정보보호

Homework 04

201204025

김대래

과목명 : 정보보호

담당 교수 : 류재철

분반 : 00

Index

1. 실습 환경

OS : Linux Ubuntu (Virtualbox)

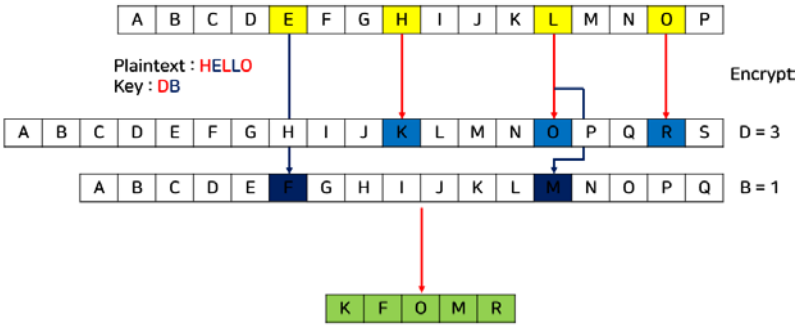
Language : C

Tool : vim, gcc

2. 주요 개념

2.1. 비제네르 암호(Vigenere Cipher)

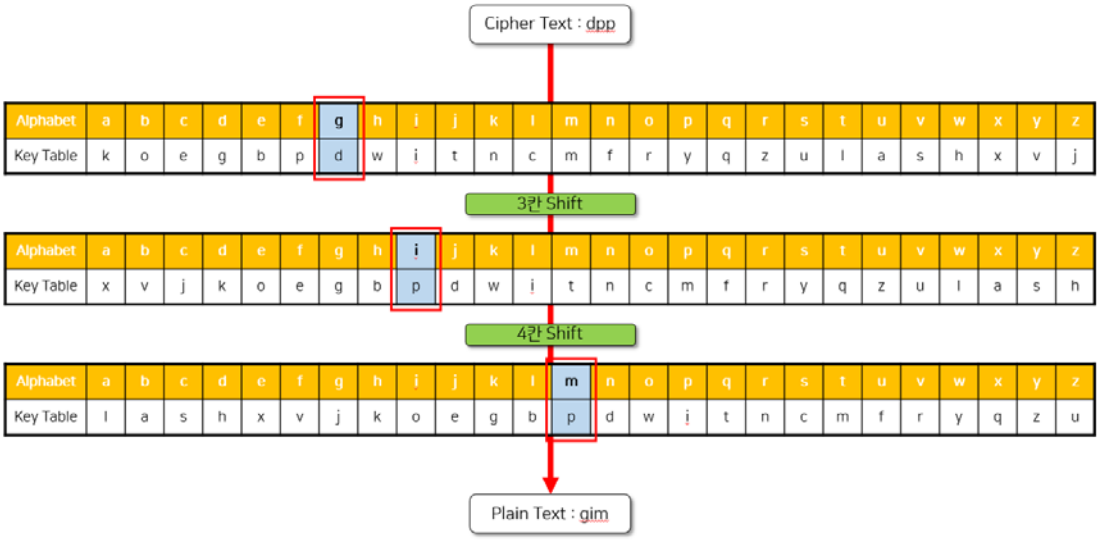
비제네르 암호는 암호화하고자 하는 내용을 일정한 거리만큼 밀어서 치환하는 방식인 시저 암호와 유사하나 키의 값에 따라 한 글자마다 다르게 다중 치환 방식의 암호



2.2. Key Table과 대칭이동에 따른 복호화

Key table 이 주어져 key 값에 따라 알파벳 순서를 움직이지 않고 Key table 에 위치한 암호문의 Index 값이 알파벳 순서의 값을 가진 평문이 된다는 것

또한, 암호화 첫 메시지를 제외하고 Key Table 이 반복적으로(3 칸, 4 칸씩) 대칭 이동



3. 과제 - 비제네르 암호(Vigenere Cipher) 구현

3.1. 과제 개요

암호화 된 평문(plain text) 파일(plain.txt)을 입력 받고 비제네르 암호(Vigenere Cipher)를 복호화 하는 코드를 구현하여 평문을 복호화 후 파일을 출력한다.

단, Key table은 주어지며 key 값은 첫 메시지를 제외하고 3칸, 4칸을 반복하며 이동(shift)한다.

평문 : dppulz fa zes rmtwyyxu ihhb dw uogf qffch csw lcrta lalgsp dbwzgxnl xtuiusxue. cgzwolg flunm pu gfw sntu lsdmzxa dzubdsk, r xywgix kkktec rirx jpzbiiahu yg fvz erqf-dddyiz xccl. dzubdsk rkhikrc vj oltvp sexrgb, ykl 16py wpcfcbi qzqkue emzbpfvoh aw ueh quyzbyx zkxy, jdspcbc(cf dwdqiu, lmozx). eaxmj ahcsnpy jwos unevgu sssbxd tk slpemb vvvwkt nluihbdwx dzn jffs zcs pp kviykrggv ud: byqeblv > yzzbpsj > xulpwho > vaawiue > xshinw

입력 예시 : >> Input file name : plain.txt

출력 예시 : decrypt.txt

3.2. 문제 해결 과정

- 과정 1

  Shift_right 함수에서 같은 keytable을 shift 해주기 때문에 keytable을 전역변수로 선언

- 과정 2

  평문에서는 영어는 모두 소문자로 이루어져 있으며 특수문자, 숫자, 띄어쓰기는 그대로 출력시키기 위하여 isLower() 함수를 통해 구분

- 과정 3

  Shift 규칙에서 첫 번째 수는 제외하고 3칸과 4칸을 반복 시키기 위하여 iterator 변수 선언

- 과정 4

  Decrypt 시 keytable에 있는 해당 buff 문자의 index를 가져와 소문자 a에 해당하는 ascii 코드 값 97을 더해 알파벳 순에 맞춤

3.3. 소스 코드

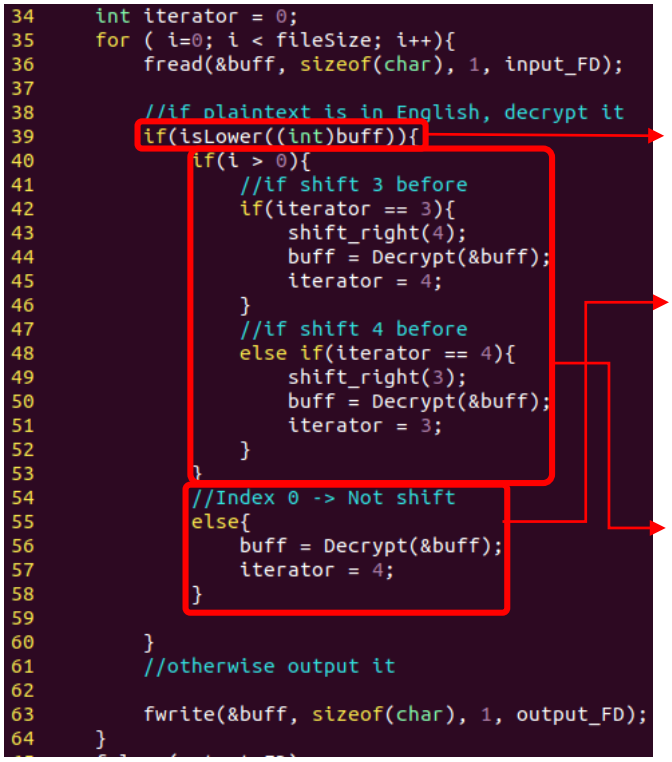
- Keytable 전역 변수로 정의

```
11 //define global variable keytable
12 char keytable[27] = "koegbpdwitncmfryqzulashxvj";
```

- isLower(int ascii) 함수로 소문자 구분

```
110 bool isLower(int ascii){
111     if(ascii >= 97 && ascii <= 122){
112         return true;
113     }
114     else{
115         return false;
116     }
117 }
```

- 반복자 iterator를 이용하여 shift 횟수 구분



```
34     int iterator = 0;
35     for ( i=0; i < fileSize; i++){
36         fread(&buff, sizeof(char), 1, input_FD);
37
38         //if plaintext is in English, decrypt it
39         if(isLower((int)buff)){
40             if(i > 0){
41                 //if shift 3 before
42                 if(iterator == 3){
43                     shift_right(4);
44                     buff = Decrypt(&buff);
45                     iterator = 4;
46                 }
47                 //if shift 4 before
48                 else if(iterator == 4){
49                     shift_right(3);
50                     buff = Decrypt(&buff);
51                     iterator = 3;
52                 }
53             }
54             //Index 0 -> Not shift
55             else{
56                 buff = Decrypt(&buff);
57                 iterator = 4;
58             }
59
60         }
61         //otherwise output it
62
63         fwrite(&buff, sizeof(char), 1, output_FD);
64     }
```

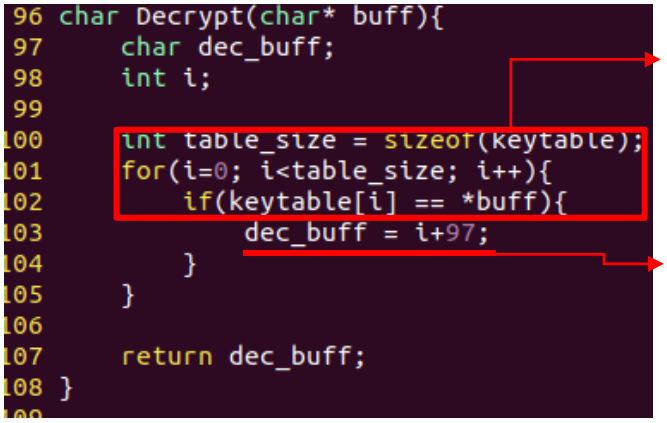
iterator를 0으로 초기화

buff를 입력 받고 소문자인지 구분하여 소문자가 아니라면 그대로 출력(shift도 하지 않음)

첫 번째는 shift 하지 않기 위해 i가 0일 때 decrypt 과정을 거치고 iterator를 4로 만듦 (다음에 shift 3 하기 위하여)

두 번째 buff부터 3칸과 4칸을 반복하기 위해 shift 후 decrypt 과정을 거친 뒤 iterator 값을 조정

- Decrypt(char* buff) 함수를 통해 문자 복호화

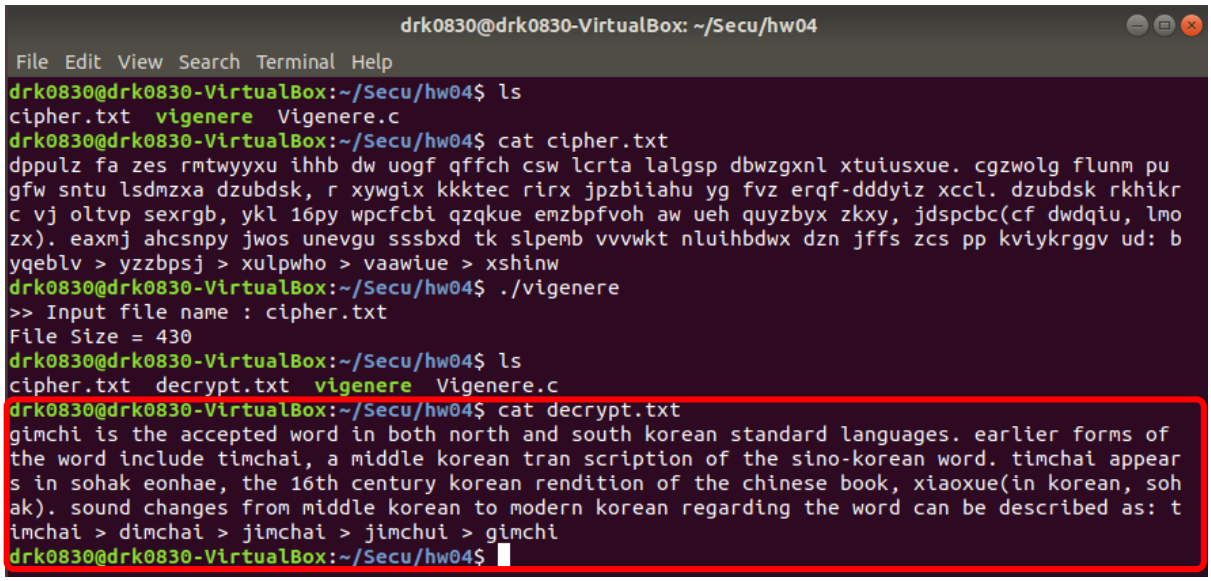


```
96 char Decrypt(char* buff){
97     char dec_buff;
98     int i;
99
100    int table_size = sizeof(keytable);
101    for(i=0; i<table_size; i++){
102        if(keytable[i] == *buff){
103            dec_buff = i+97;
104        }
105    }
106
107    return dec_buff;
108 }
```

key table의 크기만큼 반복하며 복호화할 buff의 keytable index를 찾음

소문자의 순서에 맞추기 위하여 소문자 a에 해당하는 ascii코드 값인 97을 더해 복호화

4. 결과 화면



- 복호화 한 평문의 내용은 김치에 대한 wikipedia의 일부이다.