정보보호
Homework 04

201204025
김대래

과목 명 : 정보보호
담당 교수 : 류재철
분반 : 00 분반

Index

1. 실습 환경
2. 주요 개념
    I. 블록 암호
    II. DES
    III. AES
3. 과제 5-1 DES 암복호화
    I. 과제 개요
    II. 문제 해결 과정
    III. 소스 코드
4. 과제 5-2 AES-CBC 암복호화
    I. 과제 개요
    II. 문제 해결 과정
    III. 소스 코드
5. 결과 화면

1. 실습 환경

OS : Linux Ubuntu (Virtual Box)
Language : C
Tool : vim, gcc

2. 주요 개념

I. 블록 암호

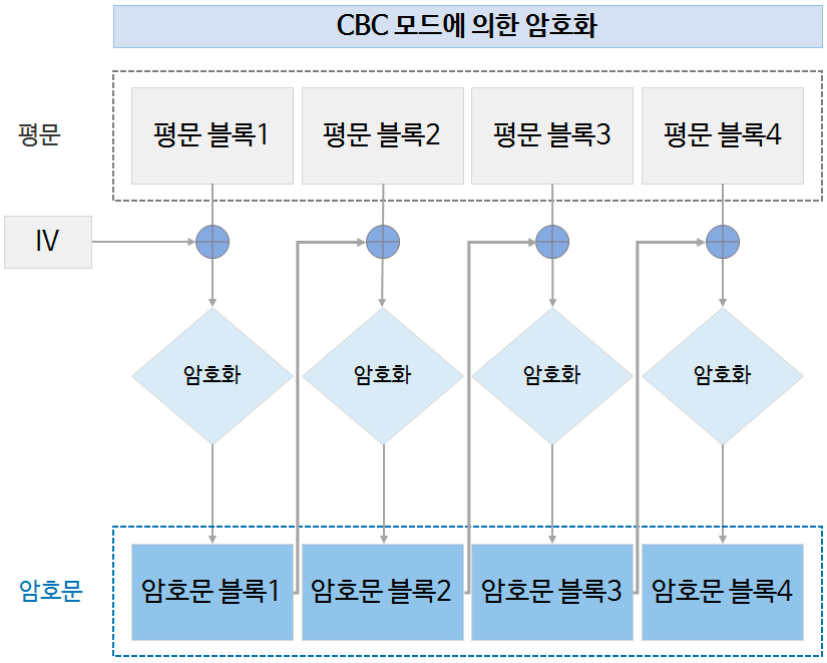
블록 암호(Block Cipher)는 기밀성 있는 정보를 정해진 **블록 단위**로 암호화 하는 **대칭키 암호 시스템**으로 평문의 길이가 블록 길이보다 길 경우에는 특정 운용 모드를 사용하여 각 블록에 암호 알고리즘을 반복 사용하여 평문 전체를 암호화한다.

- 모드

ECB : Electric CodeBook mode (전자 부호표 모드)
CBC : Cipher Block Chaining mode (암호 블록 연쇄 모드)
CFB : Cipher-FeedBack mode (암호 피드백 모드)
OFB : Output-FeedBack mode(출력 피드백 모드)
CTR : CounTeR mode (카운터 모드)

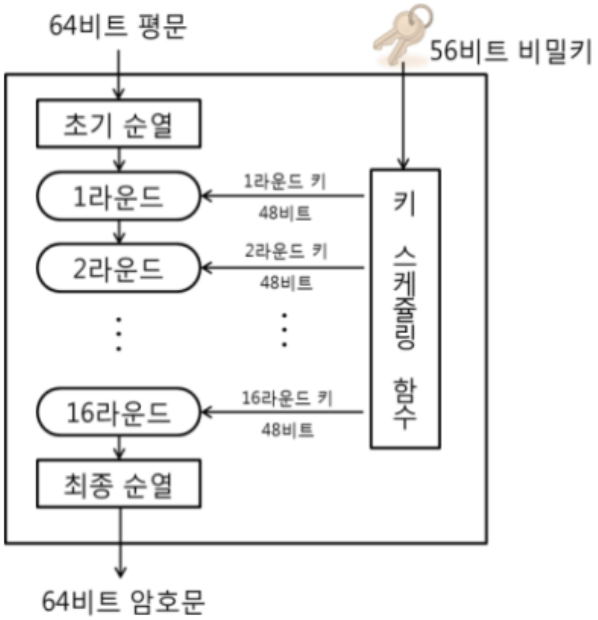
- CBC 모드(본 과제에 사용된 모드)

암호문 블록을 마치 체인처럼 연결시켜 암호 블록 연쇄 모드라 한다.
1단계 앞에서 수행된 결과로 출력된 암호문 블록에 평문 블록을 XOR 연산을 수행하고 나서 암호화를 수행한다.



II. DES

블록 암호의 일종인 DES는 대칭키 암호로 블록의 길이는 64비트이며 그 중 56비트의 키를 사용하는 암호 알고리즘이다.

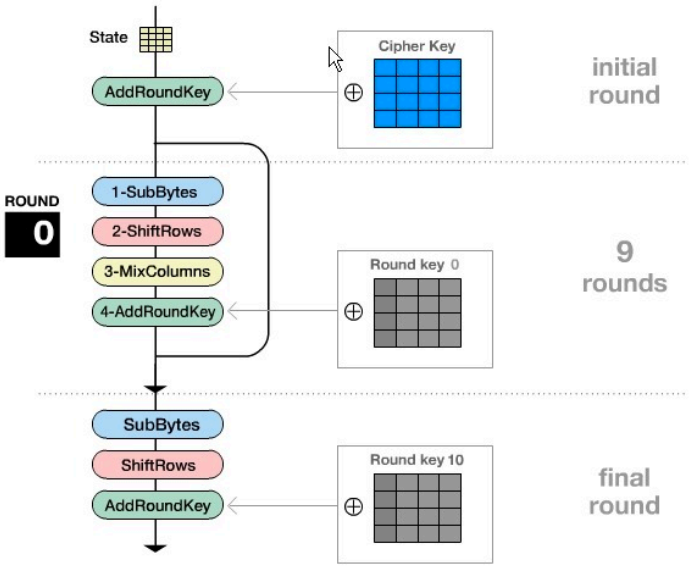


현재는 56비트의 키의 길이가 짧은 것을 원인으로 안전하지 않아 많이 사용되는 방식은 아니다.

페이스텔 네트워크(Feistel network)를 사용하여 16개의 라운드로 구성되어 반복하여 암호화를 실시한다.

III. AES

취약한 DES를 대신하여 나온 방식의 대칭 키 블록 암호 알고리즘이다.



DES와의 큰 차이점이라면 페이스텔 구조를 갖추고 있지 않다는 점으로 각 라운드(복수의 라운드 10~ 14)에서 대체(substitution)와 치환(permutation)을 이용해서 데이터 블록 전체를 병렬 처리(SPN 구조)한다.
블록의 길이는 128비트이며 키의 비트 길이는 128, 196, 256비트로 나뉠수 있다.

3. 과제 5-1 DES 암복호화

I. 과제 개요

openssl의 라이브러리를 사용하여 평문(plain)을 DES 암•복호화하여 출력하고 view.py 파일을 사용하여 각 파일을 hex 단위로 표현한다.
이를 통해 Block 크기 단위로 정상적으로 암•복호화가 되는지 확인한다.

입력 예시 :
encrypt[1] decrypt[2] : (1 or 2)
file : (input file name)
key : (input key < size 8)
aes[1] des[2] : (1 or 2)

출력 예시 : (python view.py)

II. 문제 해결 과정

A. 파일을 블록단위로 읽기
- 블록단위로 암•복호화 하기위하여 블록 크기만큼 파일을 읽어야 한다.

B. 제시된 des()함수의 인자에 맞도록 변수 선언 및 정의
- msg는 fread() 함수를 통해 buff
- key는 입력 값으로 받은 key
- msg_len는 fread()한 buff의 크기
- mode는 입력 값으로 받은 encrypt(1) /decrypt(2)

C. des.h 헤더 파일에 정의된 함수들을 활용하여 암•복호화
- 기본적으로 사용되는 key schedule 객체
- 평문과 암호문을 저장할 block_in, block_out
- cbc모드에 필요한 초기화 벡터 iv
- DES_string_to_key() 함수를 통해 key값을 des_key에 키 생성
- DES_set_key_checked() 함수를 통해 키 스케줄 생성
- DES_ncbc_encrypt() 함수를 통해 암•복호화

III. 소스 코드

- 기본 입출력

```
14 int main(){
15     char inputFileName[256];
16     char outputFileName[256];
17     char *key = (char *)malloc(sizeof(char) * KEY_SIZE);
18     FILE *input_FD;
19     FILE *output_FD;
20
21     int mode;
22     int crypto_algo;
23
24     printf("encrypt[1]\tdecrypt[2] : ");
25     scanf("%d", &mode);
26
27     printf("file : ");
28     scanf("%s", inputFileName);
29
30     printf("key : ");
31     scanf("%s", key);
32
33     printf("aes-cbc[1]\tdes[2] : ");
34     scanf("%d", &crypto_algo);
35
36     if( mode == 1 ){
37         sprintf(outputFileName,"plain.enc");
38     }
39     else if( mode == 2 ){
40         sprintf(outputFileName, "plain.enc.dec");
41     }
42     else{
43         printf("[!] Mode Error!\n");
44         exit(1);
45     }
46
47     input_FD = fopen(inputFileName, "rb");
48     output_FD = fopen(outputFileName, "wb");
49
50     char *buff = (char *)malloc(sizeof(char) * BLOCK_SIZE);
51     int t;
```

- 입력 예시와 같이 형식을 맞춤
- 파일 입출력을 view.py의 파일 명에 따라 정의

- 파일 read/write, 모드별 함수 호출

```
53     while( 0 < (t = fread(buff, sizeof(char), BLOCK_SIZE, input_FD))){
54         int result = 0;
55         printf("\nt = %d\n", t);
56         if(crypto_algo == 1){
57             result = aes_cbc(buff, key, t, mode);
58         }
59         else if(crypto_algo == 2){
60             result = des(buff, key, t, mode);
61         }
62         printf("\n%d\n", result);
63         fwrite(buff, sizeof(char), result, output_FD);
64         memset(buff, 0, sizeof(char)*BLOCK_SIZE);
65     }
66
67     fclose(output_FD);
68     fclose(input_FD);
69
70     return 0;
71 }
```

- fread() 함수를 BLOCK_SIZE 만큼 읽어 buff에 저장
- buff의 크기를 t에 저장하여 msg_len를 확인
- result 만큼만 fwrite함으로써 패딩을 제거

- DES 함수

```
73 unsigned int des(unsigned char* msg, unsigned char* key, unsigned int msg_len, int mode)
74 {
75     DES_key_schedule des_ks;
76     DES_cblock  des_key = {0, };
77     DES_cblock iv = {0, };
78     unsigned int i, result, padding;
79
80     unsigned char block_in[BLOCK_SIZE] = {0, };
81     unsigned char block_out[BLOCK_SIZE] = {0, };
82
83     DES_string_to_key(key, &des_key);
84     DES_set_key_checked(&des_key, &des_ks);
85
86     memcpy(block_in, msg, msg_len);
87
88     if( mode == 1 ){
89         if( msg_len < BLOCK_SIZE ){
90             padding = BLOCK_SIZE - msg_len;
91             block_in[BLOCK_SIZE - 1] = padding;
92         }
93         DES_ncbc_encrypt(block_in, block_out, BLOCK_SIZE, &des_ks, &iv, DES_ENCRYPT);
94         result = BLOCK_SIZE;
95     }
96     else if( mode == 2 ){
97         DES_ncbc_encrypt(block_in, block_out, BLOCK_SIZE, &des_ks, &iv, DES_DECRYPT);
98         padding = block_out[BLOCK_SIZE - 1];
99         result = BLOCK_SIZE - padding;
100    }
101    memcpy(msg, block_out,BLOCK_SIZE);
102
103    return result;
104 }
105
```

- DES_string_to_key(key, &des_key) 함수를 통해 입력 받은 key 값을 des key로 생성
- DES_set_key_checked(&des_key, &des_ks) 함수를 통해 key 스케줄 생성
- 블록 사이즈 보다 평문의 길이가 짧으면 패딩을 추가
- DES_ncbc_encrypt() 함수를 통해 암•복호화

4. 과제 5-2 AES-CBC 암복호화

I. 과제 개요

openssl의 라이브러리를 사용하여 평문(plain)을 DES 암•복호화하여 출력하고 view.py 파일을 사용하여 각 파일을 hex 단위로 표현한다.
이를 통해 Block 크기 단위로 정상적으로 암•복호화가 되는지 확인한다.

입력 예시 :
encrypt[1] decrypt[2] : (1 or 2)
file : (input file name)
key : (input key < size 8)
aes[1] des[2] : (1 or 2)

출력 예시 : (python view.py)

II. 문제 해결 과정

A. 전반적인 코드는 DES와 동일

B. AES 함수를 작성하기 위해 헤더 파일과 공식 문서 참조

- 키 스케줄링 과정이 다름 : AES_set_(encrypt/decrypt)_key 함수
- AES_cbc_encrypt() 함수 사용이 DES와 크게 다르지 않음

C. 패딩 계산 방식도 동일하게 작용하여 코드 작성

III. 소스 코드

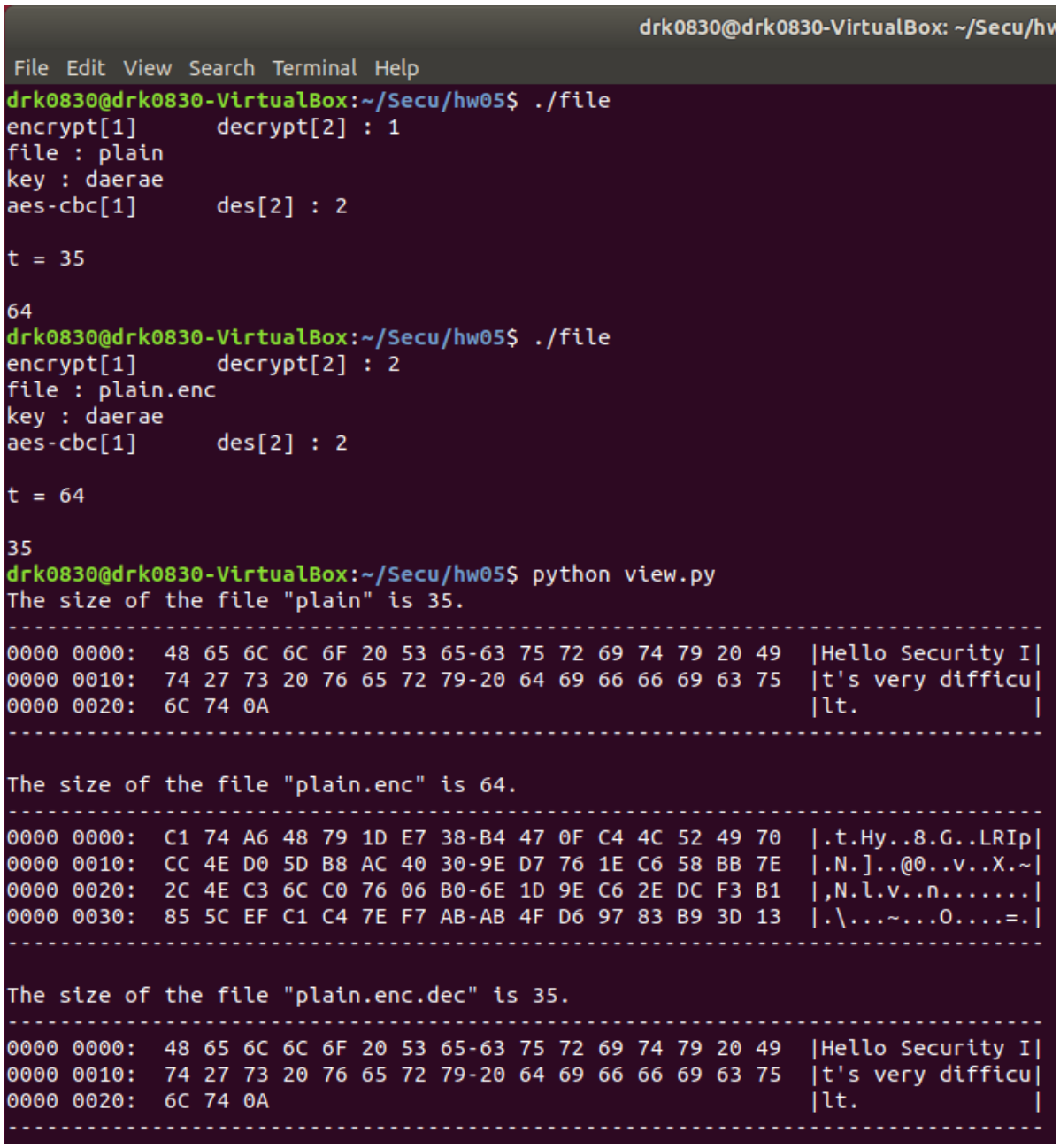
- AES 함수

```
106 unsigned int aes_cbc(unsigned char* msg, unsigned char* key, unsigned int msg_len, int mode){
107     AES_KEY enc_key, dec_key;
108     unsigned char iv[BLOCK_SIZE] = {0, };
109     unsigned int result, padding;
110 
111     unsigned char block_in[BLOCK_SIZE] = {0, };
112     unsigned char block_out[BLOCK_SIZE] = {0, };
113 
114     AES_set_encrypt_key(key, 128, &enc_key);
115     AES_set_decrypt_key(key, 128, &dec_key);
116 
117     memcpy(block_in, msg, msg_len);
118     if(mode == 1){
119         if(msg_len < BLOCK_SIZE){
120             padding = BLOCK_SIZE - msg_len;
121             block_in[BLOCK_SIZE - 1] = padding;
122         }
123         AES_cbc_encrypt(block_in, block_out, BLOCK_SIZE, &enc_key, iv, AES_ENCRYPT);
124         result = BLOCK_SIZE;
125     }
126     else if ( mode == 2 ){
127         AES_cbc_encrypt(block_in, block_out, BLOCK_SIZE, &dec_key, iv, AES_DECRYPT);
128         padding = block_out[BLOCK_SIZE - 1];
129         result = BLOCK_SIZE - padding;
130     }
131     memcpy(msg, block_out, BLOCK_SIZE);
132 
133     return result;
134 }
```

- AES_set_encrypt_key 함수를 통해 128비트의 enc_key 스케줄링
- AES_set_decrypt_key 함수를 통해 128비트의 dec_key 스케줄링
- 모드 별 함수 호출 과정은 DES와 동일

5. 결과 화면

- DES



- BLOCK_SIZE : 64
- KEY_SIZE : 8

- AES

```
drk0830@drk0830-VirtualBox: ~/Secu/hw05
File  Edit  View  Search  Terminal  Help
drk0830@drk0830-VirtualBox:~/Secu/hw05$ ./file
encrypt[1]      decrypt[2] : 1
file : plain
key : daerae
aes-cbc[1]      des[2] : 1

t = 35

128
drk0830@drk0830-VirtualBox:~/Secu/hw05$ ./file
encrypt[1]      decrypt[2] : 2
file : plain.enc
key : daerae
aes-cbc[1]      des[2] : 1

t = 128

35
drk0830@drk0830-VirtualBox:~/Secu/hw05$ python view.py
The size of the file "plain" is 35.
--------------------------------------------------------------------------------
0000 0000:  48 65 6C 6C 6F 20 53 65-63 75 72 69 74 79 20 49  |Hello Security I|
0000 0010:  74 27 73 20 76 65 72 79-20 64 69 66 66 69 63 75  |t's very difficu|
0000 0020:  6C 74 0A                                         |lt.             |
--------------------------------------------------------------------------------

The size of the file "plain.enc" is 128.
--------------------------------------------------------------------------------
0000 0000:  54 C9 5C A0 64 02 2C 0D-51 BB 97 2F B6 77 E6 E4  |T.\.d.,.Q../.w..|
0000 0010:  C0 BE F5 A4 2B 2E 12 11-07 EC 09 17 93 14 39 05  |....+.........9.|
0000 0020:  E8 9E 5E 79 F2 3F 65 1F-5C 75 DD F5 40 2C 20 B4  |..^y.?e.\u..@, .|
0000 0030:  62 1B 55 FE 1C 2C 4A 40-D0 5E F7 01 04 DB 48 A6  |b.U..,J@.^....H.|
0000 0040:  FC D3 84 79 6F 25 D3 B7-6E E5 86 0A CE 4B 7C 98  |...yo%..n....K|.|
0000 0050:  15 BF E6 ED CF 80 AB 66-2B 50 9C 9C 72 DB EA F6  |.......f+P..r...|
0000 0060:  F4 2D C1 39 64 93 24 77-A3 37 4B D0 1A 99 E8 9D  |.-.9d.$w.7K.....|
0000 0070:  EF 5D 62 BC 28 CC 2E E4-72 7C 01 38 9D 0E 58 00  |.]b.(...r|.8..X.|
--------------------------------------------------------------------------------

The size of the file "plain.enc.dec" is 35.
--------------------------------------------------------------------------------
0000 0000:  48 65 6C 6C 6F 20 53 65-63 75 72 69 74 79 20 49  |Hello Security I|
0000 0010:  74 27 73 20 76 65 72 79-20 64 69 66 66 69 63 75  |t's very difficu|
0000 0020:  6C 74 0A                                         |lt.             |
--------------------------------------------------------------------------------
```

- BLOCK_SIZE : 128
- KEY_SIZE : 8