암호인재 인력양성 1차 과제

정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

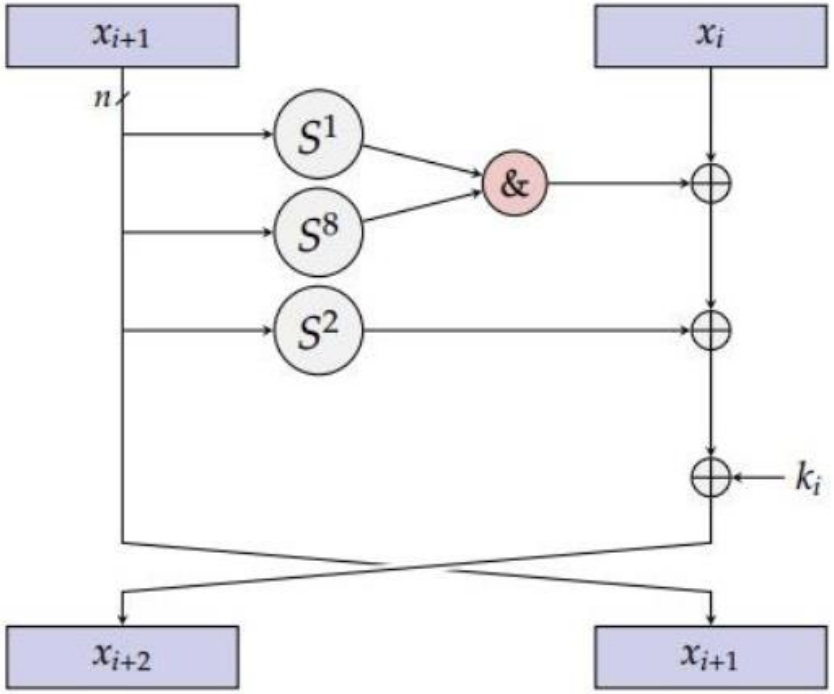
CryptoCraft LAB
https://crypto.modoo.at

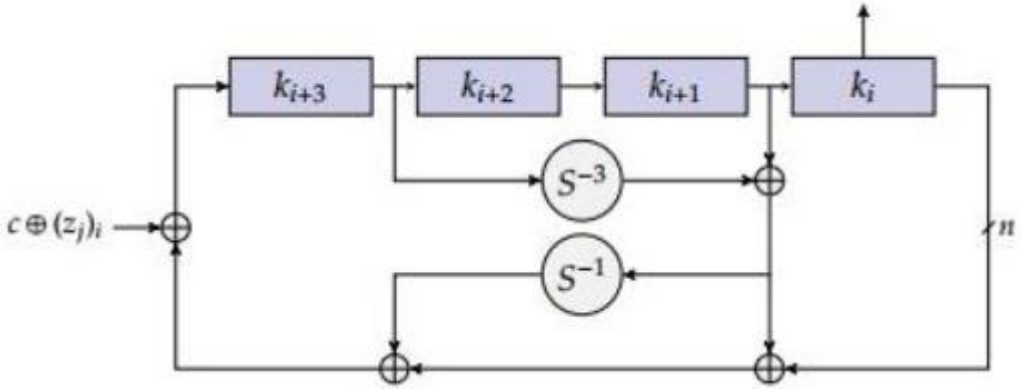
Contents

CryptoCraft LAB

환경설정

- Simon 암호 사용





환경설정

- Simon 암호 사용

블록 크기	키 크기	라운드
32	64	32
48	72	36
	96	36
64	96	42
	128	44
96	**96**	**52**
	144	54
128	128	68
	192	69
	256	72

환경설정

- Simon-96/96

- 키: 01 02 03 04 05 06 07 08 09 0a 0b 0c
- 평문: fe ed de ad be ef ca fe be d1 23 45
- 암호문: 6b d3 6e 75 6d 7b a8 2b a7 b8 7b e7

환경설정

- 평문을 다음 규칙과 생성하고, 키 k로 암호화
  - k는 모두 동일
- n=15000 블록 평문, $M_{ECB} = m_0, m_1, m_2, \ldots, m_{n-1}$ 생성
- M_{CBC} 첫 블록은 m_1, 백번째 블록은 m_{101}, 나머지는 무작위
- M_{CTR} 첫 블록은 m_2, 백번째 블록은 m_{102}, 나머지는 무작위
- M_{CFB} 첫 블록은 m_3, 백번째 블록은 m_{103}, 나머지는 무작위
- M_{OFB} 첫 블록은 m_4, 백번째 블록은 m_{104}, 나머지는 무작위
- 모든 평문 길이는 동일
- 모든 평문 M_*를 *에 해당하는 운용모드로 암호화

환경설정

- SIMON-96/96은 48bit 단위로 동작
  - 96bit 블록을 두개로 나누기 때문
- C에는 48bit를 제공하는 자료형이 존재하지 않음
- **64bit 자료형의 앞 16bit를 0으로 마스킹하여 구현**

```c
void enc(u64 *pt, u64 *key, u64 *ct)
{
    const int Round = 52;

    u64 Y = pt[0];
    u64 X = pt[1];

    for(int i = 0 ; i < Round ; i++)
    {
        u64 temp = (ROL1(X) & ROL8(X)) ^ Y ^ ROL2(X) ^ key[i];

        Y = X;

        X = temp;
    }

    memcpy(&ct[0], &Y, sizeof(u64));
    memcpy(&ct[1], &X, sizeof(u64));
}

void dec(u64 *ct, u64 *key, u64 *pt)
{
    const int Round = 52;

    u64 X = ct[0];
    u64 Y = ct[1];

    for(int i = 0 ; i < Round ; i++)
    {
        u64 temp = (ROL1(X) & ROL8(X)) ^ Y ^ ROL2(X) ^ key[Round - i - 1];

        Y = X;

        X = temp;
    }

    memcpy(&pt[0], &X, sizeof(u64));
    memcpy(&pt[1], &Y, sizeof(u64));
}
```

```c
void key_schedule(u64 *sub)
{
    const u64 C = 0x0000FFFFFFFFFFFC;
    const u64 Z = 0b00110011011010011111100010000101000110010010110000001110111110101;

    // skip first, second

    for(int i = 2 ; i < 52 ; i++)
    {
        u64 temp1 = ROR3(sub[i-1]);
        u64 temp2 = ROR1(temp1);

        temp1 ^= sub[i-2];
        temp1 ^= temp2;
        temp2 = C ^ ((Z >> (i - 2) % 62) & 1);
        temp1 ^= temp2;

        memcpy(&sub[i], &temp1, sizeof(u64));
    }
}
```

```c
u64 sub_key[52] = {0x00000708090a0b0c, 0x0000010203040506, 0,};
u64 plaintext[2] = {0x0000cafebed12345, 0x0000feeddeadbeef};
```

키: 01 02 03 04 05 06 07 08 09 0a 0b 0c

평문: fe ed de ad be ef ca fe be d1 23 45

암호문: 6b d3 6e 75 6d 7b a8 2b a7 b8 7b e7

환경설정

```
Encrypting SIMON 96-96

KEY:    00000708090a0b0c 0000010203040506
PT:     0000cafebed12345 0000feeddeadbeef
CT:     0000a82ba7b87be7 00006bd36e756d7b
Decrypting SIMON 96-96

KEY:    00000708090a0b0c 0000010203040506
CT:     0000a82ba7b87be7 00006bd36e756d7b
DEC:    0000cafebed12345 0000feeddeadbeef
```

키:	01 02 03 04 05 06 07 08 09 0a 0b 0c
평문:	fe ed de ad be ef ca fe be d1 23 45
암호문:	6b d3 6e 75 6d 7b a8 2b a7 b8 7b e7

문제 1

- 키의 *를 복구하여 k를 완성

- 00 00 00 00 00 00 00 00 00 00 00 00
  -> 41 46 f6 54 e7 08 31 92 d2 47 e 58
- 38 97 2b 17 1e ca 97 74 1f bd fa 90
  -> 45 ba 59 a4 ae fa 3d f0 60 f5 5b fc
- 키 k: ** 49 4d ** 6e *9 *6 *5 *f 5* 49 5*
  - **키의 40bits를 완성시켜야 함**

문제 1

- 키 k: ** 49 4d ** 6e *9 *6 *5 *f 5* 49 5*

- k를 사용한 암호화 결과만 존재
- 암호화 결과 값이 맞는지 전수 조사를 시행
- 공개된 부분은 고정시킨 채, ***부분만을 1씩 증가**시키며 조사

문제

```
void attack(u64 sub_key[32][52], u64 plaintext[32][2], u64 ciphertext[32][2], u64 *ans)
{
    int aa, bb, cc, a, b, c, d, e;

    int shift0[] = {0, 16, 28, 36, 44};
    int mshift0[] = {4, 20, 32, 40, 48};
    int shift1[] = {4, 16, 40};
    int mshift1[] = {8, 24, 44};


    for(aa = 0 ; aa < 4 ; aa++)
    {
        for(bb = 0 ; bb < 256 ; bb++)   // ** (µÑ)
        {
            for(cc = 0 ; cc < 16 ; cc++)  // *9
            {
                for(a = 0 ; a < 16 ; a++)              // *6
                {
                    for(b = 0 ; b < 16 ; b++)              // *5
                    {
                        for(c = 0 ; c < 16 ; c++)              // *f
                        {
                            for(d = 0 ; d < 16 ; d++)              // 5*
                            {
                                for(e = 0 ; e < 16 ; e++)                      // 5*
                                {
                                    sub_key[omp_get_thread_num()][0] += ((u64)1 << shift0[0]);
                                    key_schedule(sub_key[omp_get_thread_num()]);
                                    enc(plaintext[omp_get_thread_num()], sub_key[omp_get_thread_num()], ciphertext[omp_get_threa

                                    if(ciphertext[omp_get_thread_num()][0] == ans[0] || ciphertext[omp_get_thread_num()][0] == a
                                    {
                                        printf("THREAD %d \n", omp_get_thread_num());

                                        printf("KEY: %016llx %016llx \n", sub_key[omp_get_thread_num()][0], sub_key[omp_get_

                                        printf("PT: %016llx %016llx \n", plaintext[omp_get_thread_num()][0], plaintext[omp_g

                                        printf("CT: %016llx %016llx \n", ciphertext[omp_get_thread_num()][0], ciphertext[omp

                                        printf("ANS: %016llx %016llx \n", ans[0], ans[1]);
                                    }
```

문제 1

```
                                        if(e == 15)
                                        {
                                                sub_key[omp_get_thread_num()][0] -= ((u64)1 << mshift0[0]);
                                                sub_key[omp_get_thread_num()][0] += ((u64)1 << shift0[1]);
                                        }
                                        //printf("THREAD: %d-%d \t %016llx %016llx \n", omp_get_thread_num(), a, sub_key[omp_get_thr
                                }
                                if(d == 15)
                                {
                                        sub_key[omp_get_thread_num()][0] -= ((u64)1 << mshift0[1]);
                                        sub_key[omp_get_thread_num()][0] += ((u64)1 << shift0[2]);
                                }
                        }
                        if(c == 15)
                        {
                                sub_key[omp_get_thread_num()][0] -= ((u64)1 << mshift0[2]);
                                sub_key[omp_get_thread_num()][0] += ((u64)1 << shift0[3]);
                        }
                }
                if(b == 15)
                {
                        sub_key[omp_get_thread_num()][0] -= ((u64)1 << mshift0[3]);
                        sub_key[omp_get_thread_num()][0] += ((u64)1 << shift0[4]);
                }
        }
        if(a == 15)
        {
                sub_key[omp_get_thread_num()][0] -= ((u64)1 << mshift0[4]);
                sub_key[omp_get_thread_num()][1] += ((u64)1 << shift1[0]);
        }
}
if(cc == 15)
{
        sub_key[omp_get_thread_num()][1] -= ((u64)1 << mshift1[0]);
        sub_key[omp_get_thread_num()][1] += ((u64)1 << shift1[1]);
}
}
if(bb == 255)
{
        sub_key[omp_get_thread_num()][1] -= ((u64)1 << mshift1[1]);
        sub_key[omp_get_thread_num()][1] += ((u64)1 << shift1[2]);
}
//printf("THREAD %d\treached %d bb loop\n", omp_get_thread_num(), bb);
}
printf("THREAD %d finished %d loop \n", omp_get_thread_num(), aa+1);
}
}
```

문제 1

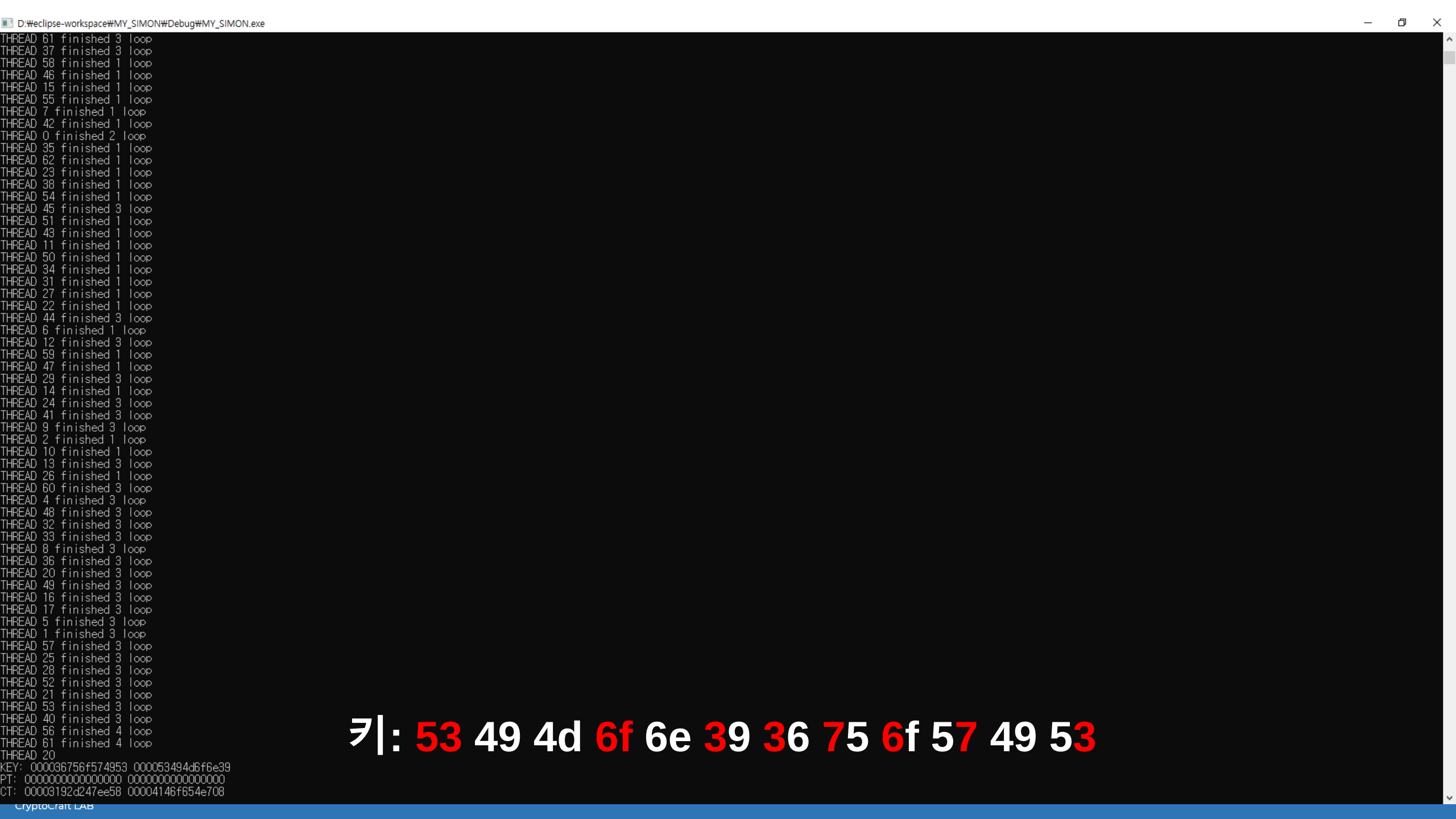
```
/*u64 sub_key[32][52] =  {{0x000006050f504950, 0x000000494d006e09, 0,},
                        {0x000006050f504950, 0x000008494d006e09, 0,},
                        {0x000006050f504950, 0x000010494d006e09, 0,},
                        {0x000006050f504950, 0x000018494d006e09, 0,},
                        {0x000006050f504950, 0x000020494d006e09, 0,},
                        {0x000006050f504950, 0x000028494d006e09, 0,},
                        {0x000006050f504950, 0x000030494d006e09, 0,},
                        {0x000006050f504950, 0x000038494d006e09, 0,},
                        {0x000006050f504950, 0x000040494d006e09, 0,},
                        {0x000006050f504950, 0x000048494d006e09, 0,},
                        {0x000006050f504950, 0x000050494d006e09, 0,},
                        {0x000006050f504950, 0x000058494d006e09, 0,},
                        {0x000006050f504950, 0x000060494d006e09, 0,},
                        {0x000006050f504950, 0x000068494d006e09, 0,},
                        {0x000006050f504950, 0x000070494d006e09, 0,},
                        {0x000006050f504950, 0x000078494d006e09, 0,},
                        {0x000006050f504950, 0x000080494d006e09, 0,},
                        {0x000006050f504950, 0x000088494d006e09, 0,},
                        {0x000006050f504950, 0x000090494d006e09, 0,},
                        {0x000006050f504950, 0x000098494d006e09, 0,},
                        {0x000006050f504950, 0x0000a0494d006e09, 0,},
                        {0x000006050f504950, 0x0000a8494d006e09, 0,},
                        {0x000006050f504950, 0x0000b0494d006e09, 0,},
                        {0x000006050f504950, 0x0000b8494d006e09, 0,},
                        {0x000006050f504950, 0x0000c0494d006e09, 0,},
                        {0x000006050f504950, 0x0000c8494d006e09, 0,},
                        {0x000006050f504950, 0x0000d0494d006e09, 0,},
                        {0x000006050f504950, 0x0000d8494d006e09, 0,},
                        {0x000006050f504950, 0x0000e0494d006e09, 0,},
                        {0x000006050f504950, 0x0000e8494d006e09, 0,},
                        {0x000006050f504950, 0x0000f0494d006e09, 0,},
                        {0x000006050f504950, 0x0000f8494d006e09, 0,}};
```

```
u64 plaintext[32][2] = {0,};
u64 ciphertext[32][2] = {0,};

printf("Finding SIMON 96-96 Key ... \n\n");

omp_set_num_threads(32);

    #pragma omp parallel
{
            #pragma omp for
    for(int tnum = 0 ; tnum < 32 ; tnum++)
            {
            attack(sub_key, plaintext, ciphertext, ans);
            }
}
```


D:₩eclipse-workspace₩MY_SIMON₩Debug₩MY_SIMON.exe
THREAD 61 finished 3 loop
THREAD 37 finished 3 loop
THREAD 58 finished 1 loop
THREAD 46 finished 1 loop
THREAD 15 finished 1 loop
THREAD 55 finished 1 loop
THREAD 7 finished 1 loop
THREAD 42 finished 1 loop
THREAD 0 finished 2 loop
THREAD 35 finished 1 loop
THREAD 62 finished 1 loop
THREAD 23 finished 1 loop
THREAD 38 finished 1 loop
THREAD 54 finished 1 loop
THREAD 45 finished 3 loop
THREAD 51 finished 1 loop
THREAD 43 finished 1 loop
THREAD 11 finished 1 loop
THREAD 50 finished 1 loop
THREAD 34 finished 1 loop
THREAD 31 finished 1 loop
THREAD 27 finished 1 loop
THREAD 22 finished 1 loop
THREAD 44 finished 3 loop
THREAD 6 finished 1 loop
THREAD 12 finished 3 loop
THREAD 59 finished 1 loop
THREAD 47 finished 1 loop
THREAD 29 finished 3 loop
THREAD 14 finished 1 loop
THREAD 24 finished 3 loop
THREAD 41 finished 3 loop
THREAD 9 finished 3 loop
THREAD 2 finished 1 loop
THREAD 10 finished 1 loop
THREAD 13 finished 3 loop
THREAD 26 finished 1 loop
THREAD 60 finished 3 loop
THREAD 4 finished 3 loop
THREAD 48 finished 3 loop
THREAD 32 finished 3 loop
THREAD 33 finished 3 loop
THREAD 8 finished 3 loop
THREAD 36 finished 3 loop
THREAD 20 finished 3 loop
THREAD 49 finished 3 loop
THREAD 16 finished 3 loop
THREAD 17 finished 3 loop
THREAD 5 finished 3 loop
THREAD 1 finished 3 loop
THREAD 57 finished 3 loop
THREAD 25 finished 3 loop
THREAD 28 finished 3 loop
THREAD 52 finished 3 loop
THREAD 21 finished 3 loop
THREAD 53 finished 3 loop
THREAD 40 finished 3 loop
THREAD 56 finished 4 loop
THREAD 61 finished 4 loop
THREAD 20
KEY: 000036756f574953 000053494d6f6e39
PT: 0000000000000000 0000000000000000
CT: 00003192d247ee58 0000146f654e708
키: 53 49 4d 6f 6e 39 36 75 6f 57 49 53
CryptoCraft LAB

문제 2

- 환경설정과 같이 암호문 파일이 5개 생성
- 각 암호문을 올바르게 복호화 하여 빈칸을 채우기

모드	암호문 파일명	IV	평문 마지막 블록
ECB		-	
CBC			
CTR			
CFB			
OFB			

data_01.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
7746877ba75f8100ae9493094b057a322bb8a13d6c5b2264549612e759a63ce3a9a67f85464ea05679af291401dc6bf7701984795cb76c97ba1684944
f1c42e1c8346c9df740d3cf53b2fbcacbe6481aa501debfafcd0559c2e78bff8c088eae09f974d591ea5ddff233aa02d0b740e90e7f3e2007b285e3d227b
dc623ead7d83e62eff3f0ea5ab1221d7b6ea44bbb32f8e30cd9c92c117caeaa551a5197bdd5e6eb0958e147a69357e6ac8eb10a2a5b4c02ddb14df4d8
67830be68737b00fbcbae4c731667824abc0d3774ccf7e5f1f39eda422ca32e59ea38612af3f76f1aa107f9b98dce561921a9c0dca922c682cd093ac58ec
a180c5bc4d38f0b8637ec9829d98b42c8565397446ab24bd5a46e53b0e5ea87240565425cfeb2b53d9cb48f17a63f53ebcab90aae7a17823564061521
9c1a6e1dd2fb1ceed138d43b85413ec75f76c02bf759c6dc66c8077ee533e37052b699f10cf53ea6421b43dd865079a841e2b8251e23400b75fc66e39
dde766f0763653e97dbf2c7fe1450cfd8d16597661db4f23da01c3fb9ec581de03c29b7b3111c110a140d5f498c8008659a1740057cd5aa3f6966fe941
def5ffea1dc05db981ba8f59aa53c7777a9c2a87c5c2092
b60482b7b470778a0dfb145e18070cf31480362a6a4ab8
9f942baf8e71d1c83252fea0898ff97ffd3b3c5cfcdb53c80
017334f18bd0780753dca1d4cadffefa7fc4595ed17ed92
5728cabdfb0fccd7f29e8c3477927ef0d0c684e2a5959ad
c13a7947de9f22cb0ee2fc958dc0ab26dbab2e4a0c1c7a
bbf532614baffce0174f4ef889e3a886045e41dcdec6356
52f01716c0e1d5c60bd17434224c27eccc7958a2df0299
3edddff654595fdcac3d41d6795391c1fe6754eebb5725
2201f3a972e2afee83ab3e3643ba3cbaa6742062b05dd5
fd2689f9f8856a776d0ca10553eb7183a984f2d5f3b0209
0758130375489ea51053d1197821fa3b5aabd928d5297
fc12ad8e2b08e6b3a2f950356dedb72377b0a6058fa059
13b53878d4e34ac210cbae8e7abc5572d14a52b4a9470
3f136264ea654bfa82652d87612ba5aaed54c1e1429228
0f0d59aaf242d64079a3814243b67f50aef788ca8709795
258472f29150c6fc9ecea532b307801b00d7f493392cae
01b86915cbfabf3a6aa94341cda4502cc9846e42e0418d
98abd3d0b9773dc7b47cfb15cb085a40f4e1270b2c5d1
2f75a86ed99e78d02b2c130c3f1e3f3c2d1d8573eeb81e
```

data_05.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
6dd58624886406c210cda91def589c711e743928702fecaefff7ac5688f0447e87b790d2c7e2b9244030b4d2a926530f2c7142a35b1c210ea799d6ef594
48c979cd00ae1406b7eced0ece489ca9b4249afeede4974c334c0fdada67cd4d080ca21ccf4052ae275d67b6a042835e61f14e9a8042c273bd8b30588
f2469ddef0186c4d41cd1ed24fc6fc93faf9bf72eae301472fc637dae9eac4094962d68cfe68eebef3773272c26b8febcf73801b6e5c0bb40fd3cd0cd8002
e6c7837e37c1d8118e35bd7f93e7f8e8a6446192ad59b4db9feff2193cb63681ce2f7e7417d43ba5928bac86d1fb7c2fc19c352d83e0bd69c6debc2d2c
1583411afb94d39daf157b3654ad38f3af7d75f859e96e52fbbbe9610eda76438695ea24752f3f156a8a964402b620b9380d90000190b4dfebebdcefd3
4b61023d730f62edbf6ea4b28a705e0ded143d828bd9f40412d68f2892559a224d1e3408c3cc26891068b479eb7b36db22fbcf4c438652fcedbd1a037
59e7a6eeda11b8b255aeda8b7bda00068554150b2d887c8e545f49b21f8ee8e9e4f903afc8e279373b0e2926d291ce799803e388b2ce77b0b400d330
e716545a553191229e7a677230aa0d8de3576071edacac12974c5b2fc661a89c9d2b075a374edfae36df483fa8e9326dfad6a43278a4b5c94be5cd736f
4de39320c11b0dc35985b80d58b625b40b198cb052dcf7f593bb73497e3e1a2729bc850a0c599ed8cd91c222c2d525225b19e90da007324ece184fb4
6a663317dee48725c3fdac07c8279e6fded44b92da8b3fb375263dc7b4f49921b4f17f85f21c53084f0042ce42e5feb636b7cea31ea84d5cad730157383
03c229f9549529d8646a17e8376e6b44a31ac8291b2920802b3ad71b0eedd96ae4570b84032a1d7568044488b9f96165c71163216f8fc15e3af1c8e51
ba6408a84736d51b3792ad4b6ba7fba41823f148533a4f27cea4a6c50dbb9698938bc8a43eb0d7c4d4fa83a247eeca5461576eda73a5f66d5382778fcf
4f9b471f5a57b0090d040a0e654be2f5a3d3b43d0141a97e2412717315c556910cbd4df586b2dc6f18e731dd43e2b354939c3aad43432466128f1b87c
66a691da2f203adcf4d16838a6a59ab602e368c5e94c4940030f6a83513c96639e84b25c10f92ef85bdce204b77a39340636934d559ebbc9259e81af40
0127b55ca696162b566a4640e3518ca8f853d038990817973e6a5fd66fb9e042390826380977144ea7141f679142c08855718c6ab77fce686acf7b2734
201b0adfb84e498f4da5d0a0fa1440cfc9e99a7416c8e96de915d80dcb1ea268443f2752cdba85365b5d19778a50e3d2819e79a4792907069b795a06e
48b8c274de725143128b2b8bb9567a5fd8d7a24f6163683cdca3ecd8ac724b1ba859cbd4f44feb4c17dd0c8a7c41a60b94681a10f39f528364ced356c
3af5c0849492281921d45ab92d3c184df1ee9643f32afb4c53143fdd74119302b3a1bce544dad567f735e3ea1029c6b92ca8d48647d12d7c48876c364
554c5f1b79caf85fad795fbfb920d3b862b140089f88209f214c1c04212dfb86102efa4b6fae36c0bd74b51abd3025347b1655a483ccc5895365f6d9820
492bd79846ea84b05b456e76e6583f30ada5f4d18dbf942e7ba5c58371e20613f50210e20cd83739c8d3845dded521cf8d820a6d18ff12b161d6c9610
d13200bf7730c737278107e2336587a9bb49a5abb208df83d6c17fed17e39add20690a0a0325af4d2d530ff6c4747b17ade8a54fb791148cf9b1172b51
3a718a8dbca71b055a3163006769944a11fa016c11c9c4f918b2fe811131d1a8e52113539240474bf77eadf1038433d1b8d81bbe5bdcae5b42bb1299c
9e690fd80dc7bb03de488749b863eac5d02406d2989aade2f9ba38455918a767616ab26aa19a7d1498d5361cbab5129e56262b61cab09988475c9184
a4b50e39bc25ba2e9b47c36eb86afa06e2294d3662d0ab8eba04c68f4b4f7ac95fdc82ae3559819cb73953fd447bc50d18dc39f95184db08b135ab3b6
83fc11ee9cb8821e2a4adfba9e27493d751ca3fe40ee44efb6bf071260845f1ee129e574fe4f934911fb9ed4f4e6cd2b661e2308266604ff8035b01dadeb
23e5499270027b0161138db32e33050d7f7fc8b701865088b22ce21969a2a881484cc6acdd443368e5536d041cb4a8a75824f1eb777749330adea39b
4a693fe098ec990e92ba0d0a5802fa56ad755777e2b4e801d9fda65fecb34f2b19b7898a5c242be53201d81fba64790b1cba5c9682224acf120977f80f2
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

CryptoCraft LAB

문제 2

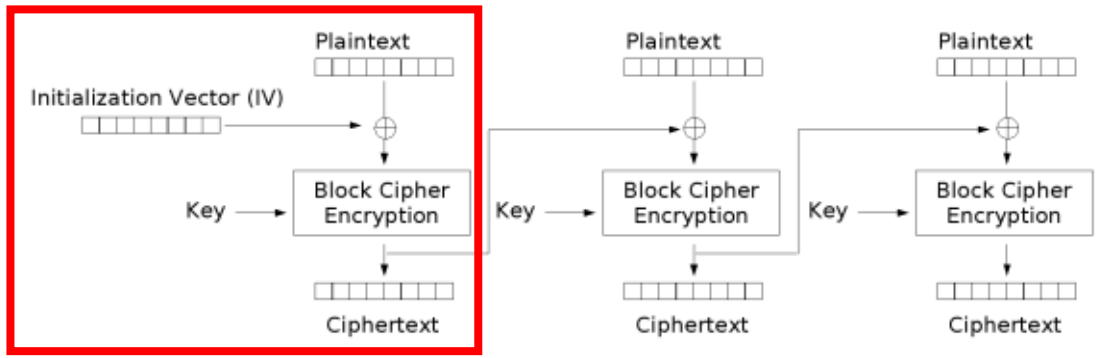
- 평문을 다음 규칙과 생성하고, 키 k로 암호화
  - k는 모두 동일
- n=15000 블록 평문, $M_{ECB} = m_0, \mathbf{m_1}, m_2, \ldots, m_{n-1}$ 생성
- M_{CBC} **첫 블록은 $\mathbf{m_1}$**, 백 번째 블록은 m_{101}, 나머지는 무작위
- 하지만 CBC 모드의 첫 블록을 복호화 하는 데는 IV 필요
- 따라서 CBC 모드 100번째와 ECB 모드 102번째를 비교
- 5개 파일의 102번째 블록을 ECB 모드로 복호화
- 5개 파일의 100번째 블록을 CBC 모드로 복호화

문제 2

ECB 모드 복호화				
1번파일				
m101	efd253ef257b	41643adc0761	02907e79373f	dd6a70a2b475
2번파일				
m101	59794b60da89	b96d0f9d7e88	da326b99b4cc	d9db043b9cc5
3번파일				
m101	3e4348d9b584	419c67faee76	0b288451c49b	ab4358361bbc
4번파일				
m101	72aa9a7587d4	d721ced8fc9b	13705fb02050	1c6da89afe01
5번파일				
m101	adc52a69a722	a19b5ebf35df	23b87828c19c	e674a4ef6480
CBC 모드 복호화 (CBC M99 = ECB M101)				
1번파일				
m98	a0b8cd936d6a	452ac5d476bb	-	-
m99	4471a94efac7	9c4dba98e76d	e7bf892e0205	3e5a095a0e70
2번파일				
m98	1f0844191f3e	25b58a200caa	-	-
m99	e35dbf9e899c	14337c12e148	13705fb02050	1c6da89afe01
3번파일				
m98	65a6bb46ff31	a4d922e83448	-	-
m99	c46cbe926a2d	9b4fbaaff907	eea4ce8d912e	3b4dcbac64f3
4번파일				
m98	679b7279bc5f	b988420e66b6	-	-
m99	4491942abfae	98d776088511	a3f0d1a4efa2	135738ec856c
5번파일				
m98	01c8388665c4	594116c8093a	-	-
m99	292635aea50b	ed00f77f3394	f14f6f8b5237	27d9b86d95dc

문제 2

- IV 복구를 위해서는 암호문을 복호화 한 다음,
  평문을 XOR해주는 것으로 획득 가능
- 암호문: CBC 파일의 첫 번째 블록
- 평문: ECB 파일의 두 번째 블록



Cipher Block Chaining (CBC) mode encryption

IV 복구				
E C B m1	c9518c78c70d	fe147bb68af8	8dd3027c714a (PT)	bc6663f5c6fc (PT)
C B C m0	3b0cb4683080	06a495bf29ad	24cafee9438d	946a2de58e1f

문제 2

- n=15000 블록 평문, $M_{ECB} = m_0, \mathbf{m_1}, m_2, \ldots, m_{n-1}$ 생성
- M_{CFB} 첫 블록은 m_3, **백 번째 블록은 m_{103}**, 나머지는 무작위
- CFB의 IV를 모르므로, 100번째 블록과 ECB 104번째 블록을 비교

문제 2

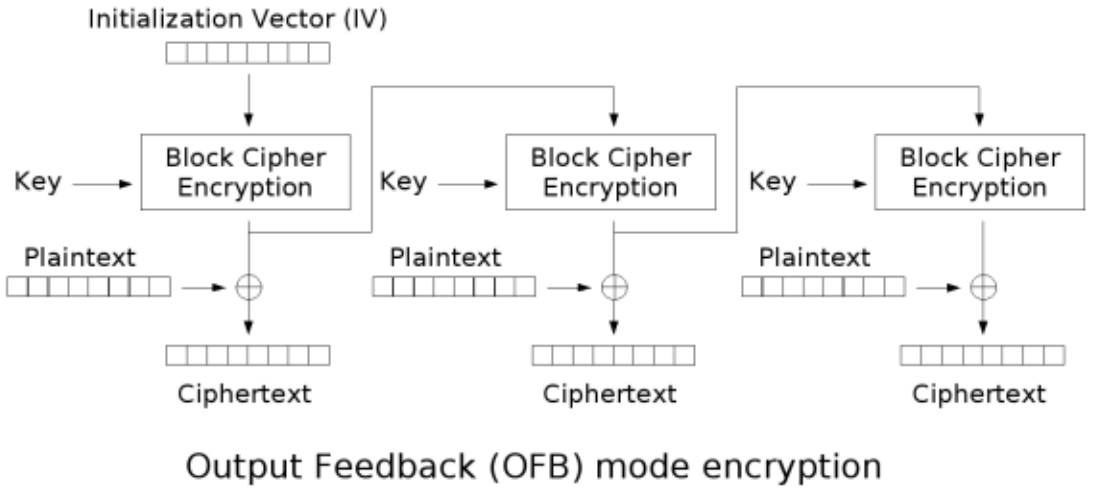
ECB 모드 복호화 - 4번파일				
m103	166eaa83e0ef	0cfff09ae7cb	3c6e5f41d01b	ca0f84ebe0c4
CFB 모드 복호화 (CFB M99 = ECB M103)				
1번파일				
m98	a0b8cd936d6a	452ac5d476bb	-	-
m99	4471a94efac7	9c4dba98e76d	3c6e5f41d01b	ca0f84ebe0c4
3번파일				
m98	65a6bb46ff31	a4d922e83448	-	-
m99	c46cbe926a2d	9b4fbaaff907	901ee0e4ef9f	2bfe6d6bb8c5
5번파일				
m98	01c8388665c4	594116c8093a	-	-
m99	292635aea50b	ed00f77f3394	db6e1551807a	23add47dec28
IV 복구				
E C B m3	e85d7a985001	ec05b34d9377	e182190104dc (PT)	14c04263584c (PT)
C F B m0	7746877ba75f	8100ae949309	7379e6d8a6b8	d5b5074352cb

문제 2

- n=15000 블록 평문, $M_{ECB} = m_0, m_1, m_2, \ldots, m_{n-1}$ 생성
- M_{OFB} 첫 블록은 m_4, 백 번째 블록은 m_{104}, 나머지는 무작위
- OFB는 IV가 계속해서 암호화 되는 특성이 존재
- **IV 획득 과정 = 파일 찾기 과정**
- 남은 두 개의 파일에서

  백 번째 블록을 100회 복호화

  -> IV 후보 값 획득



Output Feedback (OFB) mode encryption

문제 2

- ECB 다섯 번째 평문과 IV 후보를 사용한 암호화 진행
- 해당 암호 값이 일치하는 암호문 파일 확인

ECB 모드 복호화 - 4번파일				
m104	c2368b7fae5c	ae1fcd7f9a87	3cac28bcf8f6	e50587900bf0
OFB 모드 복호화 (OFB M99 = ECB M104)				
IV 복구				
3번파일				
m99	c46cbe926a2d	9b4fbaaff907	0ca10e72f5c3	23eb75310897
5번파일				
m99	292635aea50b	ed00f77f3394	79d0cb629d8a	8400c0cb366a
E C B m4	b56d71692a1f	5e4c3abc88f2	7d570b0f233b	7fe06f4b1868
O F B m0 _f3	3d5441707b74	afc5ca4d4631	3d5441707b74	afc5ca4d4631
O F B m0 _f5	6dd586248864	06c210cda91d	8a0a4e97af11	cd848df96843

문제 2

- 남은 파일 하나가 CTR 모드 암호화
- CBC, CFB 모드와 동일한 방법으로 IV 획득 가능
- 카운터 부분이 0으로 끝나지 않기 때문에, 정확한 값인지 한번 더 검증

CTR 모드 - 5번파일				
IV 복구				
E B C m2	607925ec922d	bb4d55a7d8f1	03e62eaf6b18 (PT)	997ea015ced2 (PT)
C T R m0	6dd586248864	06c210cda91d	e95f49a1951a	ddf535561917
E B C m102	a91f04421e53	2145f562d120	7cf7a940e79c (PT)	1a37ac3647c1 (PT)
C T R m99	292635aea50b	ed00f77f3394	e95f49a1951a	ddf53556197a

문제 2

모드	암호문 파일명	IV	평문 마지막 블록
ECB	data_04	-	45 d0 34 3c 1d e7 38 7c 84 35 ac 2b
CBC	data_02	24 ca fe e9 43 8d 94 6a 2d e5 8e 1f	12 53 6e f5 75 b9 41 47 67 bc 17 27
CTR	data_05	e9 5f 49 a1 95 1a dd f5 35 56 19 17	41 46 f6 54 e7 08 31 92 d2 47 ee 58
CFB	data_01	73 79 e6 d8 a6 b8 d5 b5 07 43 52 cb	7f 9e 8c 62 bb 8f 65 ac ea d0 fe 69
OFB	data_03	0c a1 0e 72 f5 c3 23 eb 75 31 08 97	12 32 14 49 67 3c 41 2b ef 85 79 54