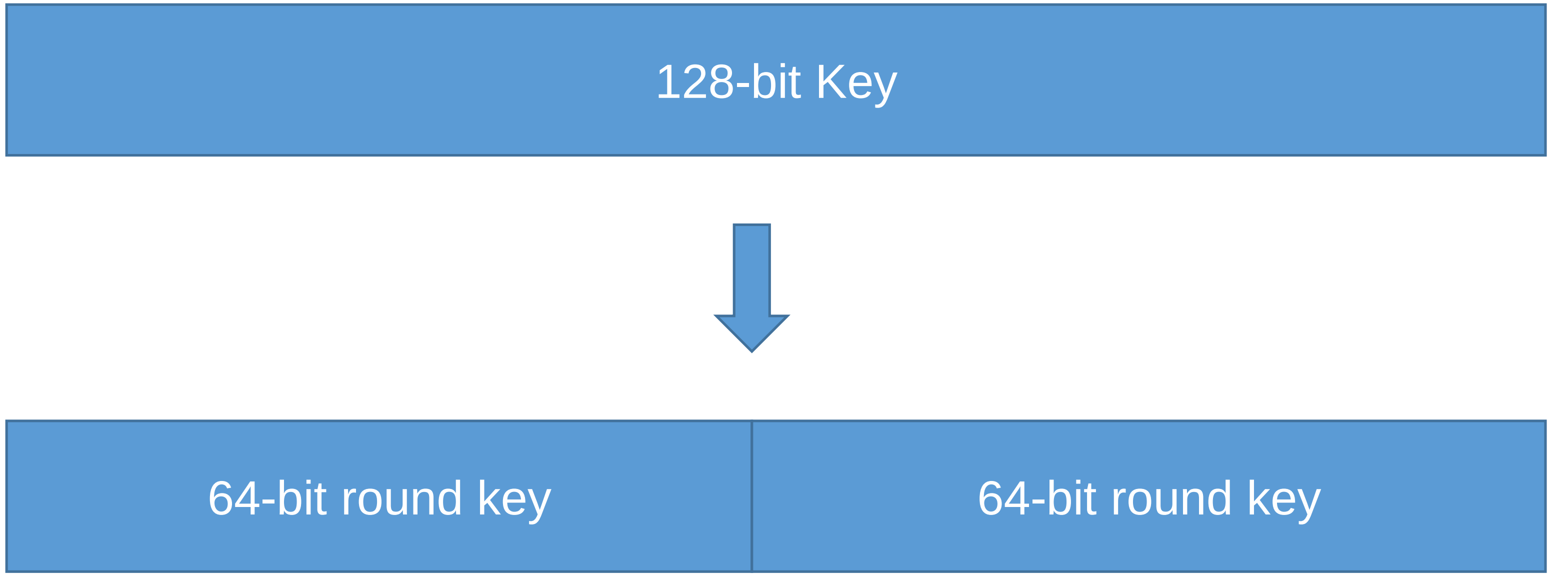
PIPO - 64/256 17ROUND





PIPO 라운드키

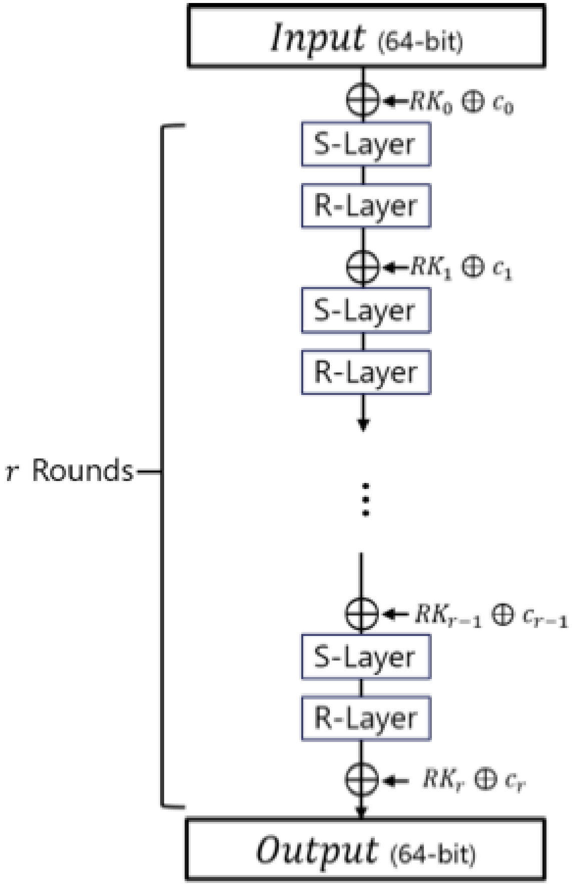


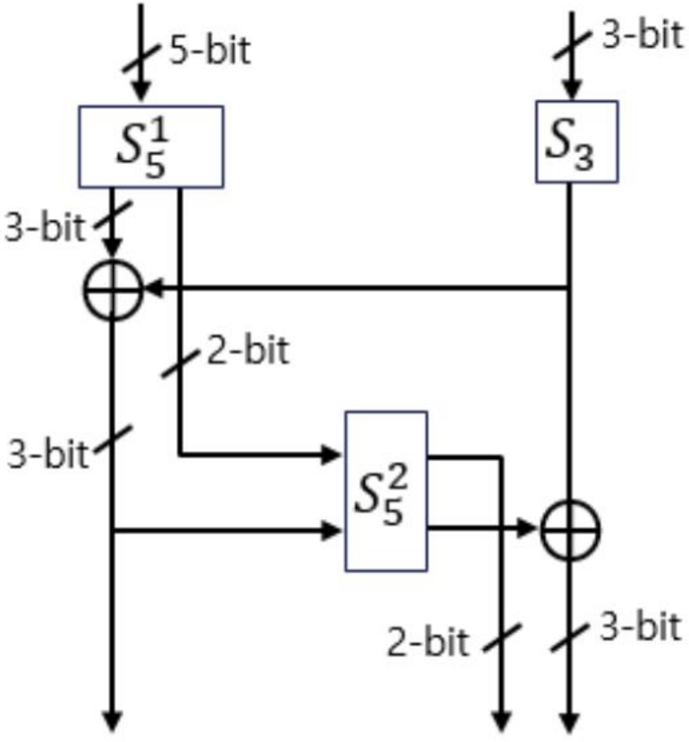

128-bit Key

64-bit round key

64-bit round key

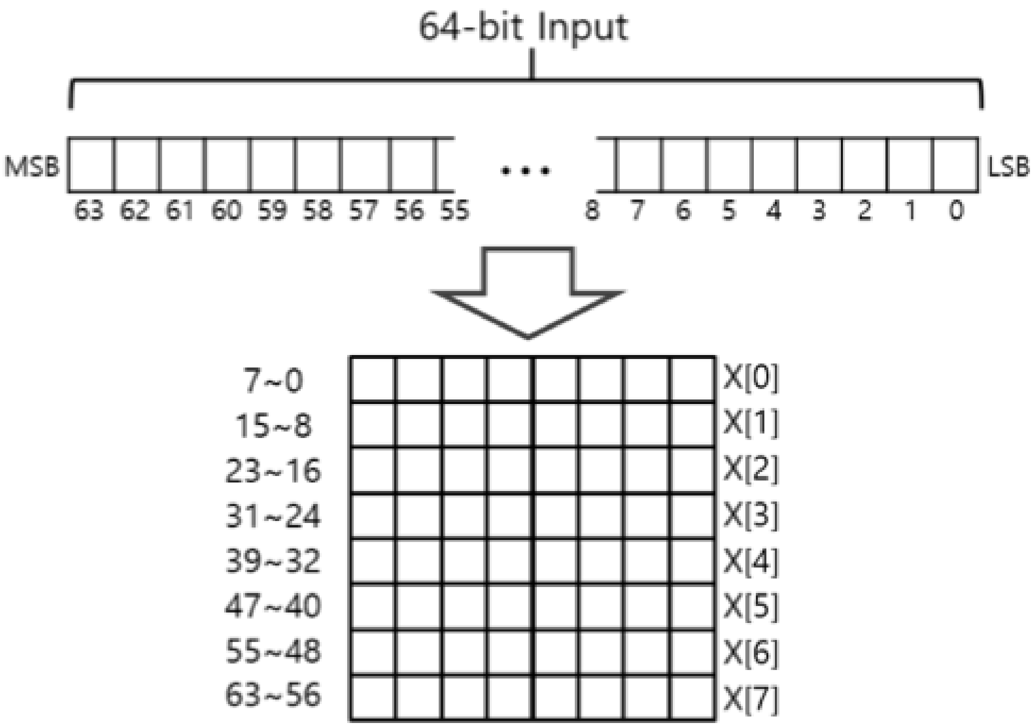
PIPO S-layer





(D) Unbalanced-Bridge

PIPO R-layer



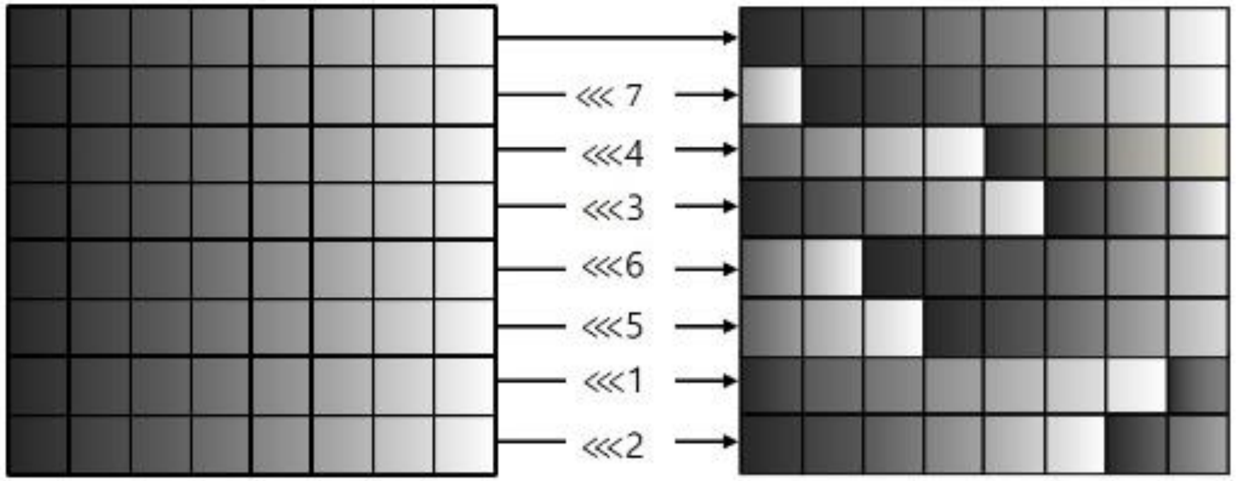
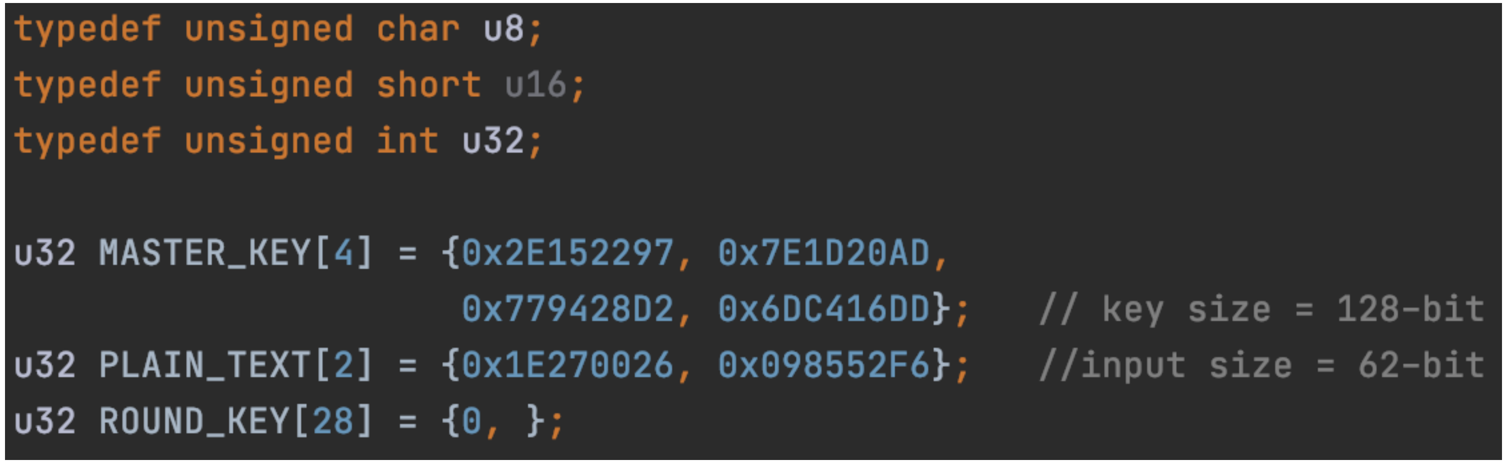


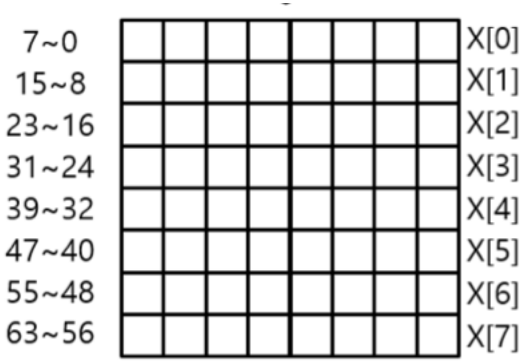
Fig. 3. R-layer

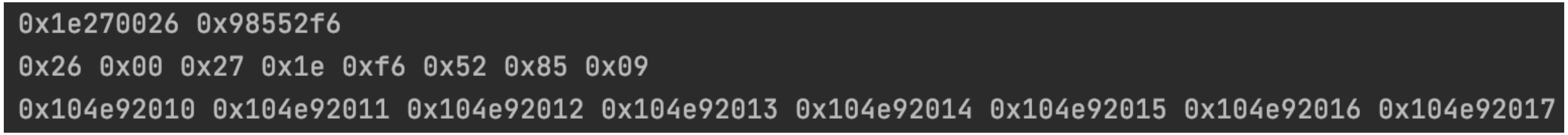
PIPO

- PIPO-64/128
  - Secret key: 0x6DC416DD_779428D2_7E1D20AD_2E152297
  - Plaintext: 0x098552F6_1E270026
  - Ciphertext: 0x6B6B2981_AD5D0327



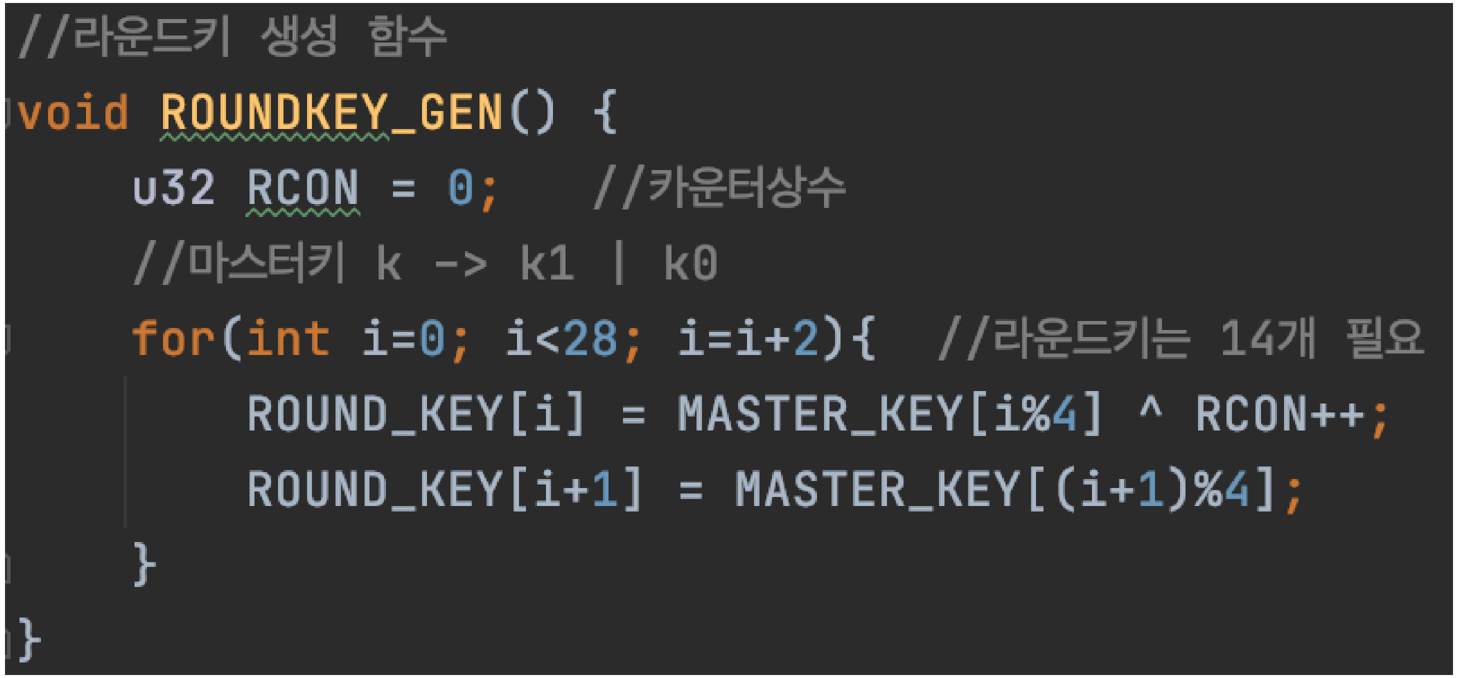
```
typedef unsigned char u8;
typedef unsigned short u16;
typedef unsigned int u32;

u32 MASTER_KEY[4] = {0x2E152297, 0x7E1D20AD,
                     0x779428D2, 0x6DC416DD};   // key size = 128-bit
u32 PLAIN_TEXT[2] = {0x1E270026, 0x098552F6};   //input size = 62-bit
u32 ROUND_KEY[28] = {0, };
```



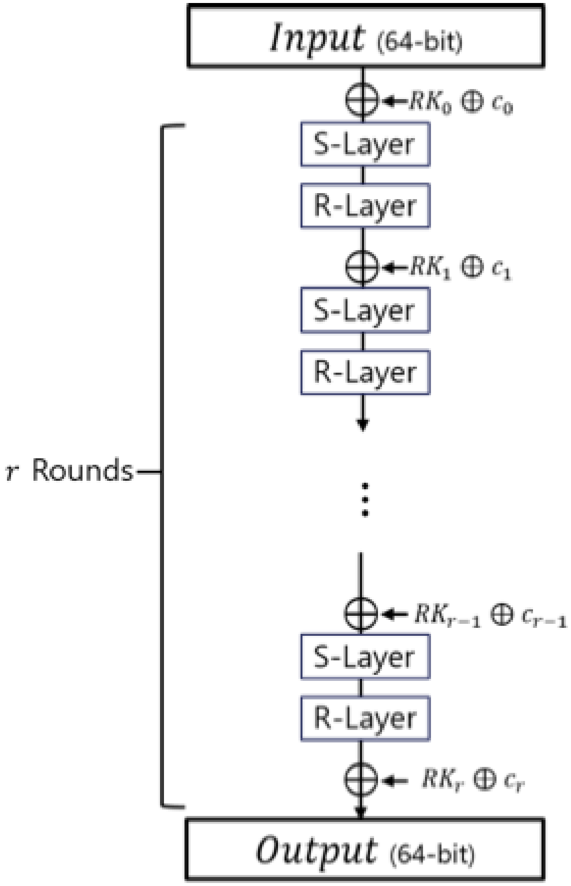


```
0x1e270026 0x98552f6
0x26 0x00 0x27 0x1e 0xf6 0x52 0x85 0x09
0x104e92010 0x104e92011 0x104e92012 0x104e92013 0x104e92014 0x104e92015 0x104e92016 0x104e92017
```

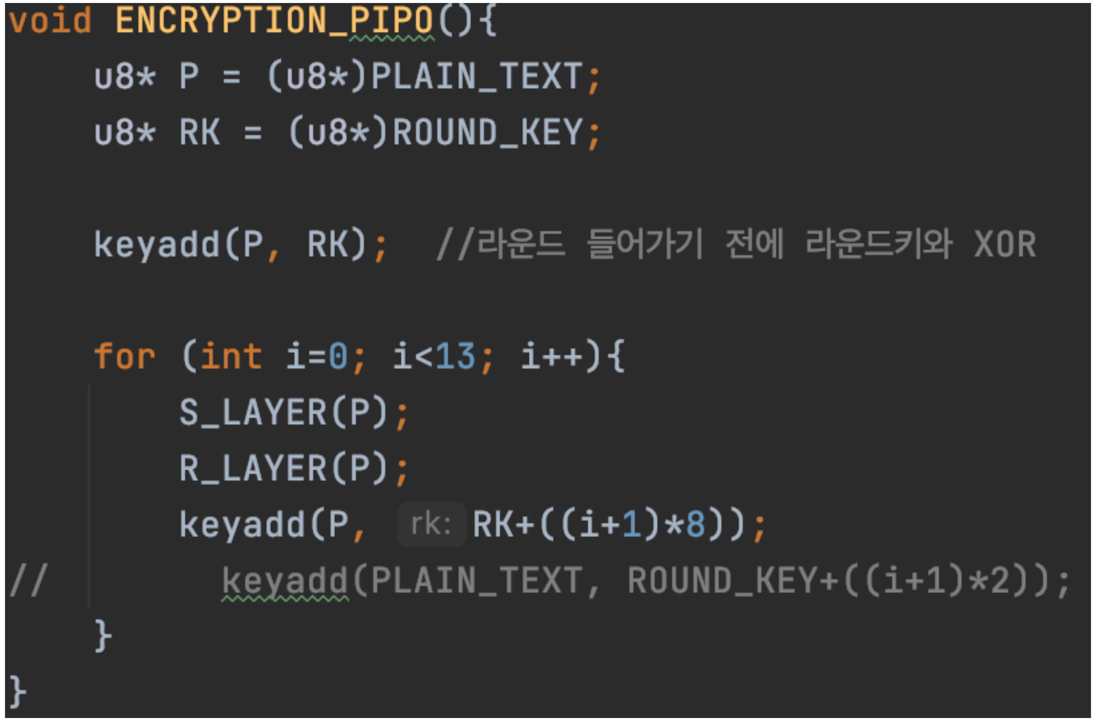
PIPO 라운드키 생성 함수



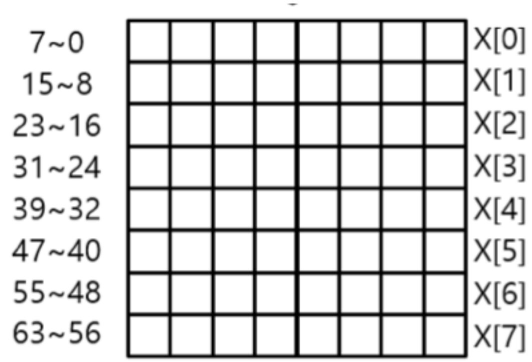
```
//라운드키 생성 함수
void ROUNDKEY_GEN() {
    u32 RCON = 0;    //카운터상수
    //마스터키 k -> k1 | k0
    for(int i=0; i<28; i=i+2){  //라운드키는 14개 필요
        ROUND_KEY[i] = MASTER_KEY[i%4] ^ RCON++;
        ROUND_KEY[i+1] = MASTER_KEY[(i+1)%4];
    }
}
```



PIPO encryption 함수



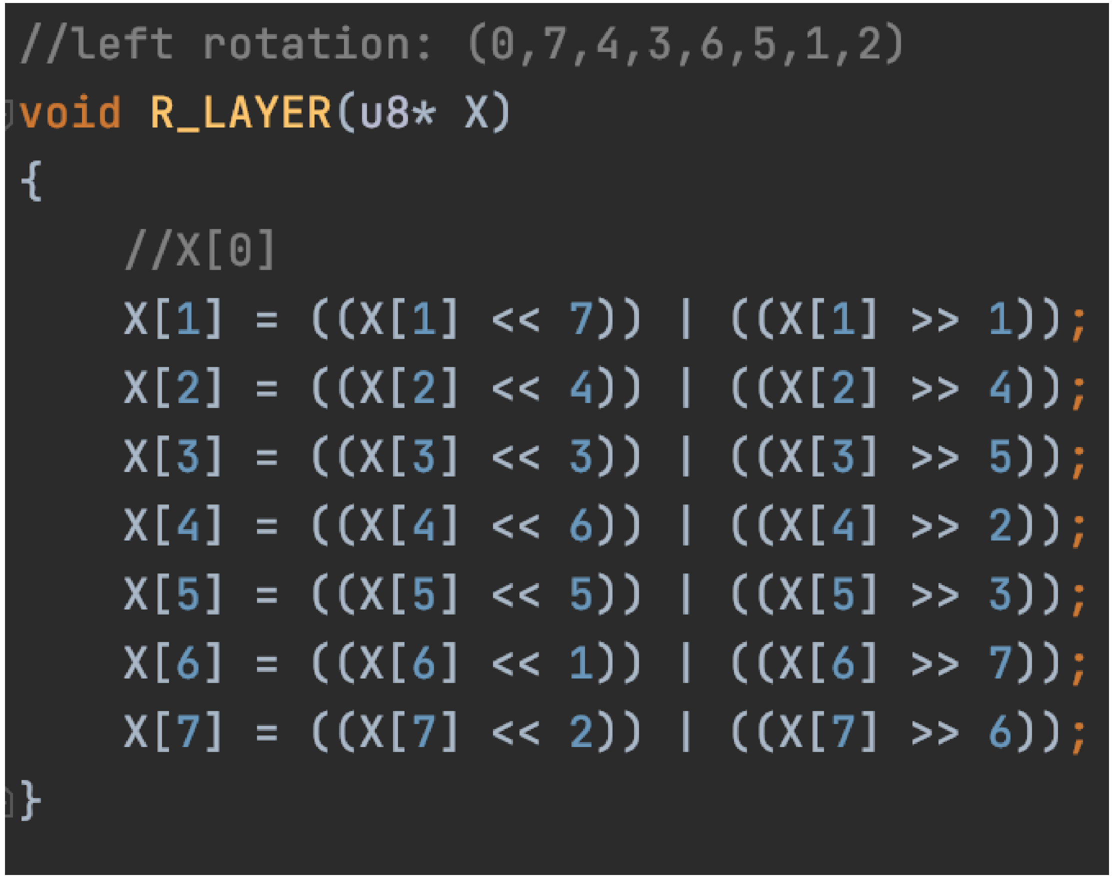
```
void ENCRYPTION_PIPO(){
    u8* P = (u8*)PLAIN_TEXT;
    u8* RK = (u8*)ROUND_KEY;

    keyadd(P, RK);  //라운드 들어가기 전에 라운드키와 XOR

    for (int i=0; i<13; i++){
        S_LAYER(P);
        R_LAYER(P);
        keyadd(P, rk: RK+((i+1)*8));
//          keyadd(PLAIN_TEXT, ROUND_KEY+((i+1)*2));
    }
}
```



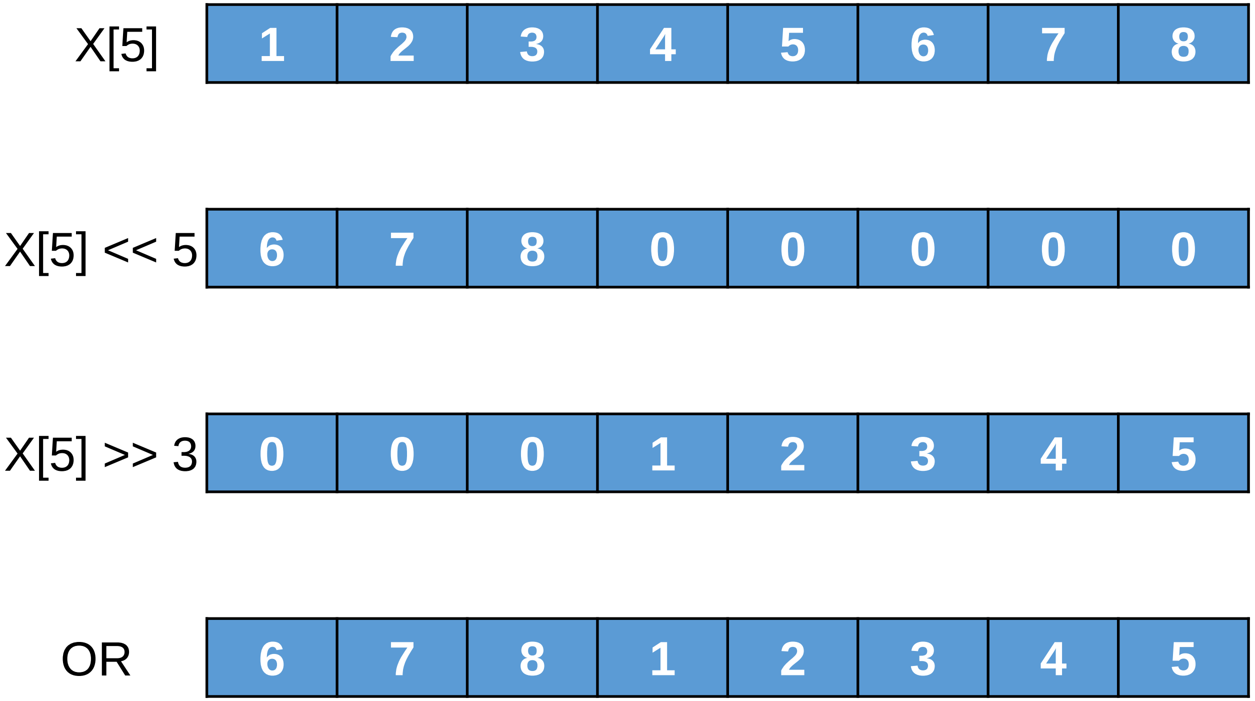
#0	#4
0x12345678	0xa1b2c3d4

#0	#1	#2	#3	#4	#5	#6	#7
0x78	0x56	0x34	0x12	0xd4	0xc3	0xb2	0xa1

PIPO R-Layer

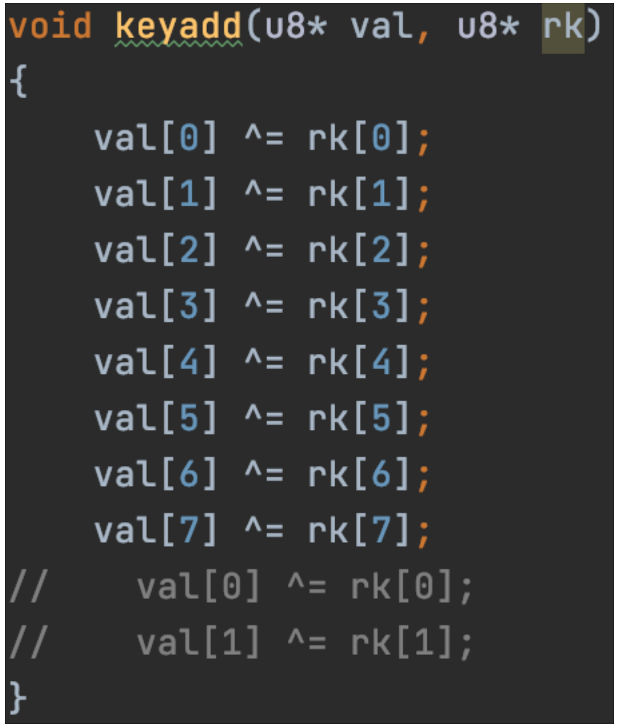


```
//left rotation: (0,7,4,3,6,5,1,2)
void R_LAYER(u8* X)
{
    //X[0]
    X[1] = ((X[1] << 7)) | ((X[1] >> 1));
    X[2] = ((X[2] << 4)) | ((X[2] >> 4));
    X[3] = ((X[3] << 3)) | ((X[3] >> 5));
    X[4] = ((X[4] << 6)) | ((X[4] >> 2));
    X[5] = ((X[5] << 5)) | ((X[5] >> 3));
    X[6] = ((X[6] << 1)) | ((X[6] >> 7));
    X[7] = ((X[7] << 2)) | ((X[7] >> 6));
}
```

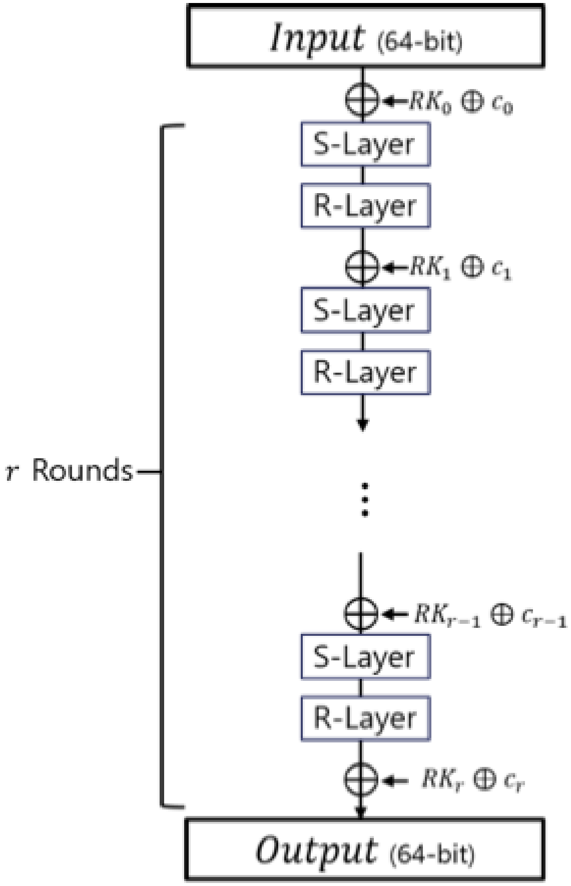


X[5]	1	2	3	4	5	6	7	8
X[5] << 5	6	7	8	0	0	0	0	0
X[5] >> 3	0	0	0	1	2	3	4	5
OR	6	7	8	1	2	3	4	5

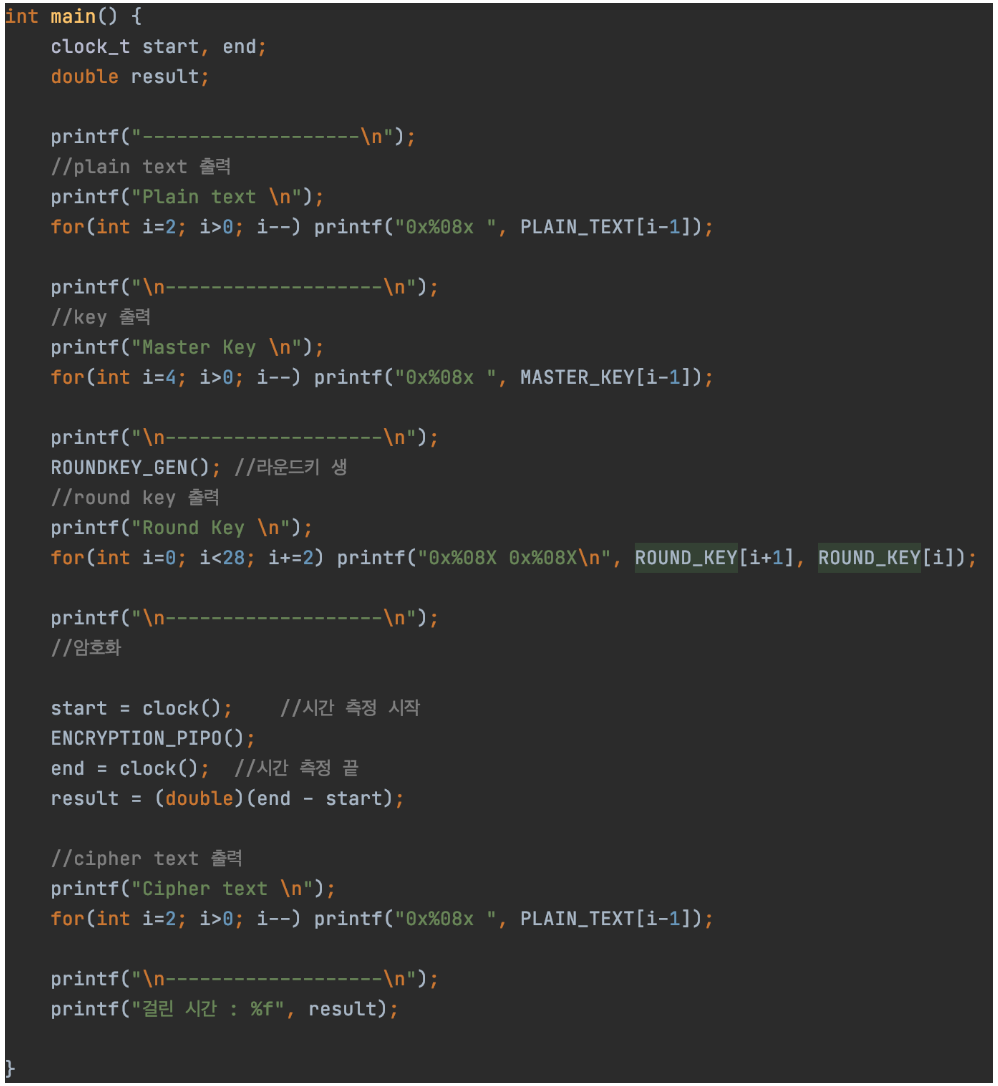
PIPO Keyadd 함수



```
void keyadd(u8* val, u8* rk)
{
    val[0] ^= rk[0];
    val[1] ^= rk[1];
    val[2] ^= rk[2];
    val[3] ^= rk[3];
    val[4] ^= rk[4];
    val[5] ^= rk[5];
    val[6] ^= rk[6];
    val[7] ^= rk[7];
//    val[0] ^= rk[0];
//    val[1] ^= rk[1];
}
```



PIPO



```c
int main() {
    clock_t start, end;
    double result;

    printf("-------------------\n");
    //plain text 출력
    printf("Plain text \n");
    for(int i=2; i>0; i--) printf("0x%08x ", PLAIN_TEXT[i-1]);

    printf("\n-------------------\n");
    //key 출력
    printf("Master Key \n");
    for(int i=4; i>0; i--) printf("0x%08x ", MASTER_KEY[i-1]);

    printf("\n-------------------\n");
    ROUNDKEY_GEN(); //라운드키 생
    //round key 출력
    printf("Round Key \n");
    for(int i=0; i<28; i+=2) printf("0x%08X 0x%08X\n", ROUND_KEY[i+1], ROUND_KEY[i]);

    printf("\n-------------------\n");
    //암호화

    start = clock();    //시간 측정 시작
    ENCRYPTION_PIPO();
    end = clock();  //시간 측정 끝
    result = (double)(end - start);

    //cipher text 출력
    printf("Cipher text \n");
    for(int i=2; i>0; i--) printf("0x%08x ", PLAIN_TEXT[i-1]);

    printf("\n-------------------\n");
    printf("걸린 시간 : %f", result);

}
```

Q & A