ARMv8상에서 Classic McEliece의 Multiplication and Inversion operation 구현

https://youtu.be/g43mKHFng94

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Classic McEliece

- NIST PQC 4라운드 후보군 중 유일한 코드기반 암호
- Key generation, Encapsulation, Decapsulation 3단계 수행
- Extended binary finite-filed $\boldsymbol{F_{2^m}}$ 상의 Multiplication 과 Inversion 연산
  - Public keygen : inverse + multiplication on $\boldsymbol{F_{2^m}}$
  - Secret keygen : inverse + multiplication on $\boldsymbol{F_{2^m}}$
  - Encap : X
  - Decap : inverse + multiplication on $\boldsymbol{F_{2^m}}$

$$\boldsymbol{F_{2^{12}}} = \mathbb{F}_2[x]\,/(x^{12} + x^3 + 1).$$

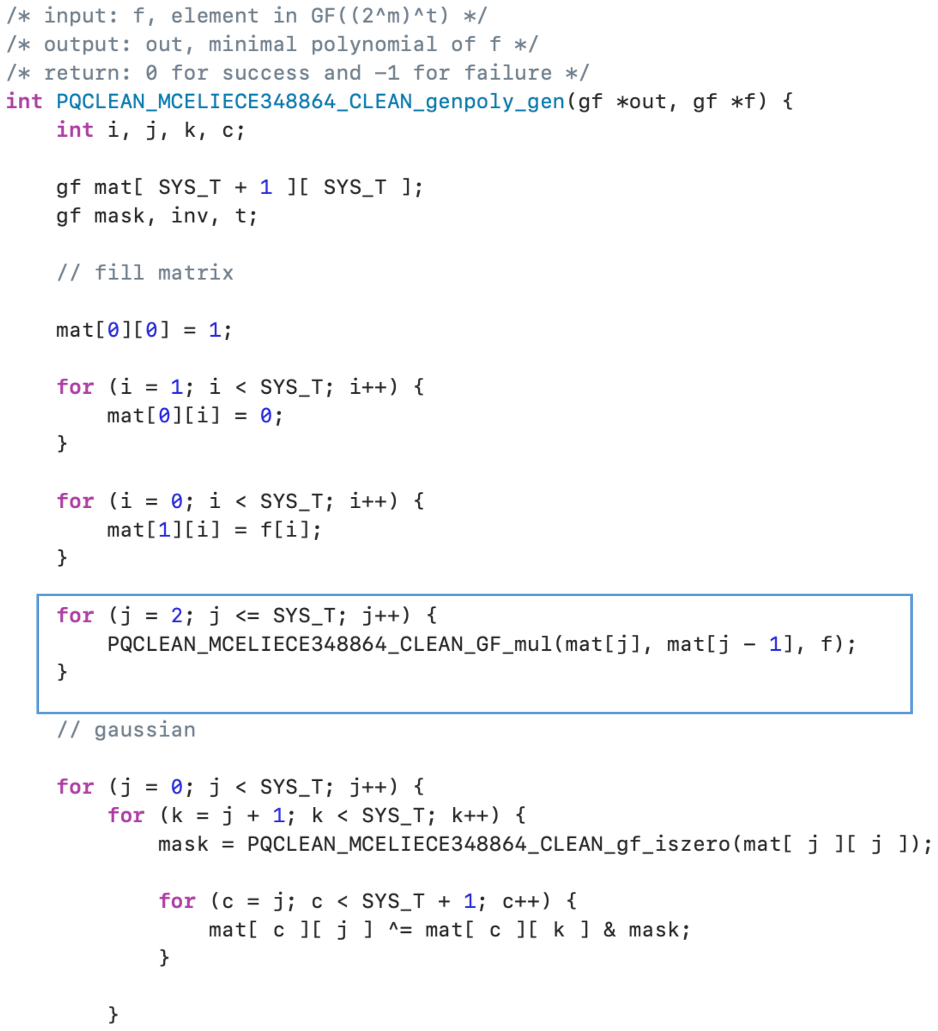
$$\boldsymbol{F_{2^{13}}} = \mathbb{F}_2[x]\,/\,(x^{13} + x^4 + x^3 + x + 1).$$

Algorithm	m	n	t	level	Public key	Secret key	Ciphertext
Mceliece 348864	12	3488	64	1	261,120	6,492	128
Mceliece 460896	13	4608	86	3	524,160	13,608	188
Mceliece 6688128	13	6688	128	5	1,044,992	13,932	240
Mceliece 6960119	13	6960	119	5	1,047,319	13,948	226
Mceliece 8192128	13	8192	128	5	1,357,824	14,120	240

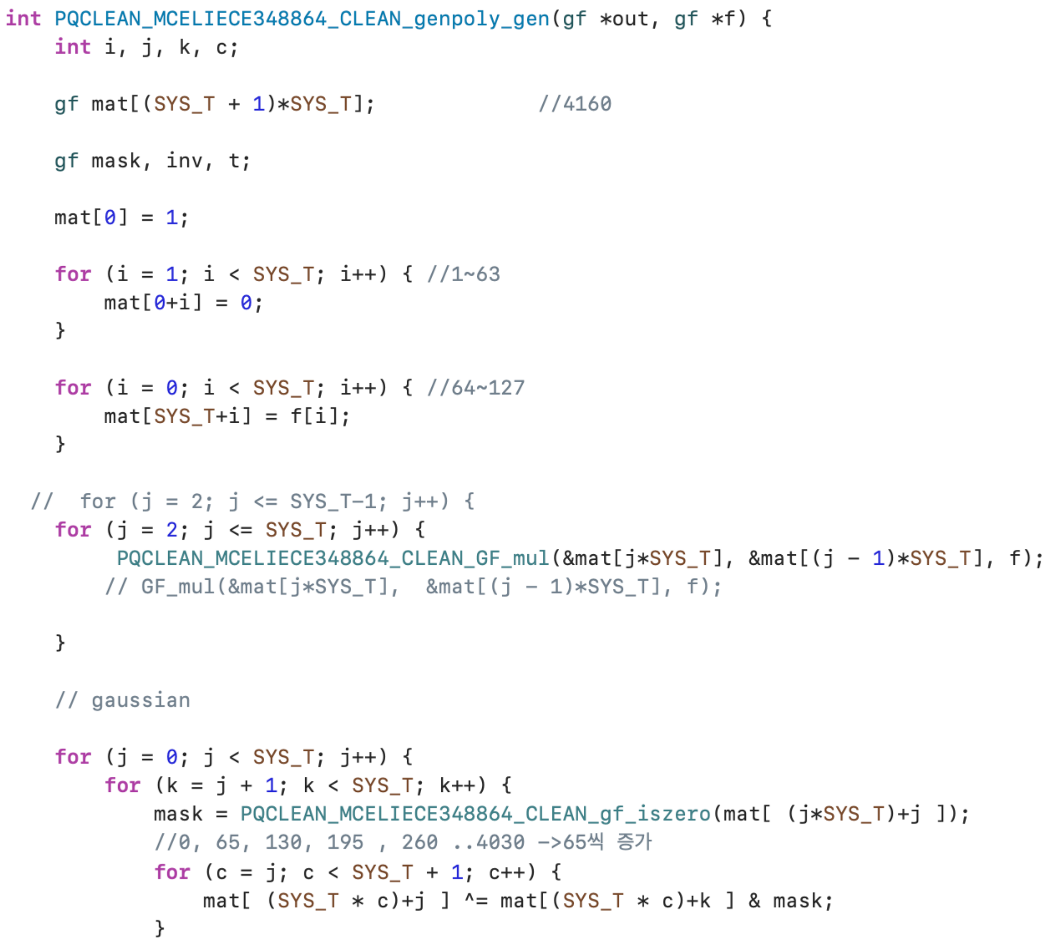
Parameters of Classic McEliece

Multiplication

- Secret key gen의 일부

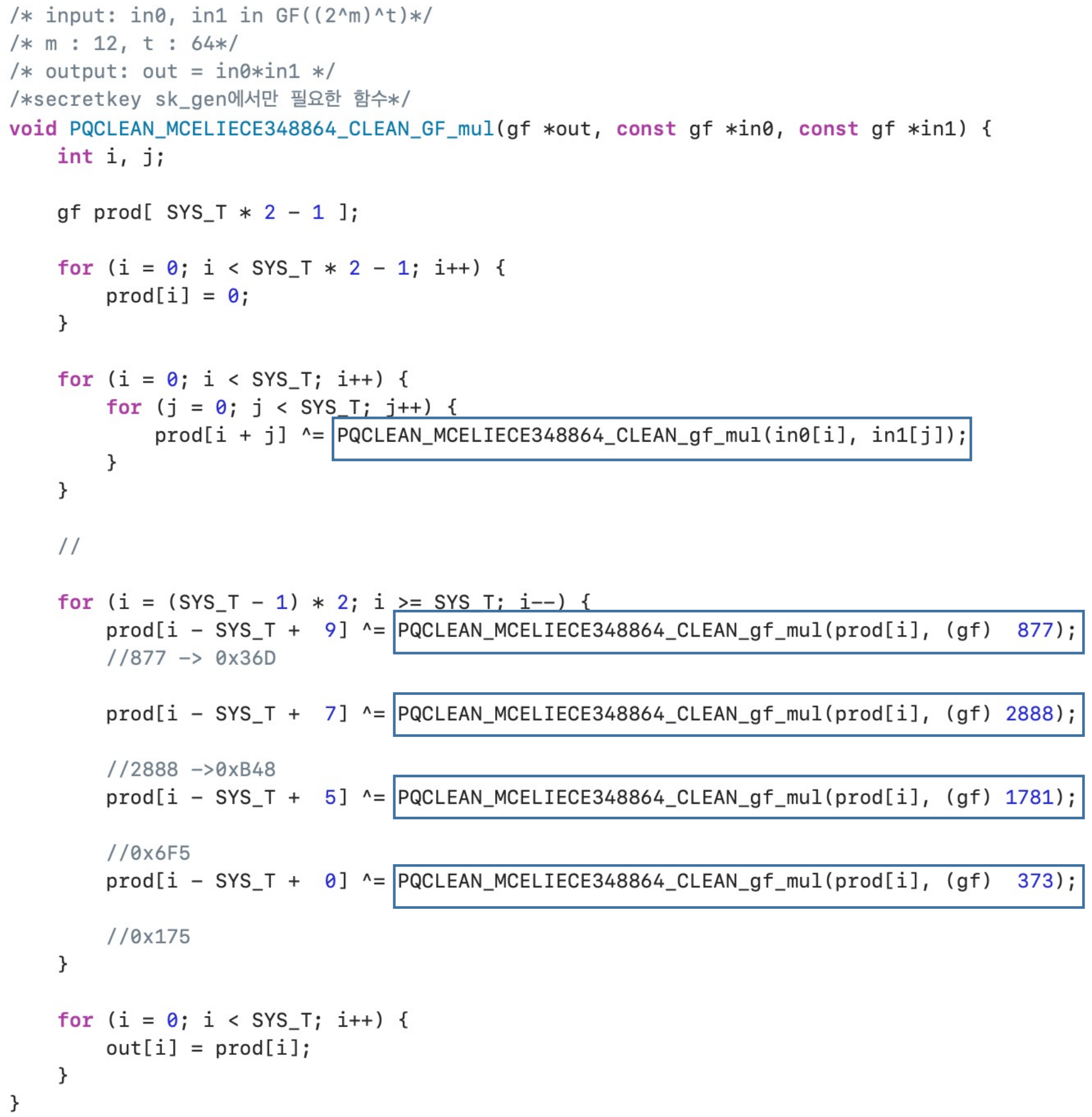


```c
/* input: f, element in GF((2^m)^t) */
/* output: out, minimal polynomial of f */
/* return: 0 for success and -1 for failure */
int PQCLEAN_MCELIECE348864_CLEAN_genpoly_gen(gf *out, gf *f) {
    int i, j, k, c;

    gf mat[ SYS_T + 1 ][ SYS_T ];
    gf mask, inv, t;

    // fill matrix

    mat[0][0] = 1;

    for (i = 1; i < SYS_T; i++) {
        mat[0][i] = 0;
    }

    for (i = 0; i < SYS_T; i++) {
        mat[1][i] = f[i];
    }

    for (j = 2; j <= SYS_T; j++) {
        PQCLEAN_MCELIECE348864_CLEAN_GF_mul(mat[j], mat[j - 1], f);
    }

    // gaussian

    for (j = 0; j < SYS_T; j++) {
        for (k = j + 1; k < SYS_T; k++) {
            mask = PQCLEAN_MCELIECE348864_CLEAN_gf_iszero(mat[ j ][ j ]);

            for (c = j; c < SYS_T + 1; c++) {
                mat[ c ][ j ] ^= mat[ c ][ k ] & mask;
            }

        }
```

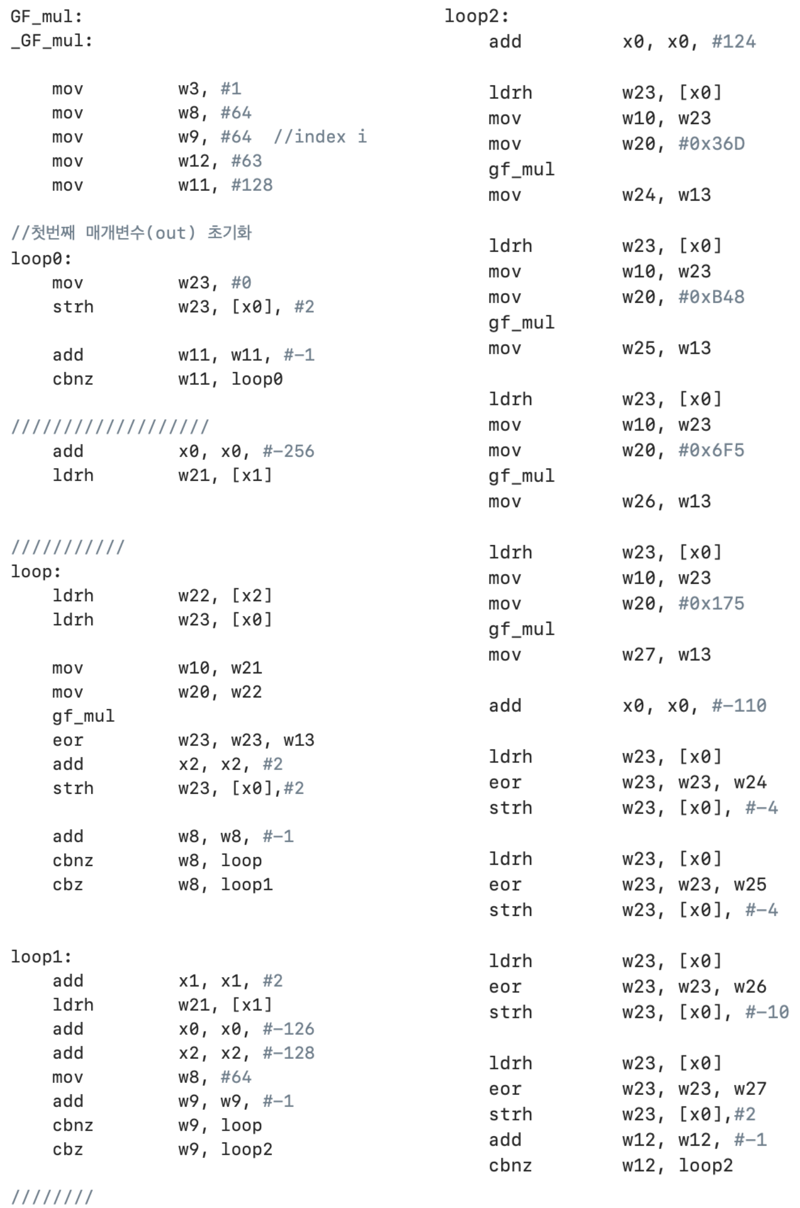


```c
int PQCLEAN_MCELIECE348864_CLEAN_genpoly_gen(gf *out, gf *f) {
    int i, j, k, c;

    gf mat[(SYS_T + 1)*SYS_T];             //4160

    gf mask, inv, t;

    mat[0] = 1;

    for (i = 1; i < SYS_T; i++) { //1~63
        mat[0+i] = 0;
    }

    for (i = 0; i < SYS_T; i++) { //64~127
        mat[SYS_T+i] = f[i];
    }

  //  for (j = 2; j <= SYS_T-1; j++) {
    for (j = 2; j <= SYS_T; j++) {
         PQCLEAN_MCELIECE348864_CLEAN_GF_mul(&mat[j*SYS_T], &mat[(j - 1)*SYS_T], f);
        // GF_mul(&mat[j*SYS_T],  &mat[(j - 1)*SYS_T], f);

    }

    // gaussian

    for (j = 0; j < SYS_T; j++) {
        for (k = j + 1; k < SYS_T; k++) {
            mask = PQCLEAN_MCELIECE348864_CLEAN_gf_iszero(mat[ (j*SYS_T)+j ]);
            //0, 65, 130, 195 , 260 ..4030 ->65씩 증가
            for (c = j; c < SYS_T + 1; c++) {
                mat[ (SYS_T * c)+j ] ^= mat[(SYS_T * c)+k ] & mask;
            }
```

Multiplication



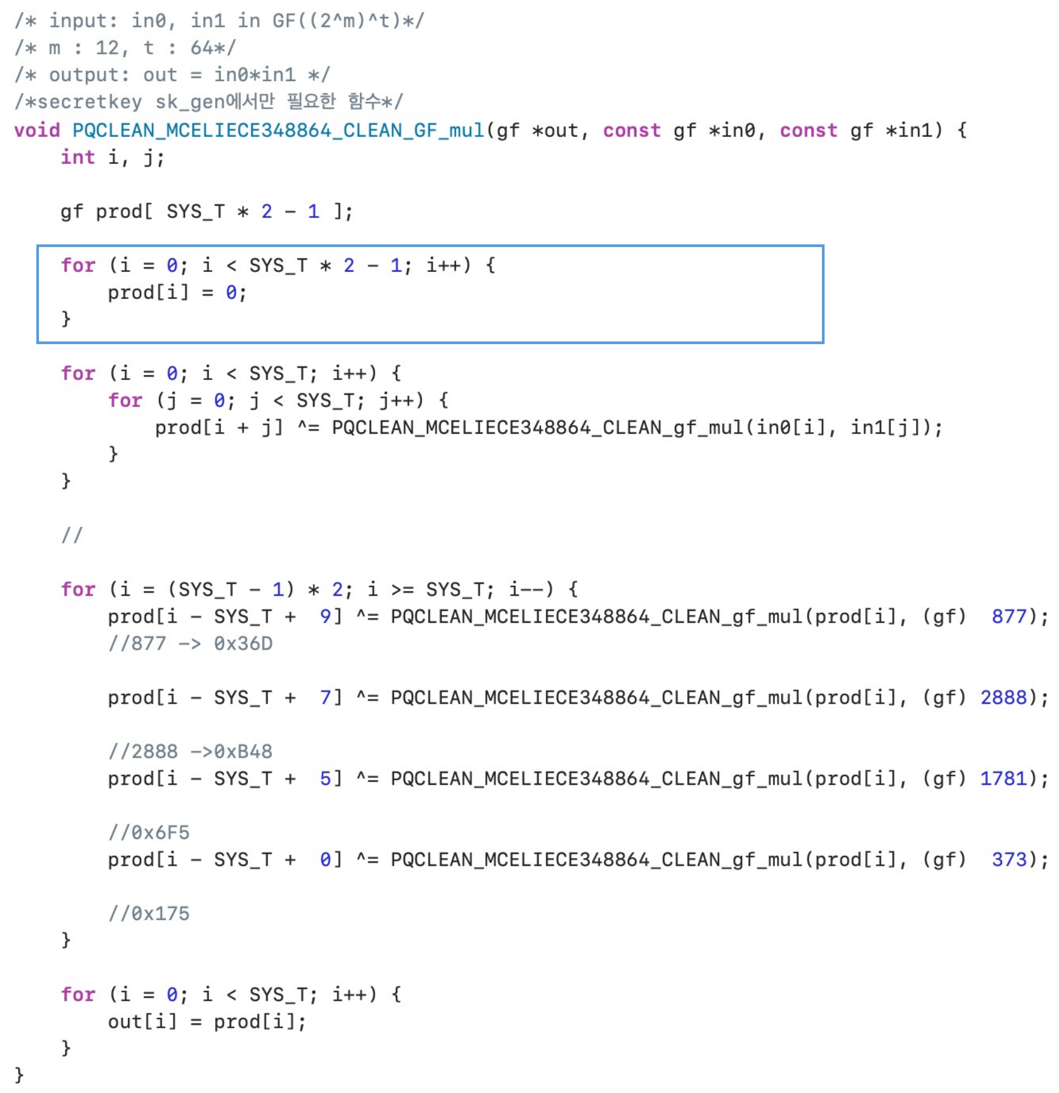
```c
/* input: in0, in1 in GF((2^m)^t)*/
/* m : 12, t : 64*/
/* output: out = in0*in1 */
/*secretkey sk_gen에서만 필요한 함수*/
void PQCLEAN_MCELIECE348864_CLEAN_GF_mul(gf *out, const gf *in0, const gf *in1) {
    int i, j;

    gf prod[ SYS_T * 2 - 1 ];

    for (i = 0; i < SYS_T * 2 - 1; i++) {
        prod[i] = 0;
    }

    for (i = 0; i < SYS_T; i++) {
        for (j = 0; j < SYS_T; j++) {
            prod[i + j] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(in0[i], in1[j]);
        }
    }

    //

    for (i = (SYS_T - 1) * 2; i >= SYS_T; i--) {
        prod[i - SYS_T +  9] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  877);
        //877 -> 0x36D

        prod[i - SYS_T +  7] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 2888);

        //2888 ->0xB48
        prod[i - SYS_T +  5] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 1781);

        //0x6F5
        prod[i - SYS_T +  0] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  373);

        //0x175
    }

    for (i = 0; i < SYS_T; i++) {
        out[i] = prod[i];
    }
}
```



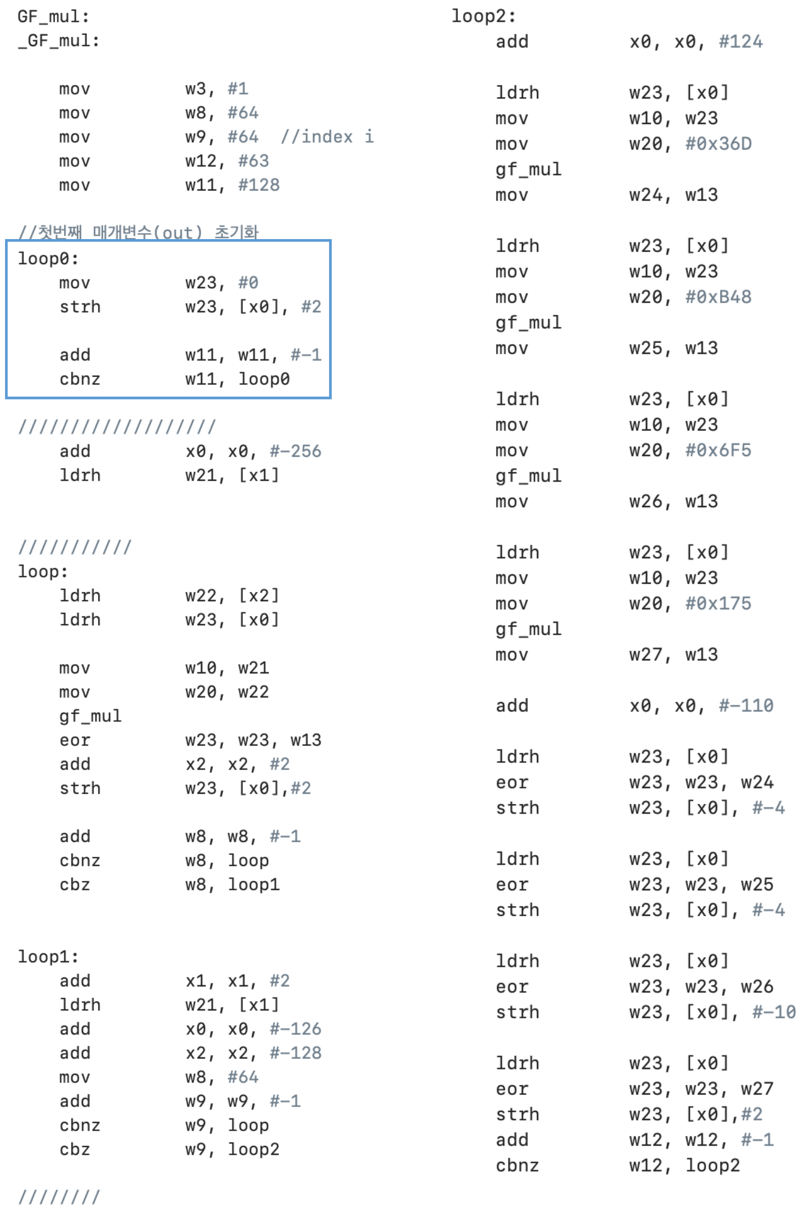
```asm
GF_mul:
_GF_mul:

    mov         w3, #1
    mov         w8, #64
    mov         w9, #64  //index i
    mov         w12, #63
    mov         w11, #128

//첫번째 매개변수(out) 초기화
loop0:
    mov         w23, #0
    strh        w23, [x0], #2

    add         w11, w11, #-1
    cbnz        w11, loop0

//////////////////
    add         x0, x0, #-256
    ldrh        w21, [x1]


//////////
loop:
    ldrh        w22, [x2]
    ldrh        w23, [x0]

    mov         w10, w21
    mov         w20, w22
    gf_mul
    eor         w23, w23, w13
    add         x2, x2, #2
    strh        w23, [x0],#2

    add         w8, w8, #-1
    cbnz        w8, loop
    cbz         w8, loop1


loop1:
    add         x1, x1, #2
    ldrh        w21, [x1]
    add         x0, x0, #-126
    add         x2, x2, #-128
    mov         w8, #64
    add         w9, w9, #-1
    cbnz        w9, loop
    cbz         w9, loop2

////////
```

```asm
loop2:
    add         x0, x0, #124

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x36D
    gf_mul
    mov         w24, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0xB48
    gf_mul
    mov         w25, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x6F5
    gf_mul
    mov         w26, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x175
    gf_mul
    mov         w27, w13

    add         x0, x0, #-110

    ldrh        w23, [x0]
    eor         w23, w23, w24
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w25
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w26
    strh        w23, [x0], #-10

    ldrh        w23, [x0]
    eor         w23, w23, w27
    strh        w23, [x0],#2
    add         w12, w12, #-1
    cbnz        w12, loop2
```

Multiplication



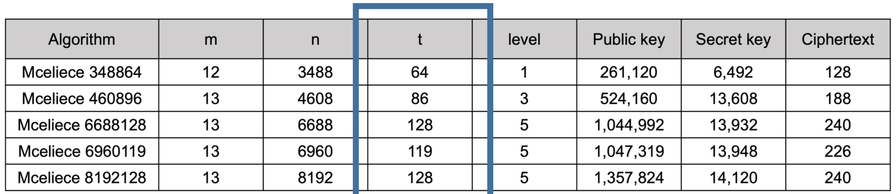
```c
/* input: in0, in1 in GF((2^m)^t)*/
/* m : 12, t : 64*/
/* output: out = in0*in1 */
/*secretkey sk_gen에서만 필요한 함수*/
void PQCLEAN_MCELIECE348864_CLEAN_GF_mul(gf *out, const gf *in0, const gf *in1) {
    int i, j;

    gf prod[ SYS_T * 2 - 1 ];

    for (i = 0; i < SYS_T * 2 - 1; i++) {
        prod[i] = 0;
    }

    for (i = 0; i < SYS_T; i++) {
        for (j = 0; j < SYS_T; j++) {
            prod[i + j] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(in0[i], in1[j]);
        }
    }

    //

    for (i = (SYS_T - 1) * 2; i >= SYS_T; i--) {
        prod[i - SYS_T +  9] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  877);
        //877 -> 0x36D

        prod[i - SYS_T +  7] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 2888);

        //2888 ->0xB48
        prod[i - SYS_T +  5] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 1781);

        //0x6F5
        prod[i - SYS_T +  0] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  373);

        //0x175
    }

    for (i = 0; i < SYS_T; i++) {
        out[i] = prod[i];
    }
}
```



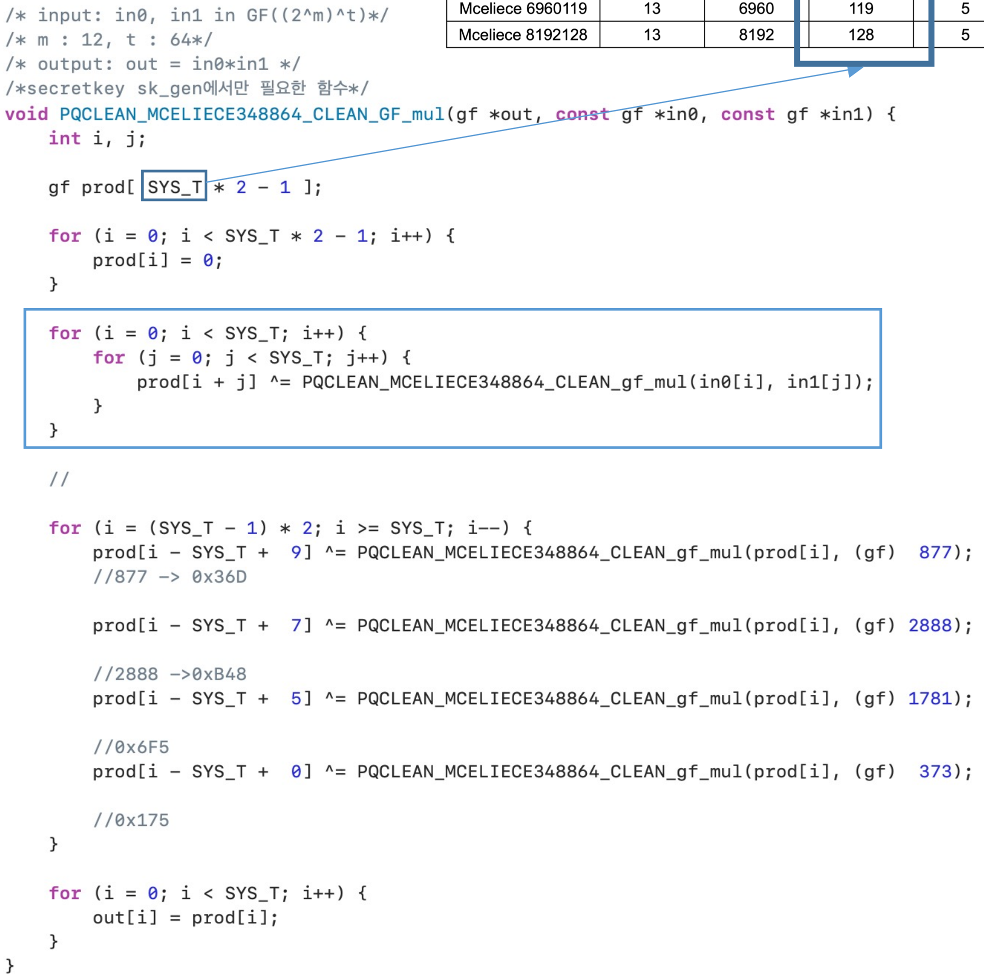
```asm
GF_mul:
_GF_mul:

    mov         w3, #1
    mov         w8, #64
    mov         w9, #64  //index i
    mov         w12, #63
    mov         w11, #128

//첫번째 매개변수(out) 초기화
loop0:
    mov         w23, #0
    strh        w23, [x0], #2

    add         w11, w11, #-1
    cbnz        w11, loop0

///////////////////
    add         x0, x0, #-256
    ldrh        w21, [x1]


///////////
loop:
    ldrh        w22, [x2]
    ldrh        w23, [x0]

    mov         w10, w21
    mov         w20, w22
    gf_mul
    eor         w23, w23, w13
    add         x2, x2, #2
    strh        w23, [x0],#2

    add         w8, w8, #-1
    cbnz        w8, loop
    cbz         w8, loop1


loop1:
    add         x1, x1, #2
    ldrh        w21, [x1]
    add         x0, x0, #-126
    add         x2, x2, #-128
    mov         w8, #64
    add         w9, w9, #-1
    cbnz        w9, loop
    cbz         w9, loop2

////////
```

```asm
loop2:
    add         x0, x0, #124

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x36D
    gf_mul
    mov         w24, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0xB48
    gf_mul
    mov         w25, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x6F5
    gf_mul
    mov         w26, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x175
    gf_mul
    mov         w27, w13

    add         x0, x0, #-110

    ldrh        w23, [x0]
    eor         w23, w23, w24
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w25
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w26
    strh        w23, [x0], #-10

    ldrh        w23, [x0]
    eor         w23, w23, w27
    strh        w23, [x0],#2
    add         w12, w12, #-1
    cbnz        w12, loop2
```

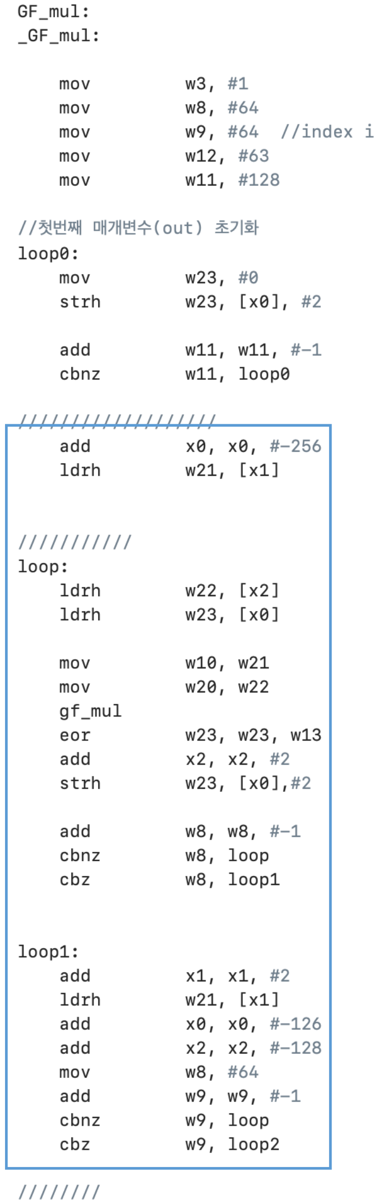
Multiplication



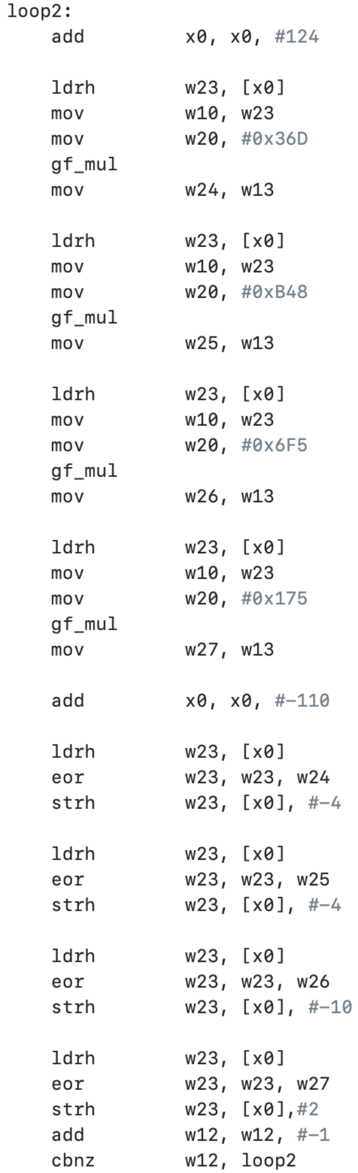
Algorithm	m	n	t	level	Public key	Secret key	Ciphertext
Mceliece 348864	12	3488	64	1	261,120	6,492	128
Mceliece 460896	13	4608	86	3	524,160	13,608	188
Mceliece 6688128	13	6688	128	5	1,044,992	13,932	240
Mceliece 6960119	13	6960	119	5	1,047,319	13,948	226
Mceliece 8192128	13	8192	128	5	1,357,824	14,120	240



```c
/* input: in0, in1 in GF((2^m)^t)*/
/* m : 12, t : 64*/
/* output: out = in0*in1 */
/*secretkey sk_gen에서만 필요한 함수*/
void PQCLEAN_MCELIECE348864_CLEAN_GF_mul(gf *out, const gf *in0, const gf *in1) {
    int i, j;

    gf prod[ SYS_T * 2 - 1 ];

    for (i = 0; i < SYS_T * 2 - 1; i++) {
        prod[i] = 0;
    }

    for (i = 0; i < SYS_T; i++) {
        for (j = 0; j < SYS_T; j++) {
            prod[i + j] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(in0[i], in1[j]);
        }
    }

    //

    for (i = (SYS_T - 1) * 2; i >= SYS_T; i--) {
        prod[i - SYS_T +  9] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  877);
        //877 -> 0x36D

        prod[i - SYS_T +  7] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 2888);

        //2888 ->0xB48
        prod[i - SYS_T +  5] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 1781);

        //0x6F5
        prod[i - SYS_T +  0] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  373);

        //0x175
    }

    for (i = 0; i < SYS_T; i++) {
        out[i] = prod[i];
    }
}
```

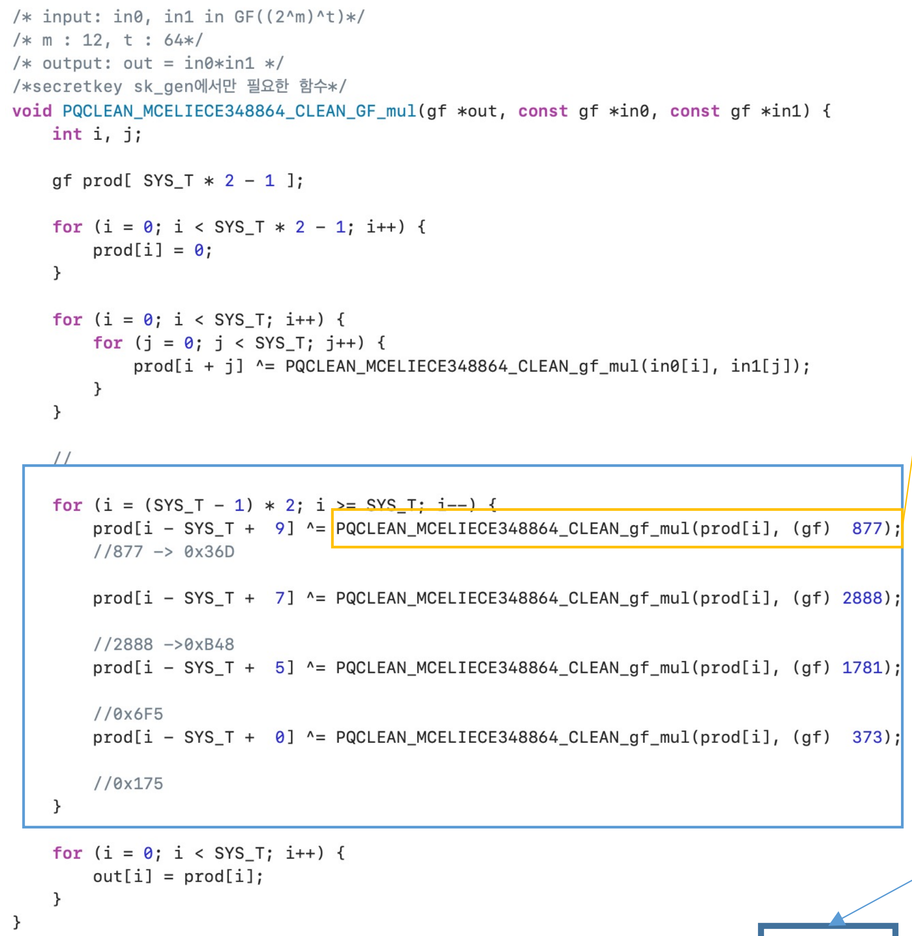


```asm
GF_mul:
_GF_mul:

    mov         w3, #1
    mov         w8, #64
    mov         w9, #64  //index i
    mov         w12, #63
    mov         w11, #128

//첫번째 매개변수(out) 초기화
loop0:
    mov         w23, #0
    strh        w23, [x0], #2

    add         w11, w11, #-1
    cbnz        w11, loop0

///////////////////
    add         x0, x0, #-256
    ldrh        w21, [x1]


///////////
loop:
    ldrh        w22, [x2]
    ldrh        w23, [x0]

    mov         w10, w21
    mov         w20, w22
    gf_mul
    eor         w23, w23, w13
    add         x2, x2, #2
    strh        w23, [x0],#2

    add         w8, w8, #-1
    cbnz        w8, loop
    cbz         w8, loop1


loop1:
    add         x1, x1, #2
    ldrh        w21, [x1]
    add         x0, x0, #-126
    add         x2, x2, #-128
    mov         w8, #64
    add         w9, w9, #-1
    cbnz        w9, loop
    cbz         w9, loop2

////////
```

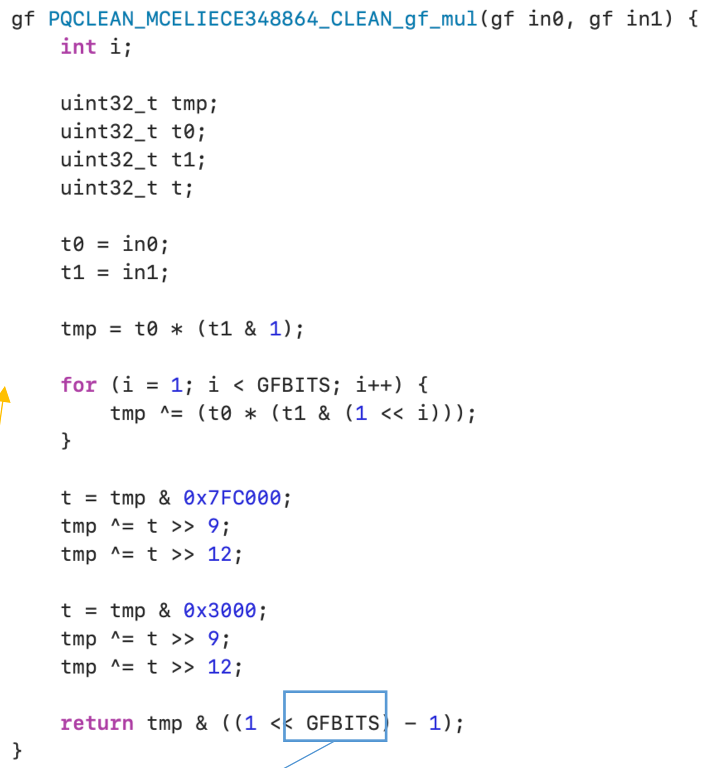


```asm
loop2:
    add         x0, x0, #124

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x36D
    gf_mul
    mov         w24, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0xB48
    gf_mul
    mov         w25, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x6F5
    gf_mul
    mov         w26, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x175
    gf_mul
    mov         w27, w13

    add         x0, x0, #-110

    ldrh        w23, [x0]
    eor         w23, w23, w24
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w25
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w26
    strh        w23, [x0], #-10

    ldrh        w23, [x0]
    eor         w23, w23, w27
    strh        w23, [x0],#2
    add         w12, w12, #-1
    cbnz        w12, loop2
```

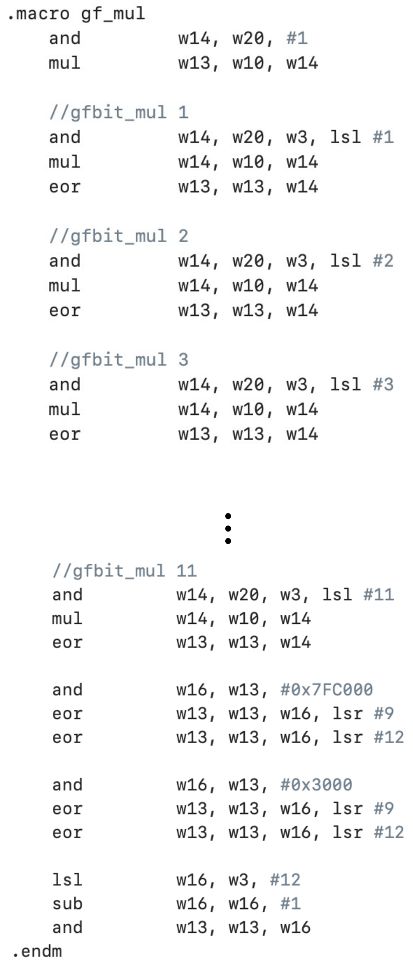
Multiplication



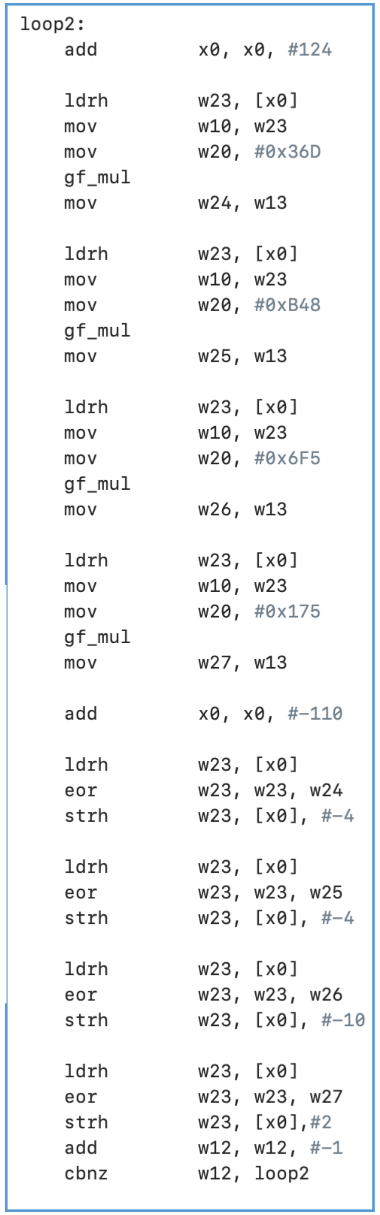
```c
/* input: in0, in1 in GF((2^m)^t)*/
/* m : 12, t : 64*/
/* output: out = in0*in1 */
/*secretkey sk_gen에서만 필요한 함수*/
void PQCLEAN_MCELIECE348864_CLEAN_GF_mul(gf *out, const gf *in0, const gf *in1) {
    int i, j;

    gf prod[ SYS_T * 2 - 1 ];

    for (i = 0; i < SYS_T * 2 - 1; i++) {
        prod[i] = 0;
    }

    for (i = 0; i < SYS_T; i++) {
        for (j = 0; j < SYS_T; j++) {
            prod[i + j] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(in0[i], in1[j]);
        }
    }

    //

    for (i = (SYS_T - 1) * 2; i >= SYS_T; i--) {
        prod[i - SYS_T +  9] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  877);
        //877 -> 0x36D

        prod[i - SYS_T +  7] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 2888);

        //2888 ->0xB48
        prod[i - SYS_T +  5] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf) 1781);

        //0x6F5
        prod[i - SYS_T +  0] ^= PQCLEAN_MCELIECE348864_CLEAN_gf_mul(prod[i], (gf)  373);

        //0x175
    }

    for (i = 0; i < SYS_T; i++) {
        out[i] = prod[i];
    }
}
```



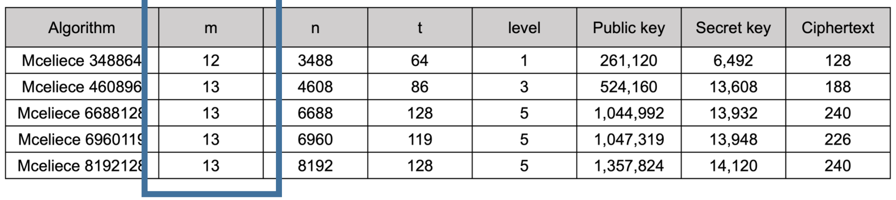
```c
gf PQCLEAN_MCELIECE348864_CLEAN_gf_mul(gf in0, gf in1) {
    int i;

    uint32_t tmp;
    uint32_t t0;
    uint32_t t1;
    uint32_t t;

    t0 = in0;
    t1 = in1;

    tmp = t0 * (t1 & 1);

    for (i = 1; i < GFBITS; i++) {
        tmp ^= (t0 * (t1 & (1 << i)));
    }

    t = tmp & 0x7FC000;
    tmp ^= t >> 9;
    tmp ^= t >> 12;

    t = tmp & 0x3000;
    tmp ^= t >> 9;
    tmp ^= t >> 12;

    return tmp & ((1 << GFBITS) - 1);
}
```



```
.macro gf_mul
    and         w14, w20, #1
    mul         w13, w10, w14

    //gfbit_mul 1
    and         w14, w20, w3, lsl #1
    mul         w14, w10, w14
    eor         w13, w13, w14

    //gfbit_mul 2
    and         w14, w20, w3, lsl #2
    mul         w14, w10, w14
    eor         w13, w13, w14

    //gfbit_mul 3
    and         w14, w20, w3, lsl #3
    mul         w14, w10, w14
    eor         w13, w13, w14

                    ⋮

    //gfbit_mul 11
    and         w14, w20, w3, lsl #11
    mul         w14, w10, w14
    eor         w13, w13, w14

    and         w16, w13, #0x7FC000
    eor         w13, w13, w16, lsr #9
    eor         w13, w13, w16, lsr #12

    and         w16, w13, #0x3000
    eor         w13, w13, w16, lsr #9
    eor         w13, w13, w16, lsr #12

    lsl         w16, w3, #12
    sub         w16, w16, #1
    and         w13, w13, w16
.endm
```



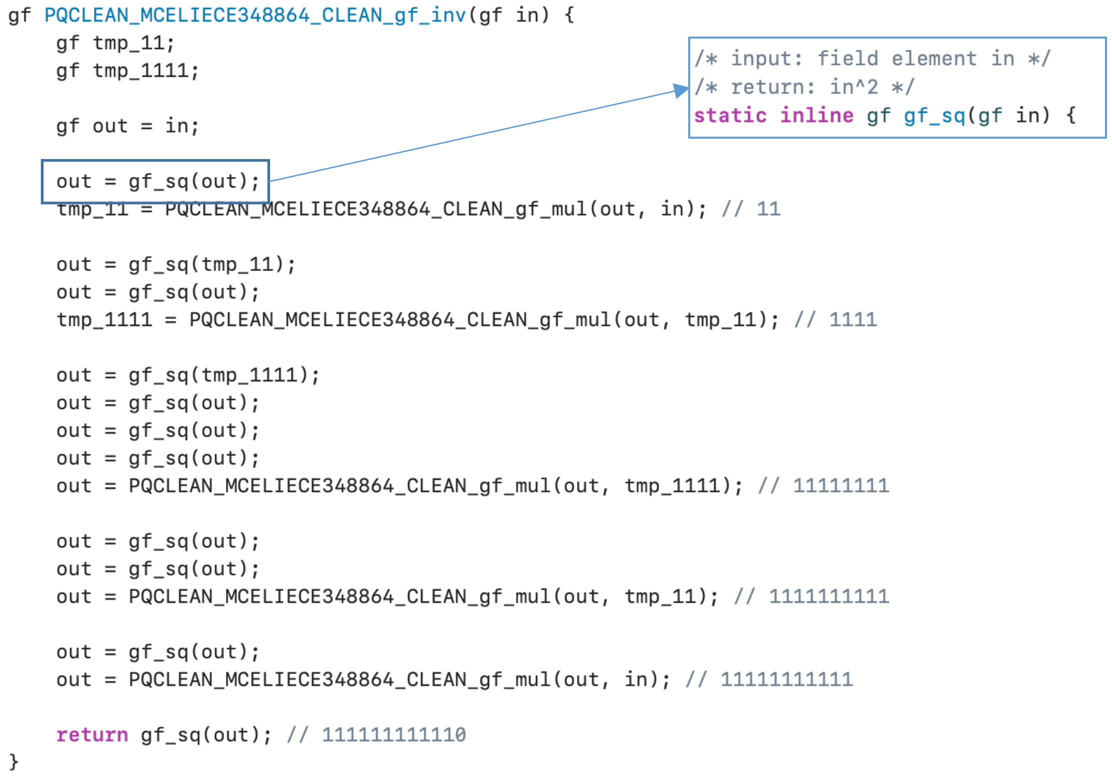
```
loop2:
    add         x0, x0, #124

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x36D
    gf_mul
    mov         w24, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0xB48
    gf_mul
    mov         w25, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x6F5
    gf_mul
    mov         w26, w13

    ldrh        w23, [x0]
    mov         w10, w23
    mov         w20, #0x175
    gf_mul
    mov         w27, w13

    add         x0, x0, #-110

    ldrh        w23, [x0]
    eor         w23, w23, w24
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w25
    strh        w23, [x0], #-4

    ldrh        w23, [x0]
    eor         w23, w23, w26
    strh        w23, [x0], #-10

    ldrh        w23, [x0]
    eor         w23, w23, w27
    strh        w23, [x0],#2
    add         w12, w12, #-1
    cbnz        w12, loop2
```



Algorithm	m	n	t	level	Public key	Secret key	Ciphertext
Mceliece 348864	12	3488	64	1	261,120	6,492	128
Mceliece 460896	13	4608	86	3	524,160	13,608	188
Mceliece 6688128	13	6688	128	5	1,044,992	13,932	240
Mceliece 6960119	13	6960	119	5	1,047,319	13,948	226
Mceliece 8192128	13	8192	128	5	1,357,824	14,120	240

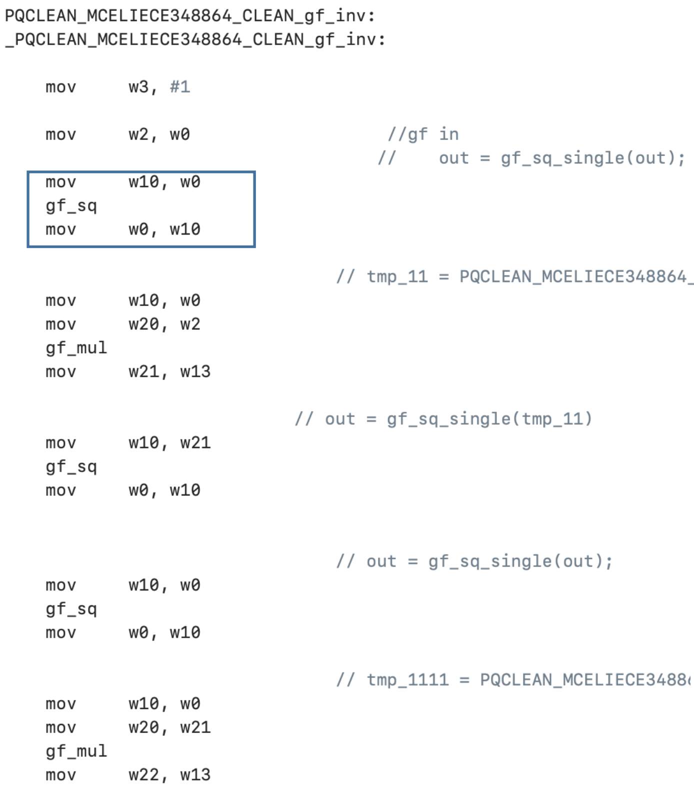
Inversion

- Inversion 연산
  - Squaring 연산과 Multiplication 연산

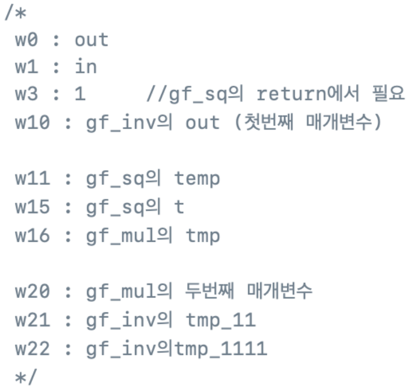


```c
gf PQCLEAN_MCELIECE348864_CLEAN_gf_inv(gf in) {
    gf tmp_11;
    gf tmp_1111;

    gf out = in;

    out = gf_sq(out);
    tmp_11 = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, in); // 11

    out = gf_sq(tmp_11);
    out = gf_sq(out);
    tmp_1111 = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, tmp_11); // 1111

    out = gf_sq(tmp_1111);
    out = gf_sq(out);
    out = gf_sq(out);
    out = gf_sq(out);
    out = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, tmp_1111); // 11111111

    out = gf_sq(out);
    out = gf_sq(out);
    out = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, tmp_11); // 1111111111

    out = gf_sq(out);
    out = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, in); // 11111111111

    return gf_sq(out); // 111111111110
}
```

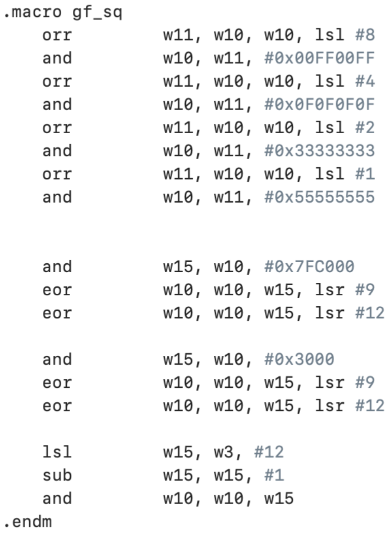
```c
/* input: field element in */
/* return: in^2 */
static inline gf gf_sq(gf in) {
```



```asm
PQCLEAN_MCELIECE348864_CLEAN_gf_inv:
_PQCLEAN_MCELIECE348864_CLEAN_gf_inv:

    mov     w3, #1

    mov     w2, w0                  //gf in
                                   //    out = gf_sq_single(out);
    mov     w10, w0
    gf_sq
    mov     w0, w10

                                   // tmp_11 = PQCLEAN_MCELIECE348864_CLEAN_gf_mul(out, in);
    mov     w10, w0
    mov     w20, w2
    gf_mul
    mov     w21, w13
                                   // out = gf_sq_single(tmp_11)

    mov     w10, w21
    gf_sq
    mov     w0, w10

                                   // out = gf_sq_single(out);
    mov     w10, w0
    gf_sq
    mov     w0, w10

                                   // tmp_1111 = PQCLEAN_MCELIECE3488
    mov     w10, w0
    mov     w20, w21
    gf_mul
    mov     w22, w13
```



```
/*
 w0 : out
 w1 : in
 w3 : 1      //gf_sq의 return에서 필요
 w10 : gf_inv의 out (첫번째 매개변수)

 w11 : gf_sq의 temp
 w15 : gf_sq의 t
 w16 : gf_mul의 tmp

 w20 : gf_mul의 두번째 매개변수
 w21 : gf_inv의 tmp_11
 w22 : gf_inv의tmp_1111
 */
```



```asm
.macro gf_sq
    orr         w11, w10, w10, lsl #8
    and         w10, w11, #0x00FF00FF
    orr         w11, w10, w10, lsl #4
    and         w10, w11, #0x0F0F0F0F
    orr         w11, w10, w10, lsl #2
    and         w10, w11, #0x33333333
    orr         w11, w10, w10, lsl #1
    and         w10, w11, #0x55555555

    and         w15, w10, #0x7FC000
    eor         w10, w10, w15, lsr #9
    eor         w10, w10, w15, lsr #12

    and         w15, w10, #0x3000
    eor         w10, w10, w15, lsr #9
    eor         w10, w10, w15, lsr #12

    lsl         w15, w3, #12
    sub         w15, w15, #1
    and         w10, w10, w15
.endm
```

Q & A