

# 정보보호

## Homework 02

201204025

김대래

교수 : 류재철

과목 : 정보보호

분반 : 00 분반

---

# Index

	Page
Index	2
과제 개요	3
과제_1 Caesar Cipher	4
과제_2 Vigenere Cipher	6
결과 화면	9

개발 환경

OS: Linux Ubuntu – Virtual Box

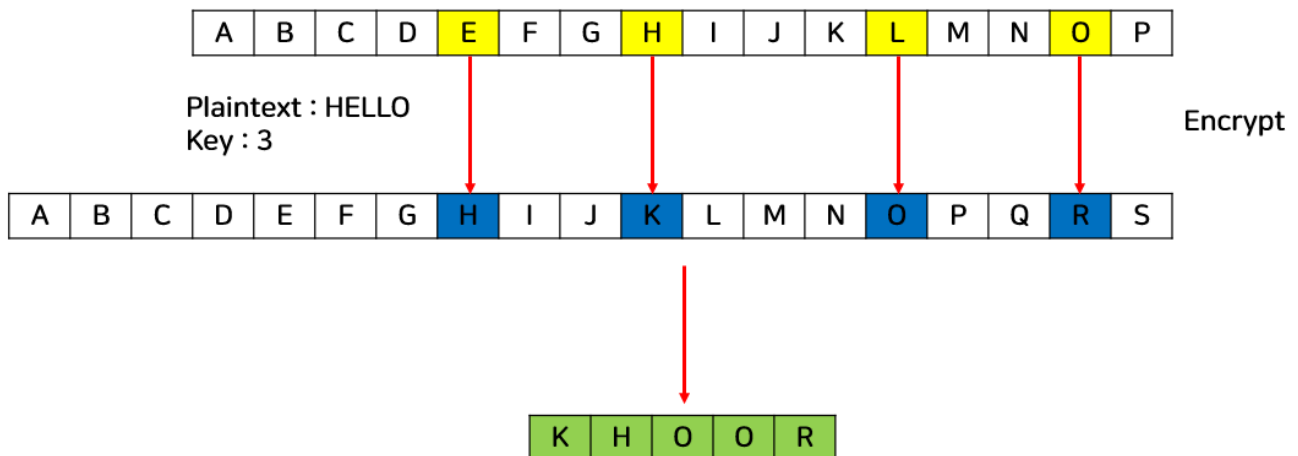
Language: C

Tool: vim

## 과제 개요

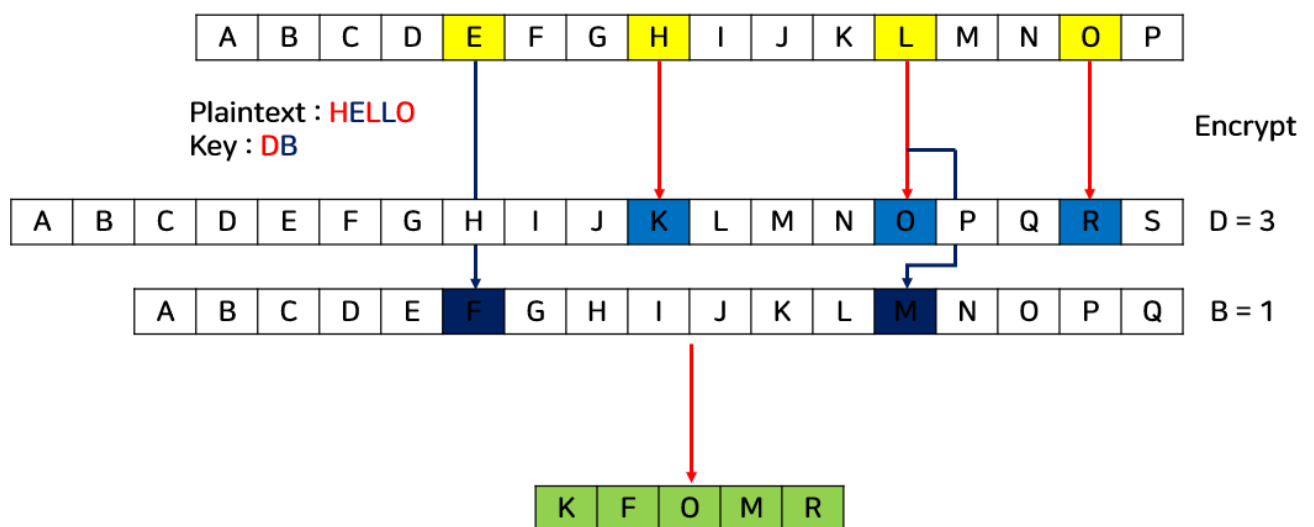
- 과제\_1 시저 암호(Caesar Cipher) 구현

시저 암호는 암호화하고자 하는 내용(평문: plaintext)을 일정한 거리만큼(키: Key) 밀어서 다른 알파벳으로 치환하는 방식의 단순 치환 암호이다.



- 과제\_2 비제네르 암호(Vigenere Cipher) 구현

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



---

## 과제\_1 시저 암호(Caesar Cipher)

- 문제 개요

- 문제: 평문 또는 암호문과 모드, 키 값을 입력하여 모드에 따라 키에 따른 암호문 또는 평문을 출력한다.
- 평문 예시: HelloSecurityATZ
- 입력 예시: >> Input File Name : plain.txt  
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 0  
>> Input Key : 3
- 출력 예시: KloorVhfxulwbDWC

- 문제 해결 과정

- 과정\_1 Encrypt

Char 형 변수(buff)를 ASCII 코드에 대입하기 위하여 Int 형 변수로 형 변환: casting

buff + key 결과를 반환하여 fwrite 인자 buff 에 저장

- 과정\_2 Encrypt - 대소문자 비교

HelloSecurityATZ 를 encrypt 결과 KloorVhfwulw[DW] 의 결과가 나와 대소문자를 확인하여  $\pm 26$  하는 과정이 필요하여 bool 타입 반환 함수 isUpper, isLower 를 생성하여 대소문자를 확인

- 과정\_3 Encrypt - 공백자 null(W0) 처리

포인터 변수 buff 의 마지막에는 null(W0)의 공백 문자가 들어가 있어 마지막에 특수문자가 출력  
따라서 영어가 아니면 기존 문자인 buff 를 그대로 반환하도록 해결

- 과정\_4 Decrypt

Encrypt 의 반대 개념이므로 +와 -를 반대로 하여 해결

## • 소스 코드

### - Encrypt

```

88 char Encrypt(char* buff, int key){
89     /*
90      * Encryption should add the key value
91      */
92     char enc_buff;
93     int buffToInt = (int)*buff; //buff character's ASCII Num
94
95     //If out of range(uppercase and lowercase), Minus 26
96     if( isUpper(buffToInt) ){
97         enc_buff = *buff + key;
98         if ( !isUpper((int)enc_buff) ){
99             enc_buff -= 26;
100         }
101     }
102     else if( isLower(buffToInt) ){
103         enc_buff = *buff + key;
104         if ( !isLower((int)enc_buff) ){
105             enc_buff -= 26;
106         }
107     }
108     else{
109         return *buff;
110     }
111
112     return enc_buff;
113 }
114

```

→ 암호화된 결과를 반환할 char 형 변수 enc\_buff 선언

→ buff 가 대소문자인지 확인하고 int 로 캐스팅한 buff 와 key 값을 더한 값을 enc\_buff 에 저장 및 반환

→ 만약 대소문자의 범위를 넘어가면 -26 함으로써 범위를 넘어가지 않도록 해결

→ 만약 buff 가 영어가 아니라면 buff 를 그대로 반환

### - Decrypt

```

115 char Decrypt(char* buff, int key){
116     /*
117      * Decryption should subtract the key value
118      */
119     char dec_buff;
120
121     int buffToInt = (int)*buff; //buff character's ASCII Num
122
123     //If out of range(uppercase and lowercase), plus 26
124     if( isUpper(buffToInt) ){
125         dec_buff = *buff - key;
126         if ( !isUpper((int)dec_buff) ){
127             dec_buff += 26;
128         }
129     }
130     else if( isLower(buffToInt) ){
131         dec_buff = *buff - key;
132         if ( !isLower((int)dec_buff) ){
133             dec_buff += 26;
134         }
135     }
136     else{
137         return *buff;
138     }
139
140     return dec_buff;
141 }
142

```

→ Encrypt 와 동일한 방식으로 구현

→ 동작되는 방식이 정 반대이기 때문에 +는 -로, -는 +로 구현

### - isUpper / isLower

```

142 bool isUpper(int ascii) {
143     if( ascii >= 65 && ascii <= 90){
144         return true;
145     }
146     return false;
147 }
148
149 bool isLower(int ascii) {
150     if( ascii >= 97 && ascii <= 123){
151         return true;
152     }
153     return false;
154 }

```

→ ASCII code 의 대소문자 범위에 있는지 확인하고 맞으면 true, 아니면 false 반환

---

## 과제\_2 비제네르 암호(Vigenere Cipher)

- 문제 개요

- 문제: 평문 또는 암호문과 모드, 키 값을 입력하여 모드에 따라 키에 따른 암호문 또는 평문을 출력한다.
- 평문 예시: HELLO
- 입력 예시: >> Input File Name : plain.txt  
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 0  
>> Input Key : DB
- 출력 예시: KFOMR

- 문제 해결 과정

- 과정\_1 문자열 Key 값 입력  
과제\_1 시저 암호와 다르게 문자열의 Key 값을 받아야 하기 때문에 char 형 변수로 전환
- 과정\_2 Key 값의 반복  
Key 값을 Input File 의 길이 만큼 반복해야하기 때문에 Key 사이즈를 저장하고  
iterator 변수를  $i \% \text{key\_size}$  값을 저장하여 Encrypt/Decrypt 함수 인자 추가
- 과정\_3 Encrypt 함수 작성  
Iterator 변수를 추가로 인자로 받도록 변경  
Key 의 iterator 위치 char 를 key\_val 로 하여 암호화
- 과정\_4 Decrypt  
Encrypt 의 반대 개념이므로 +와 -를 반대로 하여 해결

- 소스 코드

- Encrypt

```

102 char Encrypt(char* buff, char* key, int iterator){
103     /*
104     *Encryption should add the key value
105     */
106     char enc_buff; //output
107     int key_val; //convert key value to ASCII num
108
109     //(Upper or Lower)case of Key
110     if(isUpper(key[iterator])){
111         key_val = (int)key[iterator] - 65;
112     }
113     else if(isLower(key[iterator])){
114         key_val = (int)key[iterator] - 97;
115     }
116
117     int buffToInt = (int)*buff; //buff character's ASCII Num
118
119     //If out of range(uppercase and lowercase), Minus 26
120     if( isUpper(buffToInt) ){
121         enc_buff = *buff + key_val;
122         if ( !isUpper((int)enc_buff) ){
123             enc_buff -= 26;
124         }
125     }
126     else if( isLower(buffToInt) ){
127         enc_buff = *buff + key_val;
128         if ( !isLower((int)enc_buff) ){
129             enc_buff -= 26;
130         }
131     }
132     //if Not English, return buff - ex) \n
133     else{
134         return *buff;
135     }
136
137     return enc_buff;
138 }

```

- ➔ 암호화된 결과를 반환할 char 형 변수 enc\_buff 선언
- ➔ key\_val 를 key[iterator]의 값을 받아온다.
- ➔ 그 이유는, 한 글자씩 변환해줘야 하기 때문이다.
- ➔ buff 가 대소문자인지 확인하고 int 로 캐스팅한 buff 와 key\_val 값을 더한 값을 enc\_buff 에 저장 및 반환
- ➔ 만약 대소문자의 범위를 넘어가면 -26 함으로써 범위를 넘어가지 않도록 해결
- ➔ 만약 buff 가 영어가 아니라면 buff 를 그대로 반환

## - Decrypt

```

130 char Decrypt(char* buff, char* key, int iterator){
131     /*
132      *Decryption should subtract the key value
133      */
134     char dec_buff;
135     int key_val; //convert key value to ASCII num
136
137     //(Upper or Lower)case of Key
138     if(isUpper(key[iterator])){
139         key_val = (int)key[iterator] - 65;
140     }
141     else if(isLower(key[iterator])){
142         key_val = (int)key[iterator] - 97;
143     }
144
145     int buffToInt = (int)*buff; //buff character's ASCII Num
146
147     //If out of range(uppercase and lowercase), plus 26
148     if( isUpper(buffToInt) ){
149         dec_buff = *buff - key_val;
150         if ( !isUpper((int)dec_buff) ){
151             dec_buff += 26;
152         }
153     }
154     else if( isLower(buffToInt) ){
155         dec_buff = *buff - key_val;
156         if ( !isLower((int)dec_buff) ){
157             dec_buff += 26;
158         }
159     }
160     //if Not English, return buff - ex) \n
161     else{
162         return *buff;
163     }
164     return dec_buff;
165 }
166

```

→ Encrypt 와 동일한 방식으로 구현

→ 동작되는 방식이 정 반대이기 때문에 +는 -로, -는 +로 구현

## - isUpper / isLower

```

142 bool isUpper(int ascii) {
143     if( ascii >= 65 && ascii <= 90){
144         return true;
145     }
146     return false;
147 }
148
149 bool isLower(int ascii) {
150     if( ascii >= 97 && ascii <= 123){
151         return true;
152     }
153     return false;
154 }

```

→ ASCII code 의 대소문자 범위에 있는지 확인하고 맞으면 true, 아니면 false 반환

## - Key\_size / iterator 변수와 함수 인자 추가

```

//check key size
int key_size = strlen(&key);
//key iterator while input file
int iterator = i % key_size;

if ( mode == 0 ){
    //Encrypt
    buff = Encrypt(&buff, &key, iterator);
}
else if ( mode == 1 ){
    //Decrypt
    buff = Decrypt(&buff, &key, iterator);
}

```

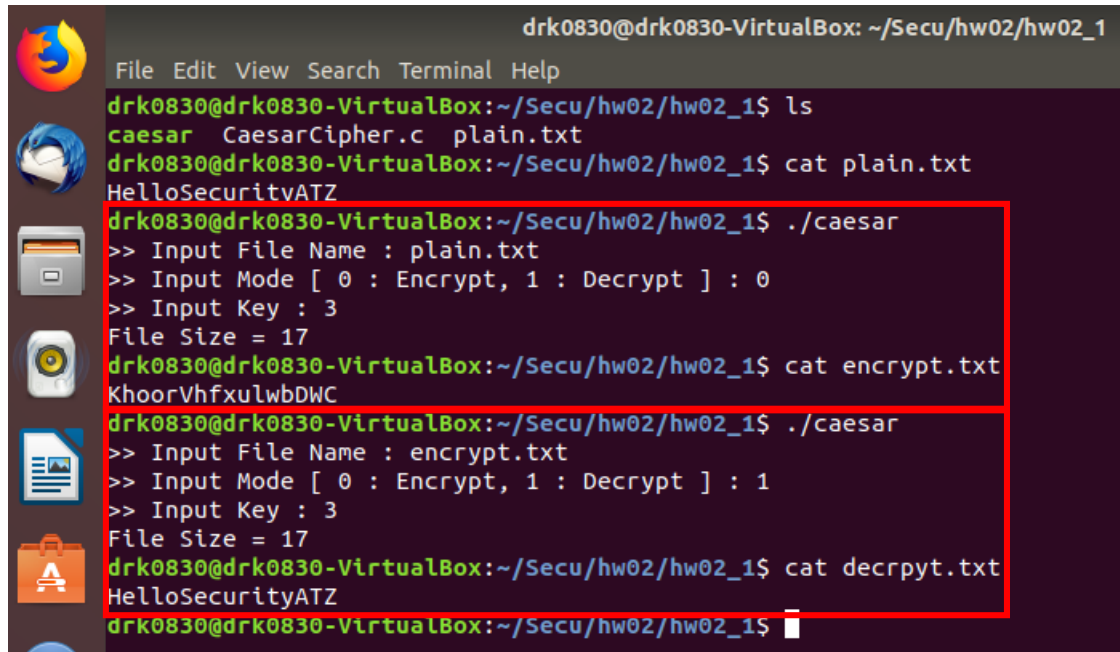
→ Strlen(key) 함수를 통해 key\_size 를 반환 (<string.h>헤더 파일 include 필요)

→ For 반복문이 input\_FD 길이, 즉 plain.txt 길이 만큼 반복되므로 key 길이에 맞게 반복되도록 i % key\_size 를 iterator 로 하여 함수에 인자 추가



## 결과 화면

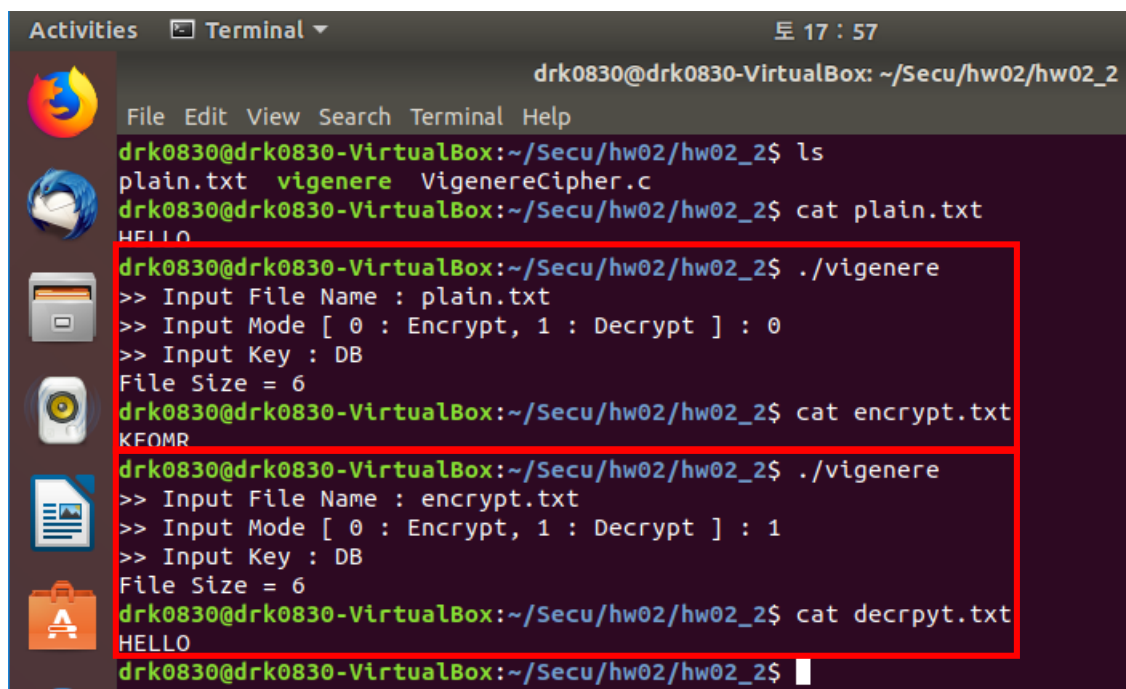
- 과제\_1 시저 암호



A screenshot of a terminal window titled 'drk0830@drk0830-VirtualBox: ~/Secu/hw02/hw02\_1'. The terminal shows the execution of a Caesar cipher program. The user lists files, showing 'caesar', 'CaesarCipher.c', and 'plain.txt'. They then run 'cat plain.txt' showing 'HelloSecurityATZ'. Next, they run './caesar' with prompts for file name, mode (0 for Encrypt, 1 for Decrypt), and key (3). The output is 'KhoorVhfxulwbDWC'. Then, they run 'cat encrypt.txt' showing the encrypted text. Finally, they run './caesar' again with mode 1 for decryption and key 3, resulting in 'HelloSecurityATZ' shown in 'cat decrypt.txt'.

```
drk0830@drk0830-VirtualBox: ~/Secu/hw02/hw02_1
File Edit View Search Terminal Help
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ ls
caesar CaesarCipher.c plain.txt
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ cat plain.txt
HelloSecurityATZ
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ ./caesar
>> Input File Name : plain.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 0
>> Input Key : 3
File Size = 17
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ cat encrypt.txt
KhoorVhfxulwbDWC
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ ./caesar
>> Input File Name : encrypt.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 1
>> Input Key : 3
File Size = 17
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$ cat decrypt.txt
HelloSecurityATZ
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_1$
```

- 과제\_2 비제네르 암호



A screenshot of a terminal window titled 'drk0830@drk0830-VirtualBox: ~/Secu/hw02/hw02\_2'. The terminal shows the execution of a Vigenere cipher program. The user lists files, showing 'plain.txt', 'vigenere', and 'VigenereCipher.c'. They then run 'cat plain.txt' showing 'HELLO'. Next, they run './vigenere' with prompts for file name, mode (0 for Encrypt, 1 for Decrypt), and key (DB). The output is 'KFQMR' shown in 'cat encrypt.txt'. Finally, they run './vigenere' again with mode 1 for decryption and key DB, resulting in 'HELLO' shown in 'cat decrypt.txt'.

```
Activities Terminal
drk0830@drk0830-VirtualBox: ~/Secu/hw02/hw02_2
File Edit View Search Terminal Help
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ ls
plain.txt vigenere VigenereCipher.c
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ cat plain.txt
HELLO
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ ./vigenere
>> Input File Name : plain.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 0
>> Input Key : DB
File Size = 6
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ cat encrypt.txt
KFQMR
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ ./vigenere
>> Input File Name : encrypt.txt
>> Input Mode [ 0 : Encrypt, 1 : Decrypt ] : 1
>> Input Key : DB
File Size = 6
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$ cat decrypt.txt
HELLO
drk0830@drk0830-VirtualBox:~/Secu/hw02/hw02_2$
```