AES 코드 분석

유튜브 주소 : https://youtu.be/jiZyOfyl0hQ

HSU 한성대학교 HANSUNG UNIVERSITY

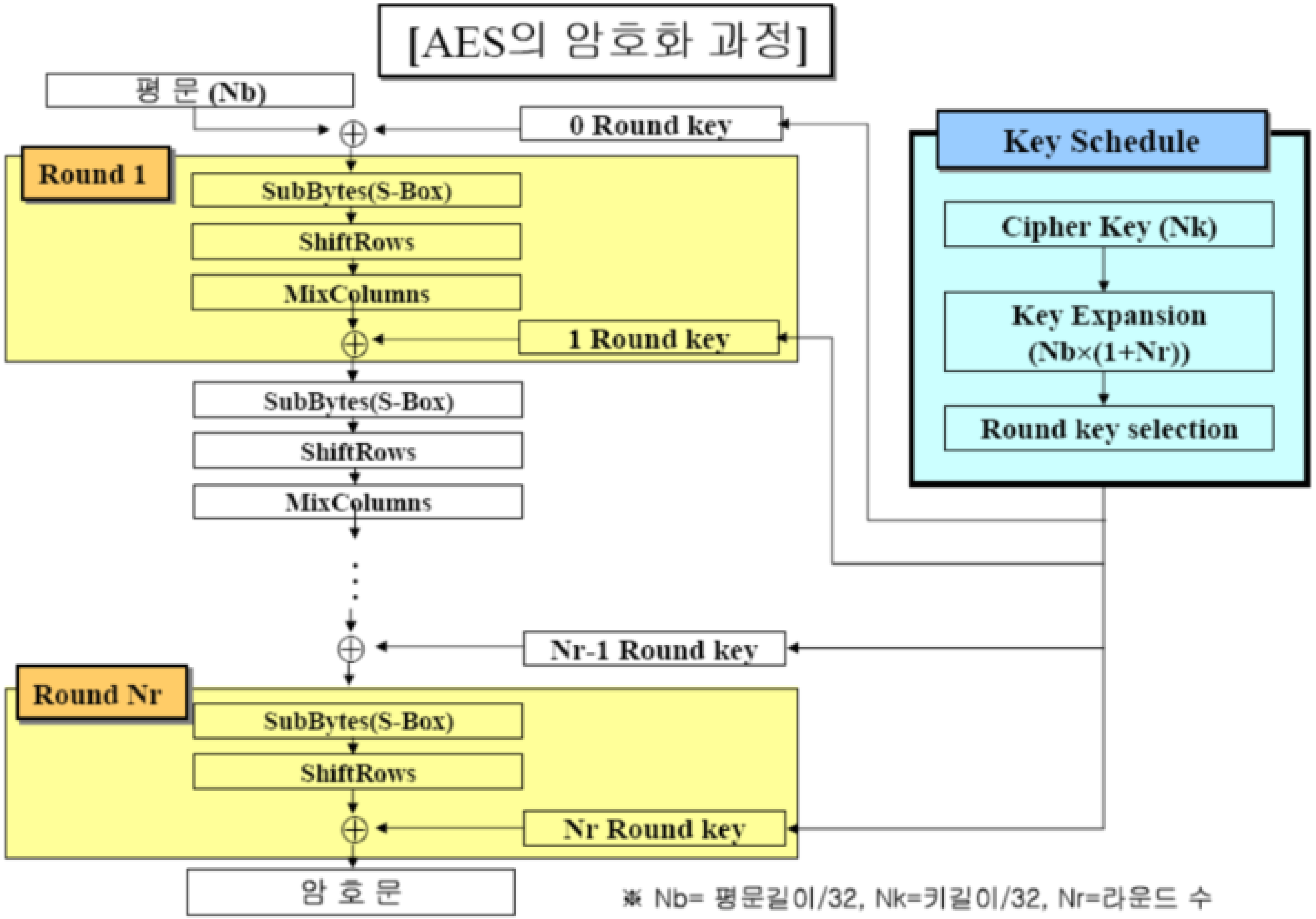
CryptoCraft LAB

암호화 코드 분석

키 스케줄 코드 분석

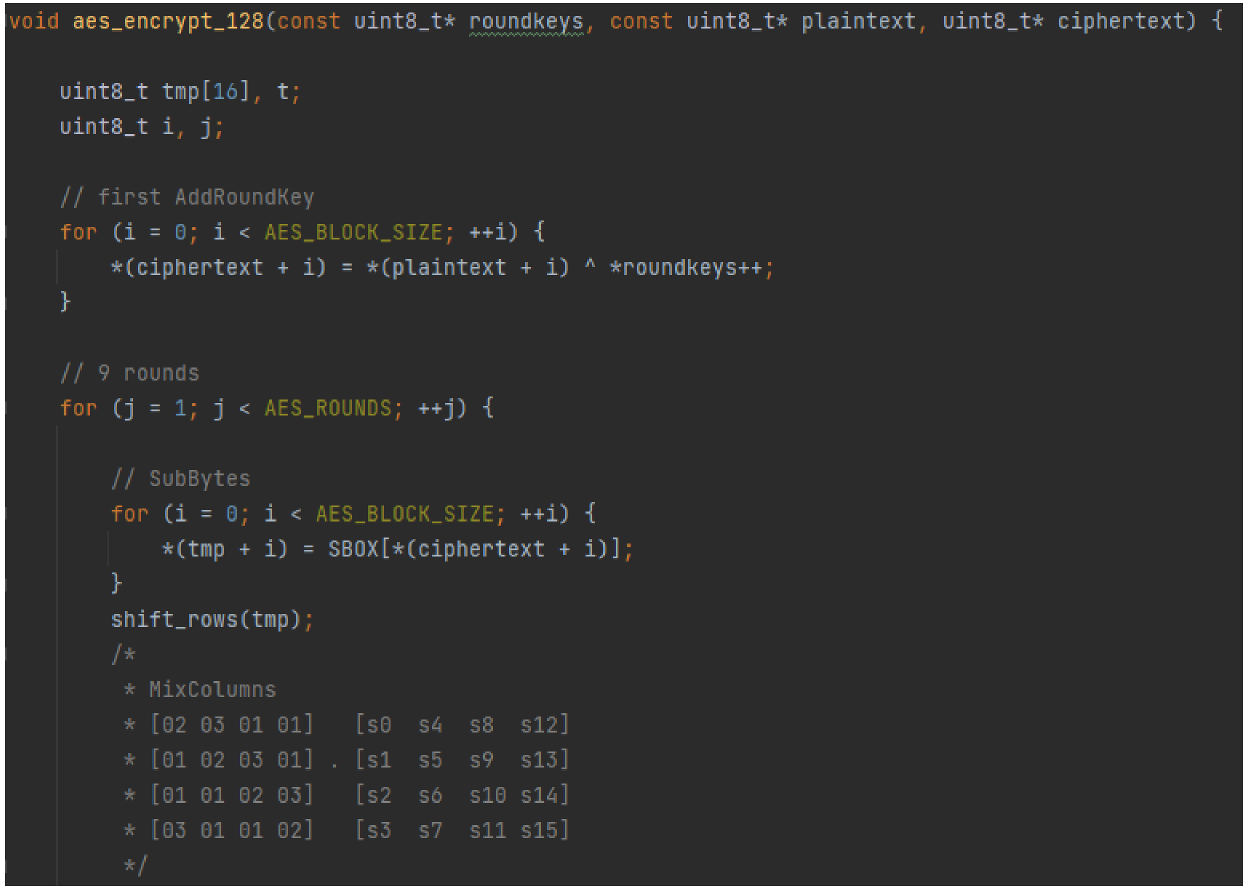
복호화 코드 분석

AES 암호화 과정

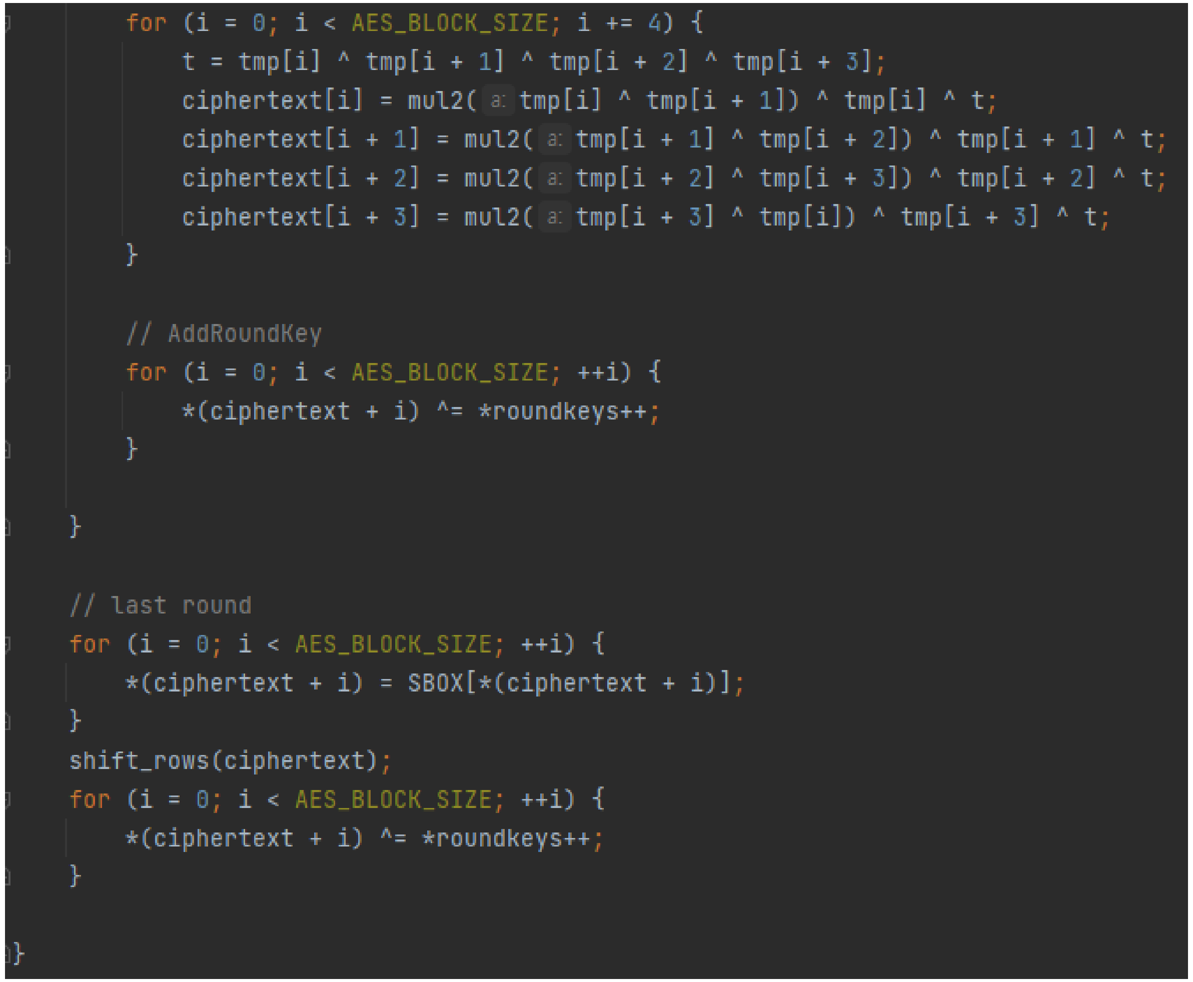


※ Nb= 평문길이/32, Nk=키길이/32, Nr=라운드 수

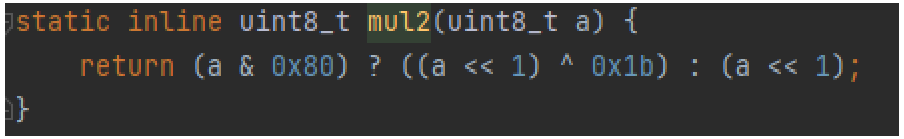
암호화 코드 분석



```c
void aes_encrypt_128(const uint8_t* roundkeys, const uint8_t* plaintext, uint8_t* ciphertext) {

    uint8_t tmp[16], t;
    uint8_t i, j;

    // first AddRoundKey
    for (i = 0; i < AES_BLOCK_SIZE; ++i) {
        *(ciphertext + i) = *(plaintext + i) ^ *roundkeys++;
    }

    // 9 rounds
    for (j = 1; j < AES_ROUNDS; ++j) {

        // SubBytes
        for (i = 0; i < AES_BLOCK_SIZE; ++i) {
            *(tmp + i) = SBOX[*(ciphertext + i)];
        }
        shift_rows(tmp);
        /*
         * MixColumns
         * [02 03 01 01]   [s0  s4  s8  s12]
         * [01 02 03 01] . [s1  s5  s9  s13]
         * [01 01 02 03]   [s2  s6  s10 s14]
         * [03 01 01 02]   [s3  s7  s11 s15]
         */
```



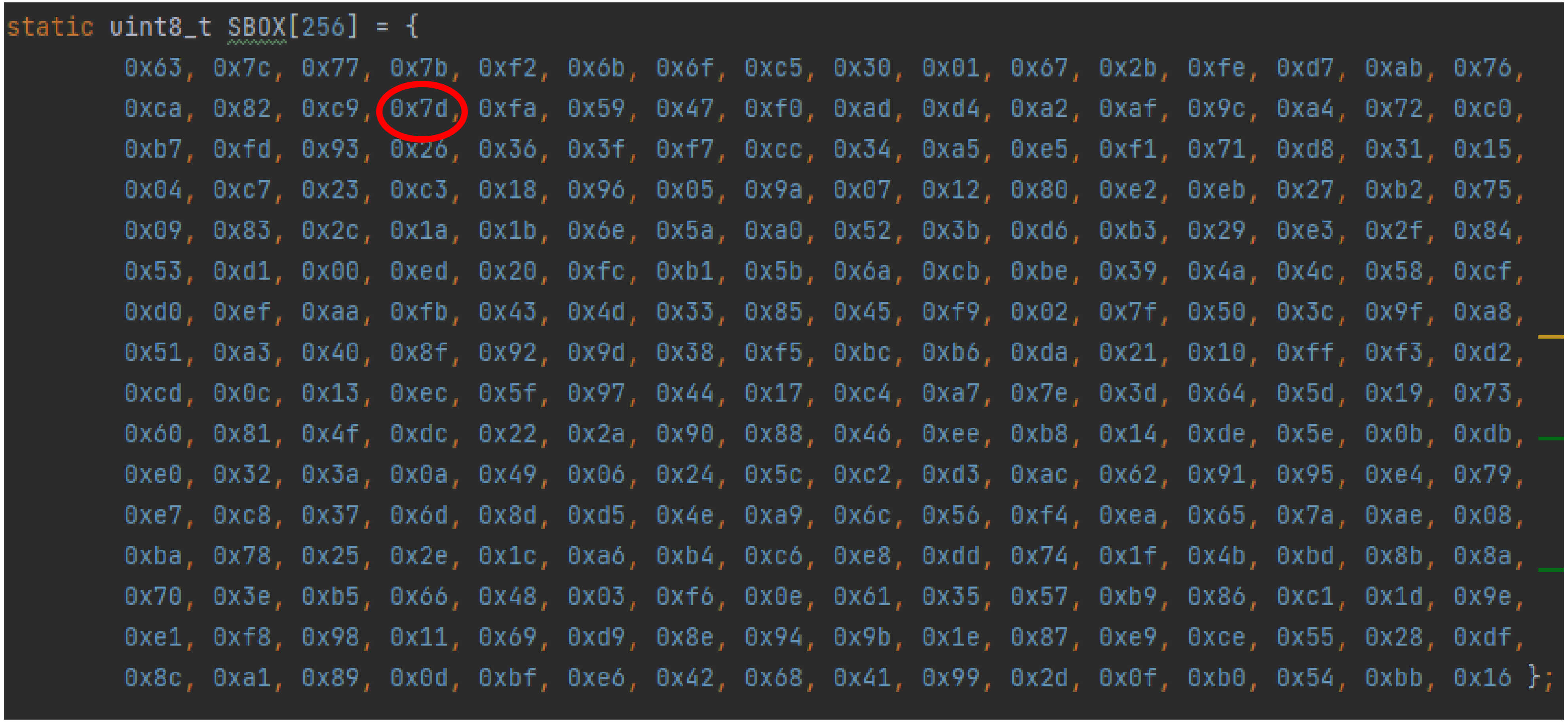
```c
        for (i = 0; i < AES_BLOCK_SIZE; i += 4) {
            t = tmp[i] ^ tmp[i + 1] ^ tmp[i + 2] ^ tmp[i + 3];
            ciphertext[i] = mul2( a: tmp[i] ^ tmp[i + 1]) ^ tmp[i] ^ t;
            ciphertext[i + 1] = mul2( a: tmp[i + 1] ^ tmp[i + 2]) ^ tmp[i + 1] ^ t;
            ciphertext[i + 2] = mul2( a: tmp[i + 2] ^ tmp[i + 3]) ^ tmp[i + 2] ^ t;
            ciphertext[i + 3] = mul2( a: tmp[i + 3] ^ tmp[i]) ^ tmp[i + 3] ^ t;
        }

        // AddRoundKey
        for (i = 0; i < AES_BLOCK_SIZE; ++i) {
            *(ciphertext + i) ^= *roundkeys++;
        }

    }

    // last round
    for (i = 0; i < AES_BLOCK_SIZE; ++i) {
        *(ciphertext + i) = SBOX[*(ciphertext + i)];
    }
    shift_rows(ciphertext);
    for (i = 0; i < AES_BLOCK_SIZE; ++i) {
        *(ciphertext + i) ^= *roundkeys++;
    }

}
```



```c
static inline uint8_t mul2(uint8_t a) {
    return (a & 0x80) ? ((a << 1) ^ 0x1b) : (a << 1);
}
```

암호화 코드 분석

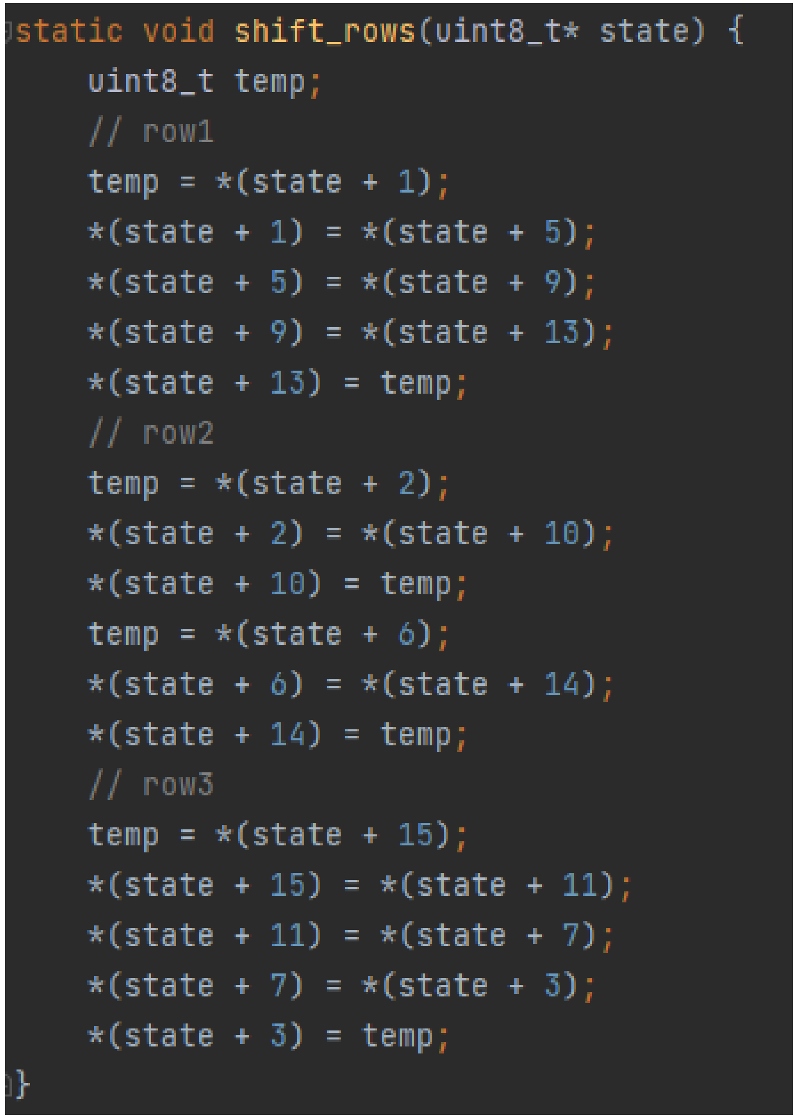
- S-box



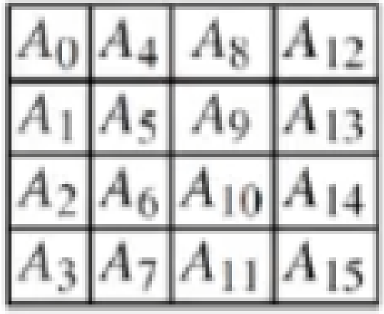
```
static uint8_t SBOX[256] = {
        0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
        0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
        0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
        0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
        0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
        0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
        0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
        0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
        0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
        0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
        0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
        0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
        0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
        0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
        0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
        0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };
```

Ex) 0x13 -> 0x7d

암호화 코드 분석

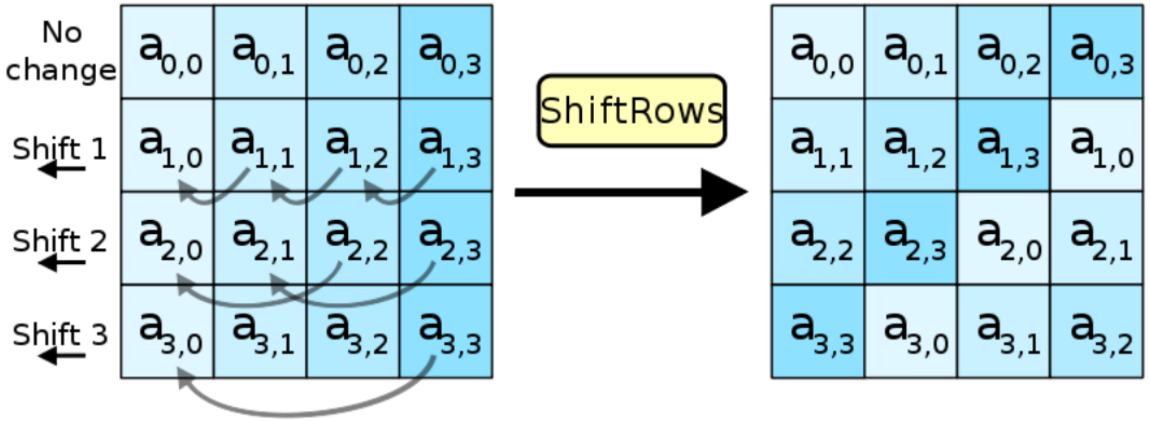


```
static void shift_rows(uint8_t* state) {
    uint8_t temp;
    // row1
    temp = *(state + 1);
    *(state + 1) = *(state + 5);
    *(state + 5) = *(state + 9);
    *(state + 9) = *(state + 13);
    *(state + 13) = temp;
    // row2
    temp = *(state + 2);
    *(state + 2) = *(state + 10);
    *(state + 10) = temp;
    temp = *(state + 6);
    *(state + 6) = *(state + 14);
    *(state + 14) = temp;
    // row3
    temp = *(state + 15);
    *(state + 15) = *(state + 11);
    *(state + 11) = *(state + 7);
    *(state + 7) = *(state + 3);
    *(state + 3) = temp;
}
```

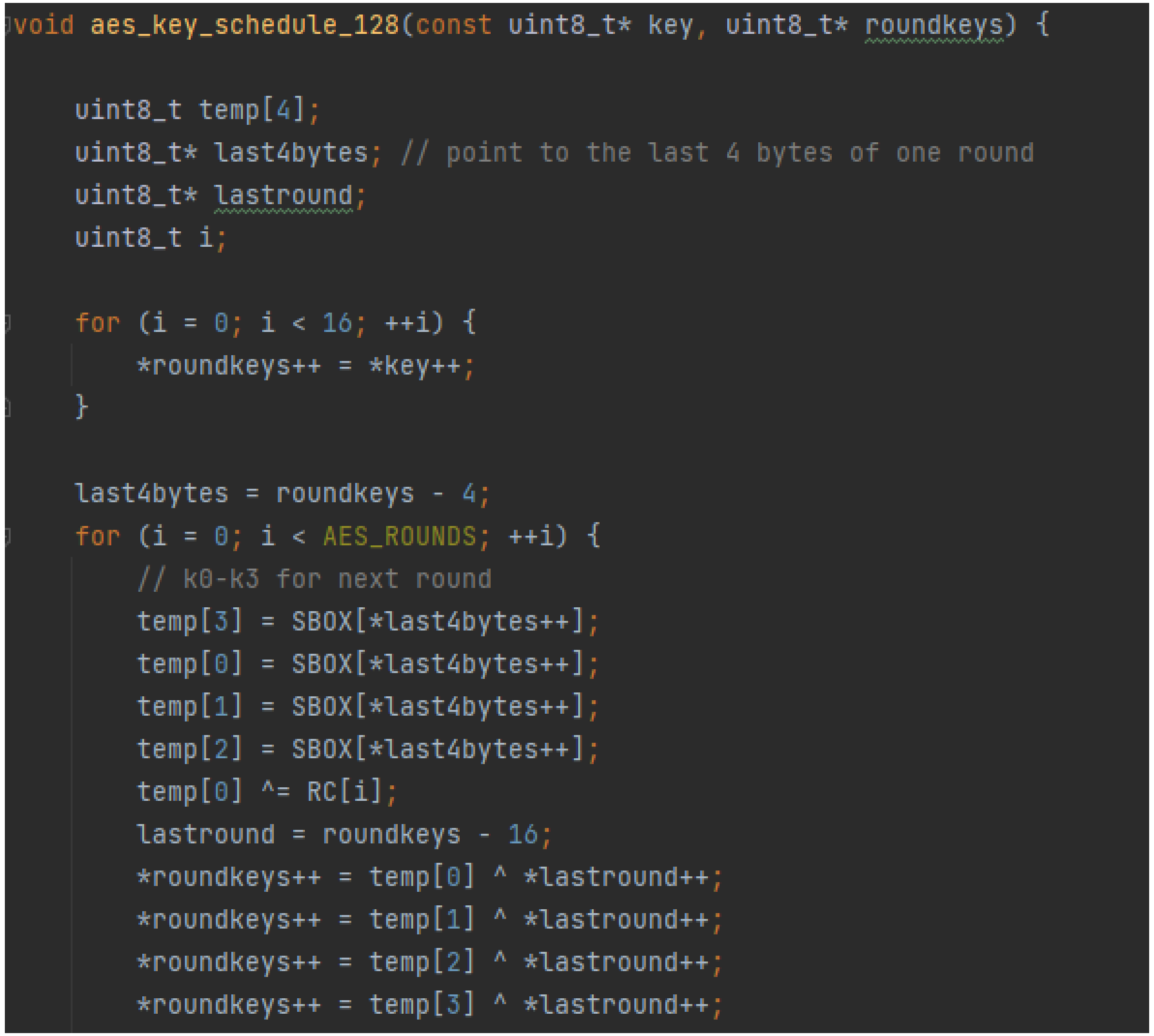


A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

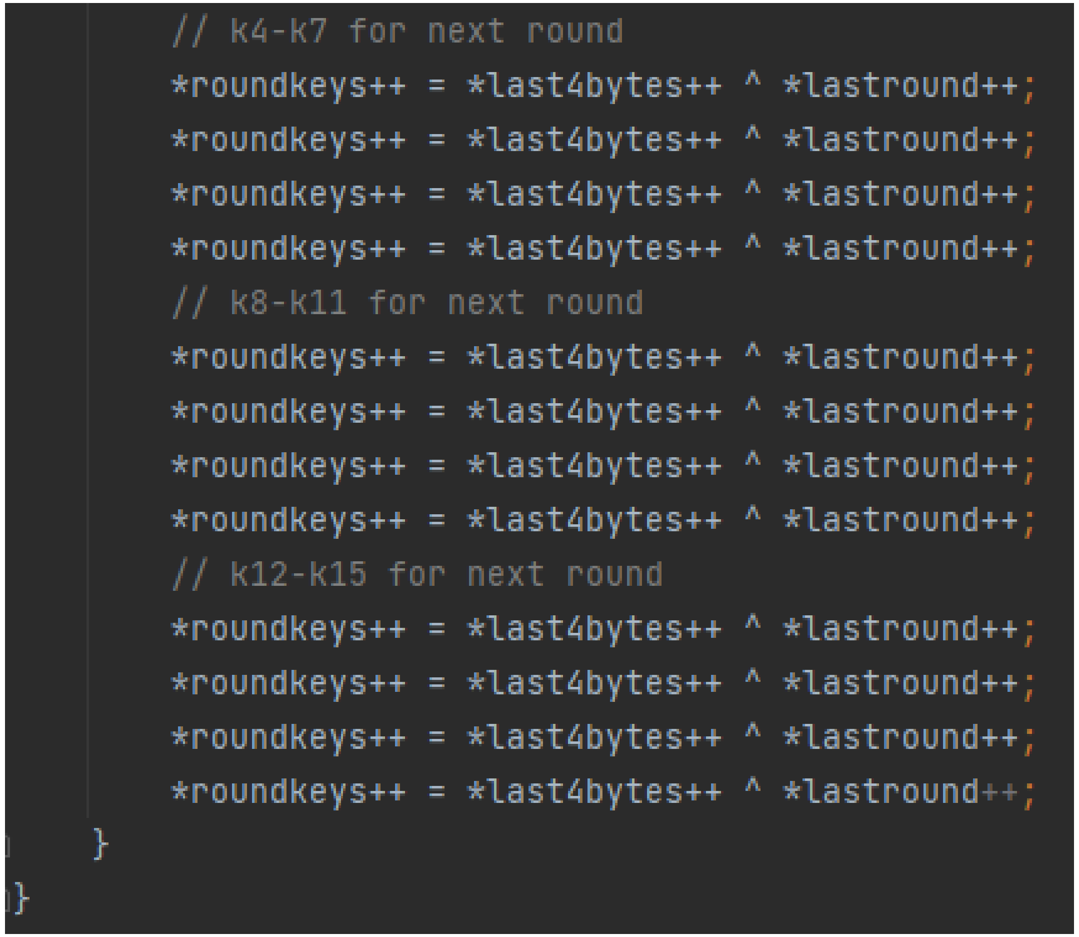
AES에서 데이터의 상태
(STATE) – 행렬 구조


No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

키 스케줄 코드 분석



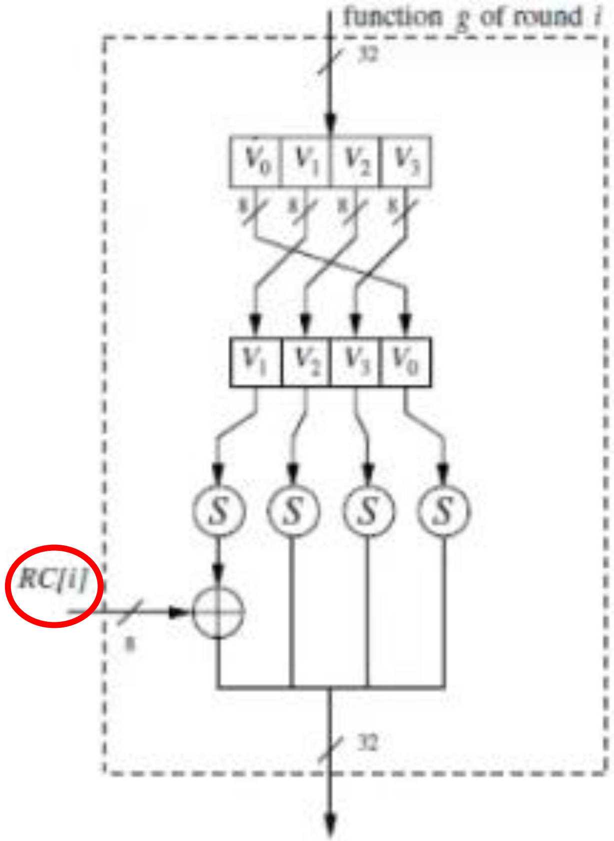
```c
void aes_key_schedule_128(const uint8_t* key, uint8_t* roundkeys) {

    uint8_t temp[4];
    uint8_t* last4bytes; // point to the last 4 bytes of one round
    uint8_t* lastround;
    uint8_t i;

    for (i = 0; i < 16; ++i) {
        *roundkeys++ = *key++;
    }

    last4bytes = roundkeys - 4;
    for (i = 0; i < AES_ROUNDS; ++i) {
        // k0-k3 for next round
        temp[3] = SBOX[*last4bytes++];
        temp[0] = SBOX[*last4bytes++];
        temp[1] = SBOX[*last4bytes++];
        temp[2] = SBOX[*last4bytes++];
        temp[0] ^= RC[i];
        lastround = roundkeys - 16;
        *roundkeys++ = temp[0] ^ *lastround++;
        *roundkeys++ = temp[1] ^ *lastround++;
        *roundkeys++ = temp[2] ^ *lastround++;
        *roundkeys++ = temp[3] ^ *lastround++;
```

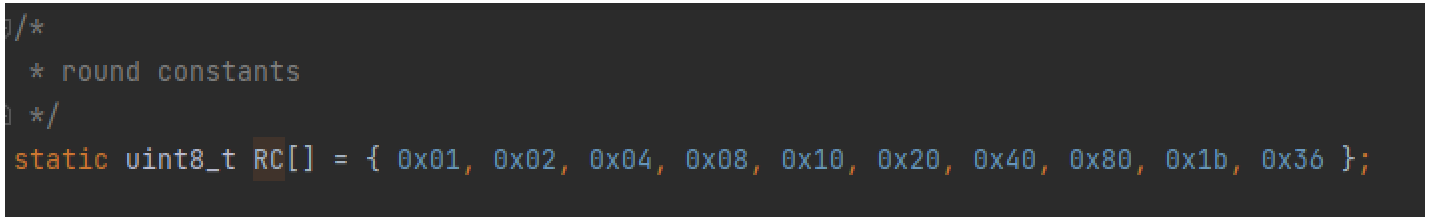


```c
        // k4-k7 for next round
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        // k8-k11 for next round
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        // k12-k15 for next round
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
        *roundkeys++ = *last4bytes++ ^ *lastround++;
    }
}
```

키 스케줄 코드 분석

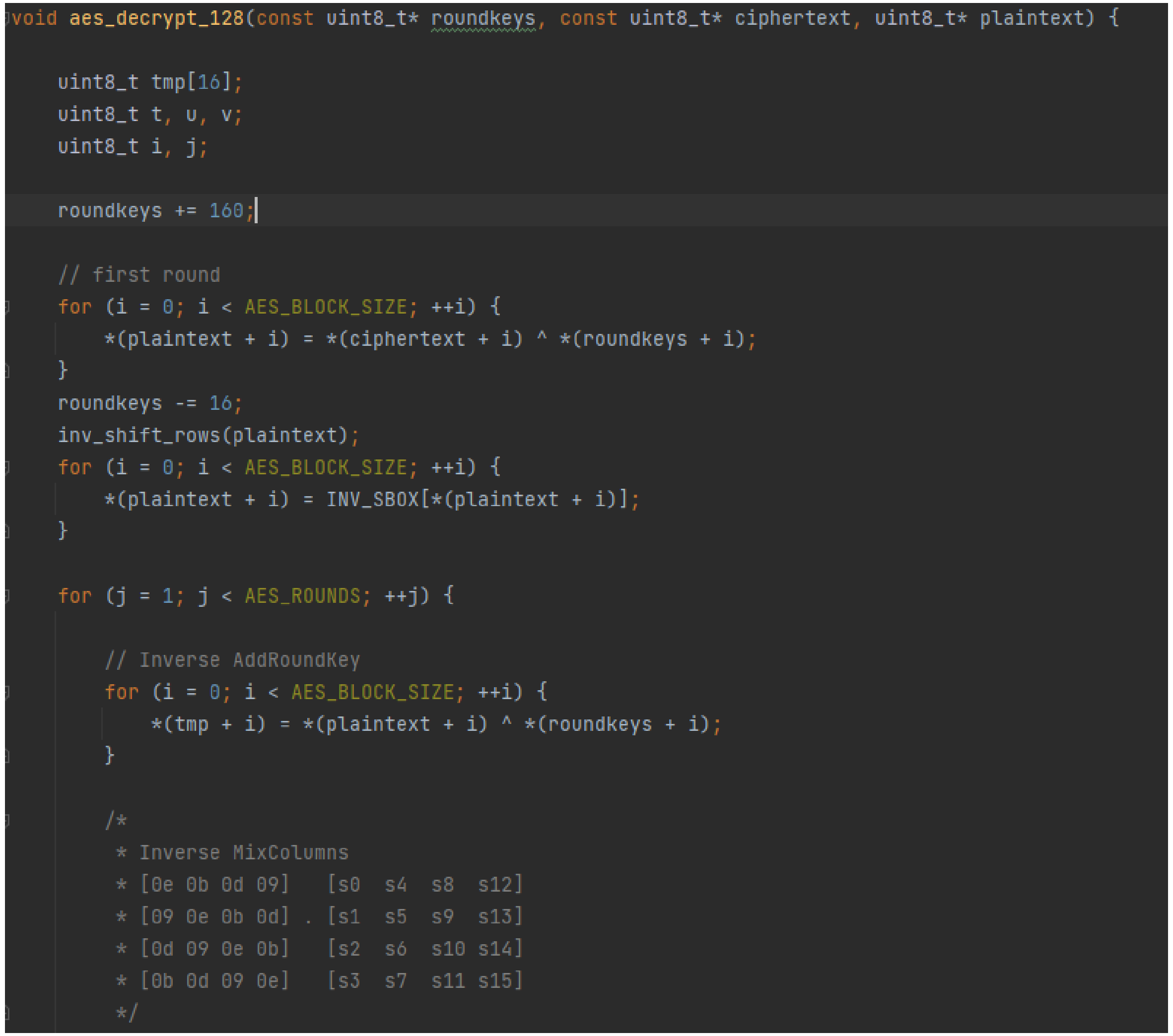
라운드 계수



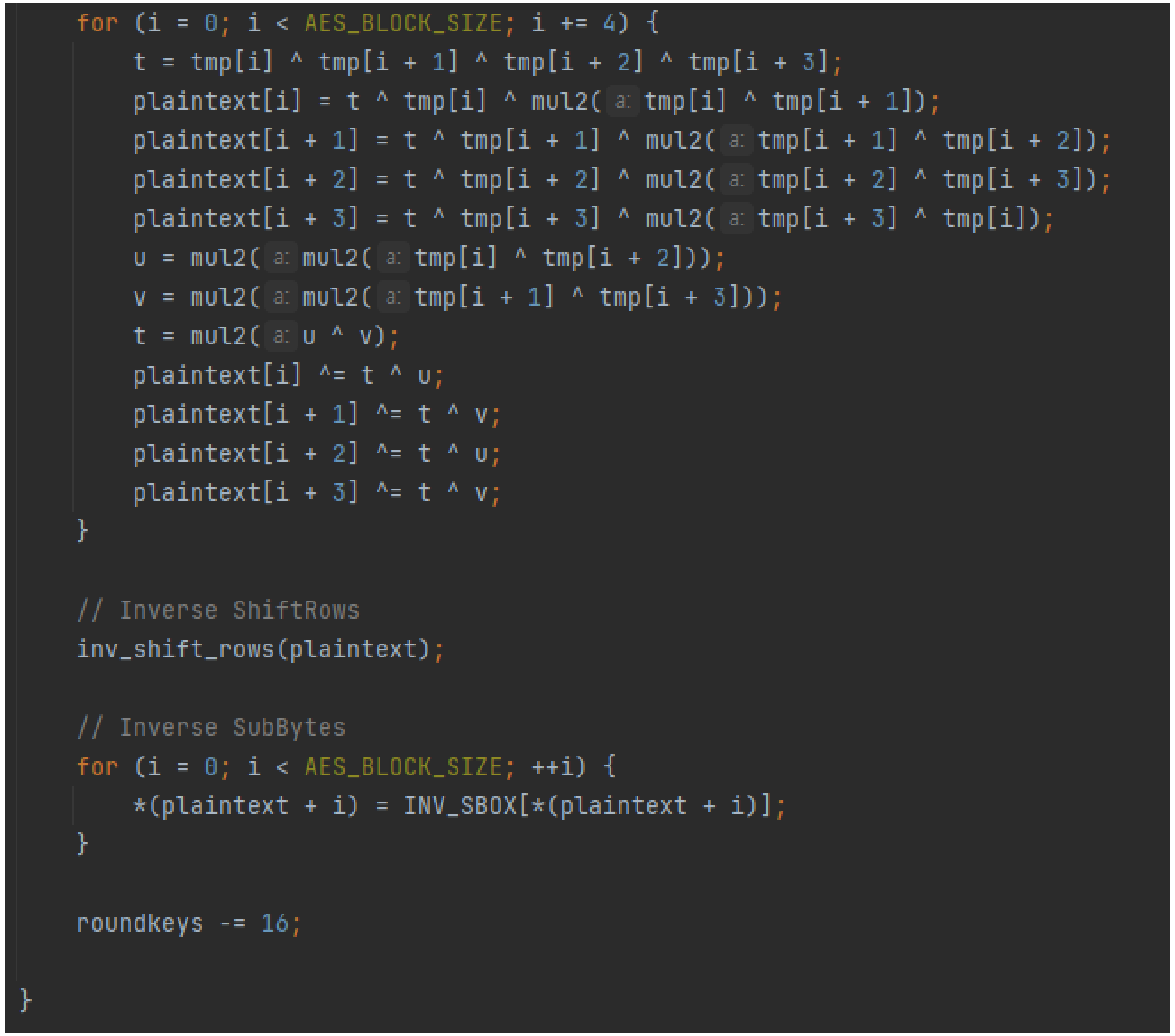


```
/*
 * round constants
 */
static uint8_t RC[] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36 };
```

복호화 코드 분석



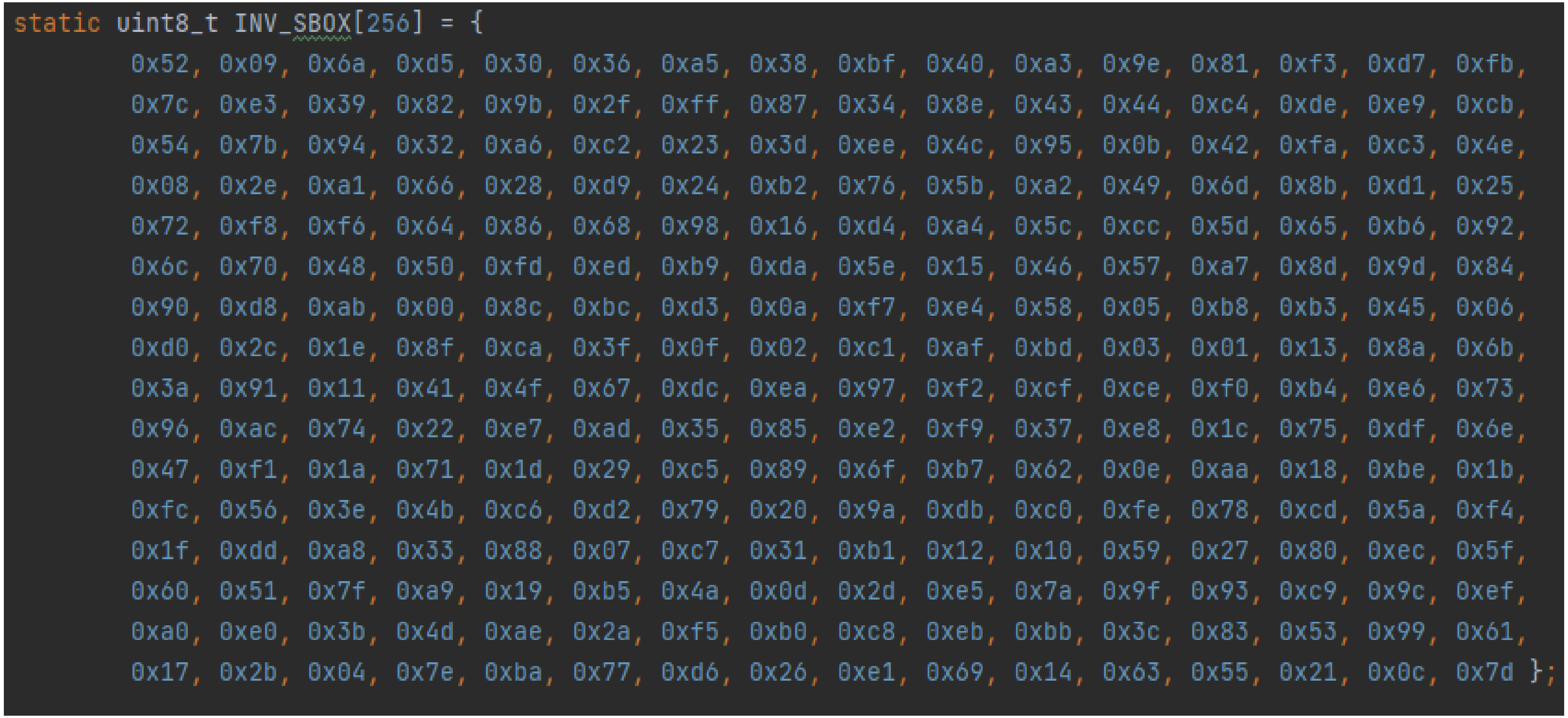
```
void aes_decrypt_128(const uint8_t* roundkeys, const uint8_t* ciphertext, uint8_t* plaintext) {

    uint8_t tmp[16];
    uint8_t t, u, v;
    uint8_t i, j;

    roundkeys += 160;

    // first round
    for (i = 0; i < AES_BLOCK_SIZE; ++i) {
        *(plaintext + i) = *(ciphertext + i) ^ *(roundkeys + i);
    }
    roundkeys -= 16;
    inv_shift_rows(plaintext);
    for (i = 0; i < AES_BLOCK_SIZE; ++i) {
        *(plaintext + i) = INV_SBOX[*(plaintext + i)];
    }

    for (j = 1; j < AES_ROUNDS; ++j) {

        // Inverse AddRoundKey
        for (i = 0; i < AES_BLOCK_SIZE; ++i) {
            *(tmp + i) = *(plaintext + i) ^ *(roundkeys + i);
        }

        /*
         * Inverse MixColumns
         * [0e 0b 0d 09]   [s0  s4  s8  s12]
         * [09 0e 0b 0d] . [s1  s5  s9  s13]
         * [0d 09 0e 0b]   [s2  s6  s10 s14]
         * [0b 0d 09 0e]   [s3  s7  s11 s15]
         */
```



```
        for (i = 0; i < AES_BLOCK_SIZE; i += 4) {
            t = tmp[i] ^ tmp[i + 1] ^ tmp[i + 2] ^ tmp[i + 3];
            plaintext[i] = t ^ tmp[i] ^ mul2( a: tmp[i] ^ tmp[i + 1]);
            plaintext[i + 1] = t ^ tmp[i + 1] ^ mul2( a: tmp[i + 1] ^ tmp[i + 2]);
            plaintext[i + 2] = t ^ tmp[i + 2] ^ mul2( a: tmp[i + 2] ^ tmp[i + 3]);
            plaintext[i + 3] = t ^ tmp[i + 3] ^ mul2( a: tmp[i + 3] ^ tmp[i]);
            u = mul2( a: mul2( a: tmp[i] ^ tmp[i + 2]));
            v = mul2( a: mul2( a: tmp[i + 1] ^ tmp[i + 3]));
            t = mul2( a: u ^ v);
            plaintext[i] ^= t ^ u;
            plaintext[i + 1] ^= t ^ v;
            plaintext[i + 2] ^= t ^ u;
            plaintext[i + 3] ^= t ^ v;
        }

        // Inverse ShiftRows
        inv_shift_rows(plaintext);

        // Inverse SubBytes
        for (i = 0; i < AES_BLOCK_SIZE; ++i) {
            *(plaintext + i) = INV_SBOX[*(plaintext + i)];
        }

        roundkeys -= 16;

    }
```

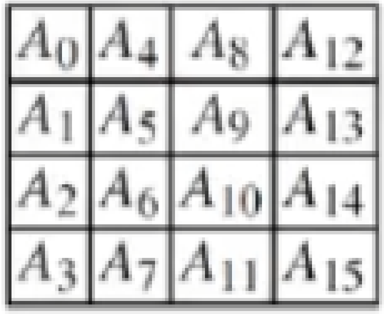
복호화 코드 분석

- Inverse S-box

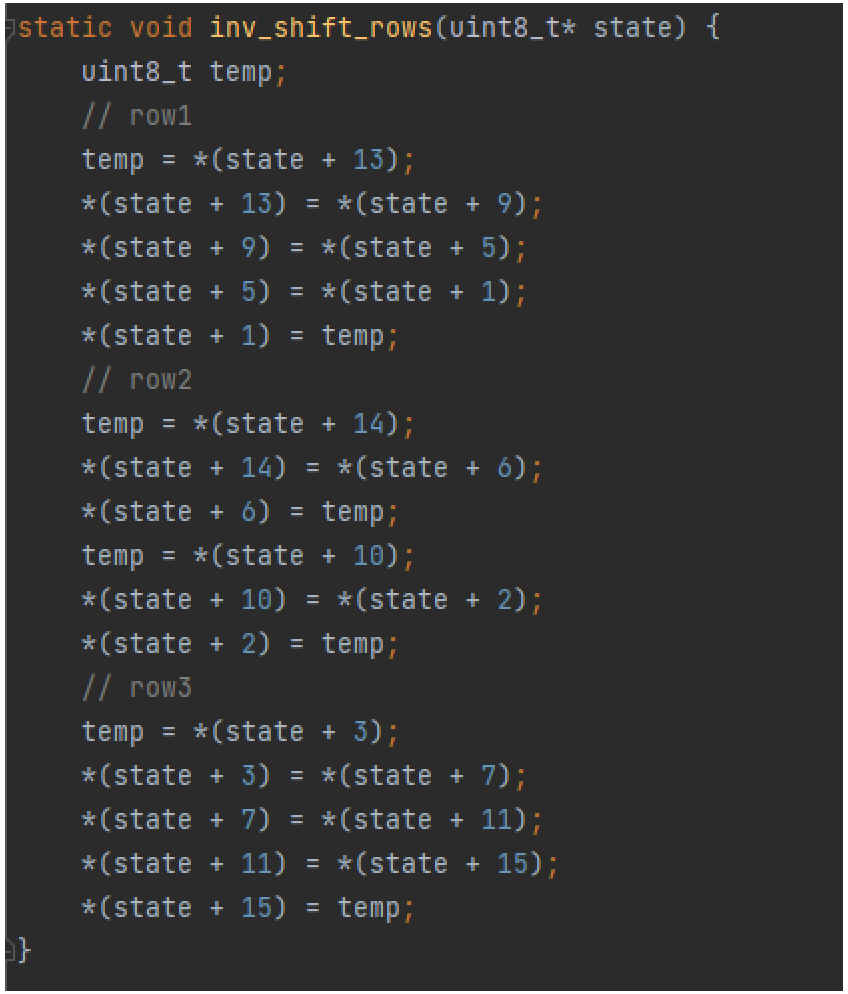


```
static uint8_t INV_SBOX[256] = {
        0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
        0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
        0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
        0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
        0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
        0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
        0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
        0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
        0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
        0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
        0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
        0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
        0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
        0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
        0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
        0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d };
```

복호화 코드 분석



A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}



```c
static void inv_shift_rows(uint8_t* state) {
    uint8_t temp;
    // row1
    temp = *(state + 13);
    *(state + 13) = *(state + 9);
    *(state + 9) = *(state + 5);
    *(state + 5) = *(state + 1);
    *(state + 1) = temp;
    // row2
    temp = *(state + 14);
    *(state + 14) = *(state + 6);
    *(state + 6) = temp;
    temp = *(state + 10);
    *(state + 10) = *(state + 2);
    *(state + 2) = temp;
    // row3
    temp = *(state + 3);
    *(state + 3) = *(state + 7);
    *(state + 7) = *(state + 11);
    *(state + 11) = *(state + 15);
    *(state + 15) = temp;
}
```

코드 출처

https://github.com/openluopworld/aes_128

Q & A