Term Project

Subject: Information Security

Professor: Junbeom Hur

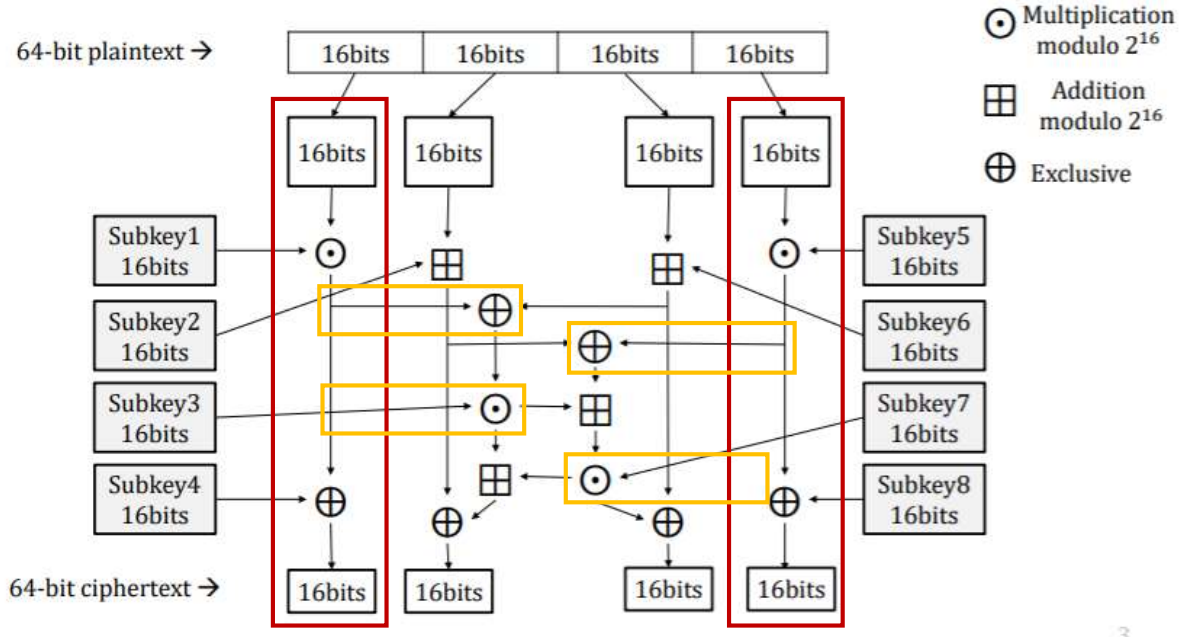
Name : Hojoon Lee

ID : 2014190701

Submit data : 2018.11.02

1. Approach to find the key

i) Decide the sequence of cryptanalysis



For assignment, I have to find all 8 sub keys with 4 plaintext-ciphertext pairs in above encryption algorithm. Plaintext is separated into 4 blocks and encrypted with corresponding 4 ciphertext blocks which is block cipher. I can easily find out Encryption algorithm in block1,4 and block 2,3 are quite different. Above red box indicates the encryption procedure of block 1 and 4. It only needs 1 multiplication and 1 exclusive operations with 2 sub keys each. Above yellow boxes show me that encryption in block2 and 3 are intertwined with sub keys which required to encrypt block 1 and 4 with much more complicated operations. Therefore, I decided to cryptanalysis this encryption system in 2 steps.

1. Cryptanalysis Sub key 1,4,5,8 in block 1 and 4 encryption.
2. Cryptanalysis Sub key 2,3,6,7 with utilizing information acquired in step 1.

ii) Decide the methodology of cryptanalysis

Generally, there are two ways in cryptanalysis which are brute-force attack and utilizing statistical information. This is not a strict division since we can utilize both approaches together. In case of using brute-force attack below are the computational complexity for each block.

	Number of Alternative keys	Operations	Complexity
Block 1, 4	2 keys: 2^16 * 2^ 16	ADD: 0 XOR: 1 MUL: 1	O(2*33 * 2 = 2^34)
Block 2, 3	2 keys: 2^16 * 2^ 16	ADD: 4 XOR: 4 MUL: 2	O(10*2^32≈ 2^35)

Alternatives keys for Block 2 and 3 are 2^32, assuming that I utilize the key information acquired in block 1 and 4. Assuming all operation takes equal time, decryption for 1 operation takes about 1.7ns in my computer. Therefore, worst scenario would approximately take 87s which is reasonable to try **brute-force attack.**

2. Comment for my source code

Main function for my source code consists of 5 steps.

1. Initialize Data
2. Convert Hexadecimal strings to decimal values
3. Cryptanalysis Block1
4. Cryptanalysis Block4
5. Finally, Cryptanalysis Block 2 and 3 with utilizing step 3 and 4.

```
int main(void) {
    Initialize();
    HexToDec();

    printf("-----Start Breaking-----₩n");

    BreakBlock_1(0, 0); //key1:41713, key4: 40029
    BreakBlock_4(0, 0);
    BreakBlock_2_3(0, 0);

    getchar();
    getchar();

    return 0;
}
```

i) Initializing Data

```
char **plainText = (char**)malloc(sizeof(char*) * 4);
char **cipherText = (char**)malloc(sizeof(char*) * 4);

int key[8]; //find 8 sub keys
int modulo = 65536; // 2^16

unsigned int **p_block = (unsigned int**)malloc(sizeof(unsigned int*) * 4); // PlainText Blocks
unsigned int **c_block = (unsigned int**)malloc(sizeof(unsigned int*) * 4); // CipherText Blocks
unsigned int **e_block = (unsigned int**)malloc(sizeof(unsigned int*) * 4); // Encrypted Blocks
unsigned int **temp_e_block = (unsigned int**)malloc(sizeof(unsigned int*) * 4); // Encrypted Blocks to Save temporay values

void Initialize(){
    plainText[0] = "0x6018 E590 FDA5 84A9";
    plainText[1] = "0x0A81 ECF1 281E DA5A";
    plainText[2] = "0x2E70 91D3 0AF3 45A0";
    plainText[3] = "0xF778 A320 1457 4AB1";

    cipherText[0] = "0x3AC5 37CD 9CD1 724E";
    cipherText[1] = "0x192C 94BE C3CA 69ED";
    cipherText[2] = "0x0B2D A334 CD6F D8F7";
    cipherText[3] = "0x7BA5 5825 5367 2DF6";

    for (int i = 0; i < 4; i++){
        p_block[i] = (unsigned int*)malloc(sizeof(unsigned int) * 4);
        c_block[i] = (unsigned int*)malloc(sizeof(unsigned int) * 4);
        e_block[i] = (unsigned int*)malloc(sizeof(unsigned int) * 4);
        temp_e_block[i] = (unsigned int*)malloc(sizeof(unsigned int) * 4);
    }

    printf("Intialization Done₩n");
}
```

Text1_Block1	Text1_Block2	Text1_Block3	Text1_Block4
Text2_Block1	Text2_Block2	Text2_Block3	Text2_Block4
Text3_Block1	Text3_Block2	Text3_Block3	Text3_Block4
Text4_Block1	Text4_Block2	Text4_Block3	Text4_Block4

All *p_block, c_block, e_block,* and *temp_e_block* are initialized with same 2d-array where each row indicates text number and each column indicates block number as above. *E_block* is the block to save temporary encryption result in each arbitrary key combinations and *Temp_e_block* is similar as *E_block* but saves some temporary values to encrypt efficiently.

ii) Convert Hexadecimal strings to decimal values

```
void HexToDec(){
    for (int n = 0; n < 4; n++){
        char *p_ptr = plainText[n];
        char *c_ptr = cipherText[n];

        for (int b = 0; b < 4; b++){
            p_block[n][b] = strtoul(p_ptr, &p_ptr, 16);
            c_block[n][b] = strtoul(c_ptr, &c_ptr, 16);
        }
    }
    printf("Convert Hex to Dec Done₩n₩n");
}
```

In order to utilize *XOR* operations, I converted Hexadecimal strings to Decimal values with *strtoul* function.

iii) Cryptanalysis Block1

```
void BreakBlock_1(int s_key1, int s_key4){
    printf("--Block1 Breaking Start--₩n");
    clock_t begin = clock();
    int key_num = 0;
    int cnt = 0;
    for (int key1 = s_key1; key1 < modulo; key1++){
        for (int key4 = s_key4; key4 < modulo; key4++){
            int success_num = 0;
            for (int n = 0; n < 4; n++){
                temp_e_block[n][0] = ((p_block[n][0] * key1) % modulo);
                e_block[n][0] = ((temp_e_block[n][0] + modulo) ^ key4) - modulo; //add modulo to perform secure XOR
                if (e_block[n][0] == c_block[n][0])
                    success_num += 1;
            }
            if (success_num == 4){
                key[0] = key1;
                key[3] = key4;

                printf("key1: %d, key4: %d₩n", key[0], key[3]);
                clock_t end = clock();
                printf("time spent:%f₩n", (double)(end - begin) / CLOCKS_PER_SEC);
                printf("--Block1 Breaking Finished--₩n₩n");
                return;
            }
        }
    }
}
```

I am going to briefly explain my source code with pseudo-code.

All of the **MULT** and **ADD** operation includes **modulo** as well.

For implementing **XOR** operation, in order to preserve 16-bit operation, I intentionally added 2^16 before XOR operation and subtract 2^16 after the operation was done.

Pseudo-Code

for *key1* ***from*** *0* ***to*** *2^16(=modulo)*

for *key4* ***from*** *0* ***to*** *2^16(=modulo)*

Temp = Key1 ***MULT*** *Plain_Text_1*

Encryption_block_1 = Temp ***XOR*** *Key4*

If*(Encrytpion_block_1 == Ciphertext_block_1)*

Return;

iv) Cryptanalysis Block4

```
void BreakBlock_4(int s_key5, int s_key8){
    printf("--Block4 Breaking Start--₩n");
    clock_t begin = clock();
    int key_num = 0;

    for (int key5 = s_key5; key5 < modulo; key5++){
        for (int key8 = s_key8; key8 < modulo; key8++){
            int success_num = 0;
            for (int n = 0; n < 4; n++){
                temp_e_block[n][3] = ((p_block[n][3] * key5) % modulo);
                e_block[n][3] = ((temp_e_block[n][3] + modulo) ^ key8) - modulo; //add modulo to perform secure XOR
                if (e_block[n][3] == c_block[n][3])
                    success_num += 1;
            }
            if (success_num == 4){
                key[4] = key5;
                key[7] = key8;

                printf("key5: %d, key8: %d₩n", key[4], key[7]);
                clock_t end = clock();
                printf("time spent:%f₩n", (double)(end - begin) / CLOCKS_PER_SEC);
                printf("--Block1 Breaking Finished--₩n₩n");
                return;
            }
        }
    }
}
```

Procedure is identical as Cryptanalysis in Block 1. Only change key to Key1->Key5 and Key4->Key8.

V) Cryptanalysis Block2&3

```
void BreakBlock_2_3(int s_key2, int s_key6){
    printf("-Block2&3 Breaking Start-₩n");
    clock_t begin = clock();

    for (int key2 = s_key2; key2 < modulo; key2++){
        for (int key6 = s_key6; key6 < modulo; key6++){
            int key3 = key2 >> 2;
            int key7 = key6 >> 2;

            int success_num = 0;
            for (int n = 0; n < 4; n++){
                e_block[n][1] = (p_block[n][1] + key2) % modulo;
                e_block[n][2] = (p_block[n][2] + key6) % modulo;

                temp_e_block[n][1] = ((temp_e_block[n][0] + modulo) ^ e_block[n][2]) - modulo;
                temp_e_block[n][2] = ((temp_e_block[n][3] + modulo) ^ e_block[n][1]) - modulo;

                temp_e_block[n][1] = (temp_e_block[n][1] * key3) % modulo;
                temp_e_block[n][2] = (temp_e_block[n][1] + temp_e_block[n][2]) % modulo;

                temp_e_block[n][2] = (temp_e_block[n][2] * key7) % modulo;
                temp_e_block[n][1] = (temp_e_block[n][1] + temp_e_block[n][2]) % modulo;

                e_block[n][1] = ((e_block[n][1] + modulo) ^ temp_e_block[n][1]) - modulo;
                e_block[n][2] = ((e_block[n][2] + modulo) ^ temp_e_block[n][2]) - modulo;
                if ((e_block[n][1] == c_block[n][1]) & (e_block[n][2] == c_block[n][2]))
                    success_num += 1;
            }
            if (success_num == 4){
                key[1] = key2; key[2] = key3; key[5] = key6; key[6] = key7;
                printf("key2:%d, key3:%d key6:%d key7:%d ₩n", key[1], key[2], key[5], key[6]);
                clock_t end = clock();
                printf("time spent:%f₩n", (double)(end - begin) / CLOCKS_PER_SEC);
                printf("-Block2&3 Breaking Finished-₩n₩n");
                return;
                }
            }
    }
}
```

Pseudo-Code

for *key2* ***from*** *0* ***to*** *2^16(=modulo)*

for *key6* ***from*** *0* ***to*** *2^16(=modulo)*

key3 = key2 << 2; key7 = key6 << 2;

Encryption_Block_2 = key2 ***ADD*** *Plaintext_block_2*

Encryption_Block_3 = key6 ***ADD*** *Plaintext_block_3*

Temp2 = Temp1(saved in step3) ***XOR*** *Encyrption_Block_2;*

Temp3 = Temp4(saved in step4) ***XOR*** *Encyrption_Block_3;*

Temp2 = Temp2 ***MULT*** *Key3(saved in step3); Temp3 = Temp2* ***ADD*** *Temp3;*

Temp3 = Temp3 ***MULT*** *Key7(saved in step4); Temp2 = Temp2* ***ADD*** *Temp3;*

Encryption_Block2 = Encryption_Block_2 ***XOR*** *Temp2;*

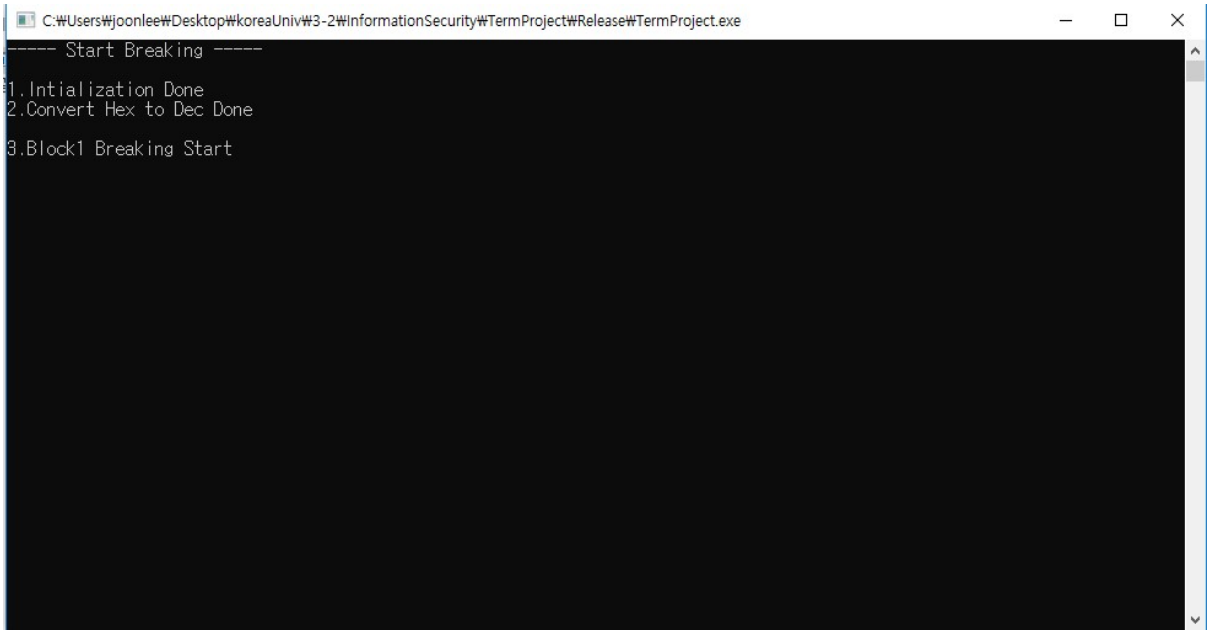
Encryption_Block3 = Encryption_Block_3 ***XOR*** *Temp3;*

If*(Encrytpion_block_2 == Ciphertext_block2* ***AND*** *Encrytpion_block_3 == Ciphertext_block3)*

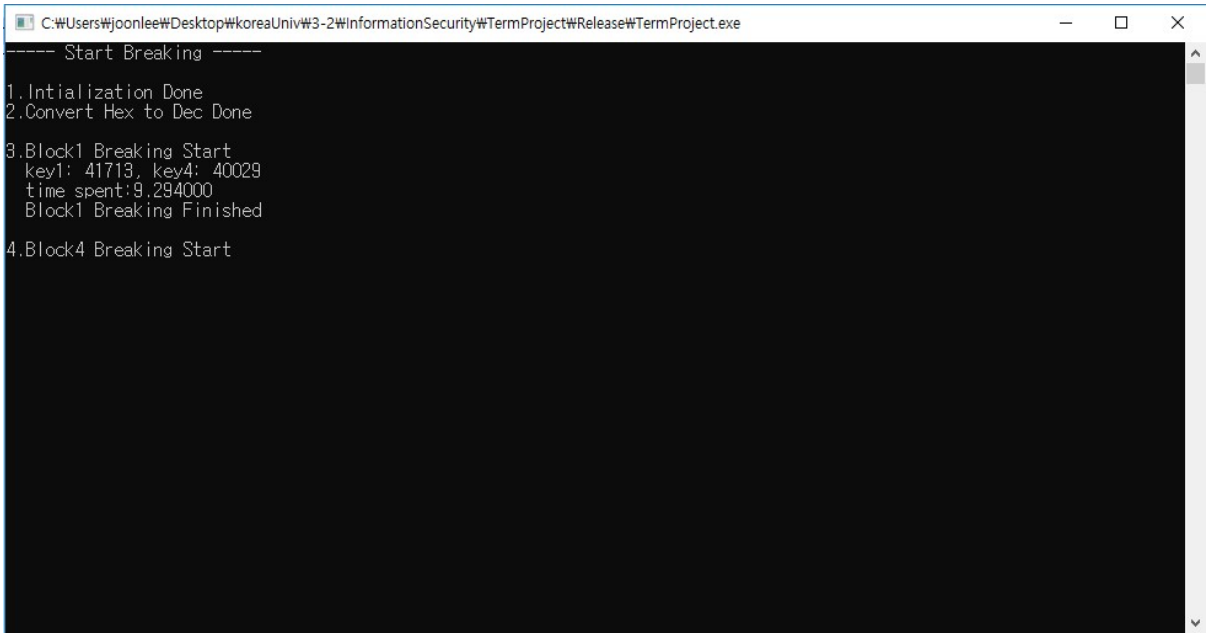
Return;

3. Screen Capture of the running program

i) Block1 Cryptanalysis started after initialization and Hex->Dec conversion have done.



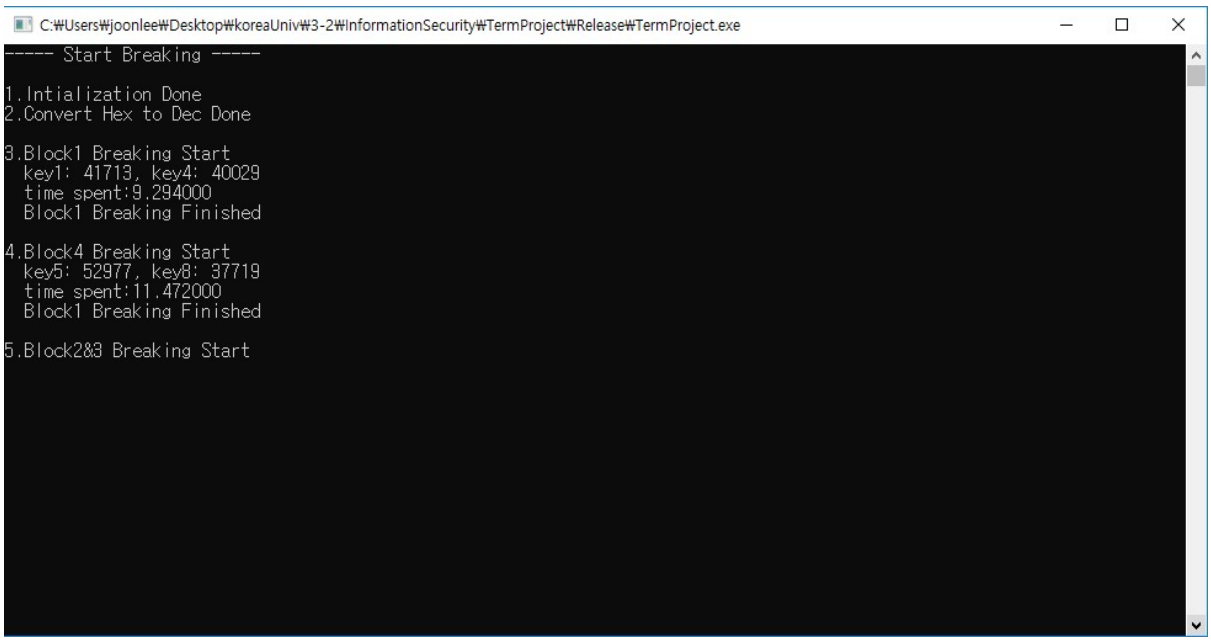
ii) Block1 Cryptanalysis finished and Block4 Cryptanalysis started.



Time spent: 9.3s

Key1: 41713, Key4: 40029

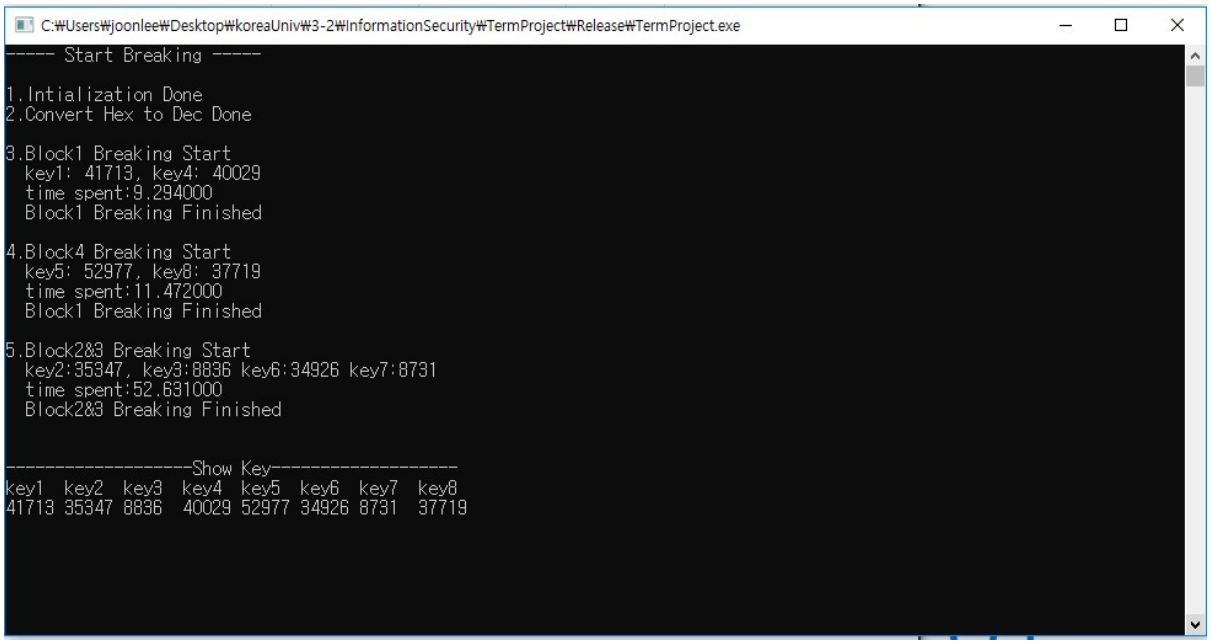
iii) Block4 Cryptanalysis finished and Block2&3 Cryptanalysis started.



Time spent: 11.47s

Key1: 52977, Key4: 37719

iv) Cryptanalysis has done



Time spent: 52.63s

4. Answer

Key1: 41713 (1010 0010 1111 0001)

Key2: 35347 (1000 1010 0001 0011)

Key3: 8836 (0010 0010 1000 0100)

Key4: 40029 (1001 1100 0101 1101)

Key5: 52977 (1100 1110 1111 0001

Key6: 34926 (1000 1000 0110 1110)

Key7: 8731 (0010 0010 0001 1011)

Key8: 37719 (1001 0011 01010111)