

## 정보보호개론 Assignment 2

2016310936 우승민

이번 과제는 AES-128(DES(plain))되어 암호화 된 cipher를 plain과 함께 비교하여, 각각에 사용된 key를 찾는 것입니다. 우선 header file과 전역변수를 보면 아래와 같습니다.

```
1 #include <stdio.h>$
2 #include <stdlib.h>$
3 #include <string.h>$
4 #include <openssl/des.h>$
5 #include <openssl/aes.h>$
6 $
7 char *plain; ^|^|^|^//plaintext$
8 char *cipher; ^|^|^|^//ciphertext$
9 char **pass; ^|^|^|^//MD5 value$
10 char **key; ^|^|^|^//MD5 key$
```

des와 aes의 암호화 알고리즘을 사용하기위해 openssl의 header파일들을 추가해주었고, plaintext, ciphertext, MD5 value, MD5 key를 저장하기 위한 공간들을 전역변수로 선언해주었습니다.

main문에서 전역변수에 선언하였던 변수들에 malloc을 사용하여 공간을 할당하는 것입니다.

```
46 int main(){
47 ^|unsigned char ch;$
48 ^|FILE *fpIn, *fpOut;$
49 ^|FILE *fpPs;$
50 $
51 ^|fpPs = fopen("passwords.txt", "r");$
52 ^|fpIn = fopen("PlaintextCiphertext.txt", "r");$
53 ^|fpOut = fopen("keys.txt", "wb");$
54 ^|$
55 ^|plain = (char*)malloc(sizeof(char)*64); ^|^|^|^//plaintext의 64byte만 저장$
56 ^|cipher = (char*)malloc(sizeof(char)*512); ^|^|^|^//ciphertext의 512byte만 저장$
57 ^|pass = (char**)malloc(sizeof(char*)*184389); ^|^|^|^//passwords.txt에 총 184389개수의 value$
58 ^|key = (char**)malloc(sizeof(char*)*184389); ^|^|^|^//passwords.txt에 총 184389개수의 key$
59 ^|int i = 0;$
60 ^|$
61 ^|for(i=0; i<184389; i++){ ^|^|^|^|^//value와 key 하나하나에 32byte크기 할당$
62 ^|^|^|^|pass[i] = (char*)malloc(sizeof(char)*32);$
63 ^|^|^|^|key[i] = (char*)malloc(sizeof(char)*32);$
64 ^|^|^|^|$
65 ^|^|^|^|$
```

plain의 경우는 처음 64byte만을 암호화한 후 비교할 것이기 때문에 64의 크기로 할당하였고, cipher는 plain의 64byte를 암호화한 결과를 여유롭게 받기위해 512의 크기로 할당하였습니다.

pass와 key의 경우는 passwords.txt의 총 key의 개수가 184389이기 때문에 각각 이에 맞게 할당하였고, 글자수 크기는 최대인 32에 맞추어서 할당해주었습니다.

다음으로 plain과 cipher에 각 크기에 맞게 입력 받는 과정입니다.

```

66 ~|for(i=0; fscanf(fpIn, "%c", &ch) != EOF && i<64; ++i){ //plaintext의 64byte -> plain에 저장$
67 ~|~|if(ch == '\n') break;$
68 ~|~|plain[i]=ch;$
69 ~|}$
70 ~|if(i<64) plain[i]='\0';$
71 ~|if(i==64){~|~|//plaintext의 크기가 64byte보다 크면 ciphertext가 올 때까지 넘김$
72 ~|~|for(i=0; fscanf(fpIn, "%c", &ch) != EOF; ++i){$
73 ~|~|~|if(ch == '\n') break;$
74 ~|~|}$
75 ~|}$
76 ~|$
77 ~|$
78 ~|int cnt = 0;$
79 ~|for(i=0; fscanf(fpIn, "%c", &ch) != EOF && i<512; ++i){ //ciphertext의 512byte -> cipher에 저장$
80 ~|~|cipher[cnt]=ch;$
81 ~|~|cnt++;$
82 ~|}$
83 ~|cipher[cnt]='\0';$
84 ~|fclose(fpIn);$

```

만약 입력 받는 plaintext가 64byte보다 작으면, for문이 종료되고 마지막 공간에 'w0'을 넣어주었습니다. 반대로 64byte보다 크면 64 byte만큼만 저장하고 이후는 'wn' 즉 cipher 문이 나올 때까지 넘겨주었습니다.

cipher도 마찬가지로 최대 512 byte만큼 저장하고 마지막에 '\0'를 넣어주었습니다. 이제는 'PlaintextCiphertext.txt'의 용도가 끝났으므로 file을 종료해주었습니다.

```

87 ^|int j=0;$
88 ^|for(i=0; i<184389; i++){^|^|//passwords.txt의 value와 key를 각각 pass와 key에 저장$
89 ^|^|int cnt = 0;$
90 ^|^|for(; fscanf(fpPs, "%c", &ch) != EOF; ++j){$
91 ^|^|^|if(ch == ' ') break;$
92 ^|^|^|pass[i][cnt++] = ch;$
93 ^|^|}$
94 ^|^|cnt = 0;$
95 ^|^|for(; fscanf(fpPs, "%c", &ch) != EOF; ++j){$
96 ^|^|^|if(ch == '\n') break;$
97 ^|^|^|key[i][cnt++] = ch;$
98 ^|^|}$
99 $
100 ^|}$
101 ^|fclose(fpPs);$

```

마찬가지로 MD5의 value와 key값도 for문을 통해서 각각 pass array와 key array에 순서대로 저장 하였습니다. 이것 또한 저장이후 'passwords.txt'를 사용할 필요가 없음으로 종료해주었습니다.

다음은 des 암호화 이전에 input할 plain문을 des\_in array에 옮겨주는 과정입니다.

```
103 ^|unsigned char des_in[8][8];^|^|^|^//des_ebc 암호화의 input 생성 64byte를 8 * 8byte로 쪼갬$
104 ^|^|$
105 ^|^|^for(int i=0; i<8; i++) memset(des_in[i],0,8);$
106 $
107 ^|^|$
108 ^|^|^for(int i=0; i<8; i++) {$
109 ^|^|^|^for(int j=0; j<8; j++) {$
110 ^|^|^|^|^if(plain[i*8+j]) des_in[i][j] = plain[i*8+j];$
111 ^|^|^|^|^else des_in[i][j]=0;$
112 ^|^|^|^}$
113 ^|^|^}$
```

openssl의 des 함수가 8byte씩 나누어서 이루어졌기 때문에 64byte의 plain을 각각 8\*8byte로 나누어서 옮겨주었습니다.

이제는 본격적으로 암호화하는 과정입니다. 제가 구현한 code는 des와 aes 모두 사용된 key의 쌍을 찾기 위해서 for문을 2번 사용하여 바깥쪽 for문은 aes 암호화를 안쪽 for문은 des 암호화를 진행합니다. 각 for문은 모든 MD5의 key를 확인하기위해 184389번씩 반복합니다.

$$AES\_CBC128_{k_i} \left( DES_{ECB_{k_j}}(des_{in}) \right)$$

```
117 ^|^|^for(int i=0; i<184389; i++){^|^|^|^//밖의 for문은 aes_cbc 암호화 반복문$
118 ^|^|^|^|^unsigned char aes_key[16];$
119 ^|^|^|^|^char ch3[3];$
120 ^|^|^|^|^int num2;$
121 ^|^|^|^|^for(int k=0; k<16; k++){^|^|^|^|^//aes 암호화의 key값 저장 -> aes_key $
122 ^|^|^|^|^|^|^|^|^|^|^ch3[0] = pass[i][k*2+0];$
123 ^|^|^|^|^|^|^|^|^|^|^ch3[1] = pass[i][k*2+1];$
124 ^|^|^|^|^|^|^|^|^|^|^ch3[2] = '\0';$
125 ^|^|^|^|^|^|^|^|^|^|^num2 = strtol(ch3, NULL, 16);$
126 ^|^|^|^|^|^|^|^|^|^|^aes_key[k] = num2;$
127 ^|^|^|^|^|^}$
128 ^|^|^|^|^|$
129 ^|^|^|^|^for(int j=0; j<184389; j++){^|^|^|^|^//안의 for문은 des_ebc 암호화 반복문$
130 ^|^|^|^|^|^|^|^|^|^|^char ch2[3];$
131 ^|^|^|^|^|^|^|^|^|^|^int num1;$
132 ^|^|^|^|^|^|^|^|^|^|^|^IDES_cblock key1;$
133 ^|^|^|^|^|^|^|^|^|^|^|^for(int k=0; k<8; k++){^|^|^|^|^|^|^|^|^|^|^//des 암호화의 key값 저장 -> key1$
134 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^ch2[0] = pass[j][k*2+0];$
135 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^ch2[1] = pass[j][k*2+1];$
136 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^ch2[2] = '\0';$
137 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^num1 = strtol(ch2, NULL, 16);$
138 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^key1[k] = num1;$
139 ^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^}$
```

각 for문의 시작할 때는 aes\_key에는 aes\_cbc에서 사용할 key를 key1에는 des\_ebc에서 사용할 key를 저장합니다. pass array에는 key가 char형으로 저장되어 있으므로 strtol함수를 사용하였습니다.

이후 des\_ecb 의 암호화를 진행합니다. des\_out 도 des\_in 과 마찬가지로 8\*8byte 형태로 만들어주었습니다.

```
141 ^|_|^|_|unsigned char des_out[8][8];^|_|//des 암호화 output 생성$
142 ^|_|^|_|for(int i=0; i<8; i++) memset(des_out[i],0,8);$
143 ^|_|^|_|$
144 ^|_|^|_|DES_key_schedule keysched;$
145 $
146 ^|_|^|_|DES_set_key(&key1, &keysched);$
147 ^|_|^|_|for(int k=0; k<8; k++) {^|_|//des 암호화과정 8byte나눈 것을 8번 반복 des_in -> des_out$
148 ^|_|^|_|DES_ecb_encrypt((DES_cblock*)des_in[k],(DES_cblock*)des_out[k], &keysched, DES_ENCRYPT);$
149 ^|_|^|_|}$
150 $
151 ^|_|^|_|unsigned char aes_input[64];^|_|//aes 암호화 input 생성$
152 ^|_|^|_|memset(aes_input,0,64);$
153 ^|_|^|_|for(int k=0; k<8; k++){
154 ^|_|^|_|for(int l=0; l<8; l++){^|_|//des의 output을 aes input으로 옮김$
155 ^|_|^|_|^|_|aes_input[k*8+l] = des_out[k][l];$
156 ^|_|^|_|}
157 ^|_|^|_|}$
158 ^|_|^|_|}$
```

DES\_ecb\_encrypt 함수를 8번 반복하여 des\_out이 나오면 aes\_cbc 암호화를 진행하기위해 다시 aes\_input에 des\_out의 값을 옮겨주었습니다.

바로 이어서 aes\_cbc의 암호화를 진행합니다. initialization vector를 NULL값으로 생성해준 후 암호화를 진행하였습니다. 이어서 바로 암호화된 enc\_out을 base64 형식으로 변환해주었습니다.

```
161 ^|_|^|_|unsigned char iv[AES_BLOCK_SIZE];^|_|//initialization vector 생성 NULL으로 사용$
162 ^|_|^|_|memset(iv, 0x00, AES_BLOCK_SIZE);$
163 ^|_|^|_|$
164 ^|_|^|_|unsigned char enc_out[sizeof(aes_input)];^|_|//aes 암호화 output 생성$
165 ^|_|^|_|memset(enc_out, 0, sizeof(enc_out));$
166 ^|_|^|_|AES_KEY key2;$
167 ^|_|^|_|AES_set_encrypt_key(aes_key, sizeof(aes_key)*8, &key2); // aes_key -> key2로 변환$
168 $
169 ^|_|^|_|// aes_cbc 암호화 aes_input -> enc_out$
170 ^|_|^|_|AES_cbc_encrypt(aes_input, enc_out, sizeof(aes_input), &key2, iv, AES_ENCRYPT);$
171 ^|_|^|_|int k=0;$
172 ^|_|^|_|$
173 ^|_|^|_|int encode_size = 4 * ((sizeof(enc_out)+ 2) / 3);^|_|//base64 변환 공간 할당$
174 ^|_|^|_|$
175 ^|_|^|_|unsigned char base_out[encode_size];$
176 ^|_|^|_|$
177 ^|_|^|_|base64_encode(enc_out, base_out, sizeof(enc_out));^|_|//base64 변환 enc_out -> base_out$
```

base64로 변환된 크기는 8bit를 6bit 형식으로 변환하는 것이기 때문에  $\frac{4}{3}$  배 해주었습니다. base64\_encode 함수는 openssl의 header 파일에서 base64 encoding을 찾지 못해 직접 구현하였습니다.

```
13 int base64[64] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/'};$
```

우선 base64 array에 base64에서 사용되는 문자를 모두 입력해주었습니다.

base64\_encode 함수 code입니다. 변환할 plain과 변환 후 저장될 encode, plain의 크기를 인자로 넘겨주었습니다.

```

15 void base64_encode(unsigned char *plain, unsigned char *encode, int plain_size){$
16 $
17 ^|int i=0;$
18 ^|int j=0;$
19 ^|$
20 ^|while(i < plain_size-2){^|^|^|^|^|^|^|^|^|^// 3개의 plain -> 4개의 encode$
21 ^|^|encode[j++] = base64[plain[i] >> 2];$
22 ^|^|encode[j++] = base64[((plain[i] << 4) & 0x30) | (plain[i+1] >> 4)];$
23 ^|^|encode[j++] = base64[((plain[i+1] << 2) & 0x3c) | (plain[i+2] >> 6)];$
24 ^|^|encode[j++] = base64[(plain[i+2] & 0x3f)];$
25 ^|^|i+=3;$
26 ^|}$
27 ^|$
28 ^|if(i == plain_size-2){^|^|^|^|^|^|^|^|^|^// 2개의 plain + padding -> 4개의 encode$
29 ^|^|encode[j++] = base64[plain[i] >> 2];$
30 ^|^|encode[j++] = base64[((plain[i] << 4) & 0x30) | (plain[i+1] >> 4)];$
31 ^|^|encode[j++] = base64[((plain[i+1] << 2) & 0x3c)];$
32 ^|^|encode[j++] = '=';$
33 ^|$
34 ^|}$
35 ^|$
36 ^|else{^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^|^// 1개의 plain + padding -> 4개의 encode$
37 ^|^|encode[j++] = base64[plain[i] >> 2];$
38 ^|^|encode[j++] = base64[((plain[i] << 4) & 0x30)];$
39 ^|^|encode[j++] = '=';$
40 ^|^|encode[j++] = '=';$
41 ^|}$
42 $
43 ^|$

```

기본적으로 base64 encoding은 8bit를 6bit로 변환하는 것임으로 3개 \* 8bit -> 4개 \* 6bit 형식으로 이루어집니다. 그렇기에 대부분의 변환은 while문을 통해 이루어집니다. 이외에 plain의 bit가 3으로 나누어 떨어지지 않을 경우를 대비해 if문과 else문을 통해 나머지 상황도 encoding 해주었습니다.

이후 다시 main문으로 넘어와서 암호화가 완료된 bit들을 cipher와 비교하는 과정만이 남았습니다. 우선 저는 40byte만큼 동일하게 암호화가 되었으면 올바른 key들을 찾았다고 생각하여 최대 비교를 40으로 설정하였고, 만약 base\_out이나 cipher문의 크기가 더 작을 경우에는 더 작은 값으로 바꾸어 주었습니다.

```
177 ^^^^base64_encode(enc_out, base_out, sizeof(enc_out));^^^base64 변환 enc_out -> base_out$
178 ^^^^$
179 ^^^^k = 0;$
180 ^^^^int diff=40;^^^//암호화 후 비교크기를 최대 40으로 설정$
181 ^^^^$
182 ^^^^while(base_out[k++]);^^^$
183 ^^^^if(diff > k) diff = k;^^^만약 base_out이나 cipher 중에 size가 40보다 작은 것이 있으면 변경$
184 ^^^^k = 0;$
185 ^^^^$
186 ^^^^while(cipher[k++]);$
187 ^^^^if(diff > k) diff = k;$
188 ^^^^k = 0;$
```

diff값이 올바르게 암호화 되었다고 판단하는 최소값인 것입니다.

최종적으로 base\_out과 cipher를 비교하는 과정입니다.

```
190 ^| | | | |while(base_out[k] && base_out[k] == cipher[k]){ //diff길이만큼 base_out과 cipher를 비교$
191 ^| | | | |if(k > diff){ | | | |//만약 diff까지 왔으면 암호화가 제대로 이루어짐 -> key값 찾음$
192 ^| | | | |char x[40];$
193 ^| | | | |int l=0;$
194 ^| | | | |while(key[j][l]){ | | | |//x에 key1, key2 차례로 옮긴 후 fpOut에 출력$
195 ^| | | | |x[l] = key[j][l];$
196 ^| | | | |l++;$
197 ^| | | | |}$
198 ^| | | | |x[l]='#0';$
199 ^| | | | |fwrite(x,l, 1, fpOut);$
200 ^| | | | |ch = '\n';$
201 ^| | | | |fwrite(&ch, sizeof(ch), 1, fpOut);$
202 ^| | | | |l=0;$
203 ^| | | | |while(key[i][l]){ $
204 ^| | | | |x[l] = key[i][l];$
205 ^| | | | |l++;$
206 ^| | | | |}$
207 ^| | | | |x[l]='#0';$
208 ^| | | | |fwrite(x,l, 1, fpOut); | | | |$
209 ^| | | | |fclose(fpOut);$
210 ^| | | | |}$
211 ^| | | | |for(int l=0; l<184389; l++){ | | | |// malloc으로 할당해준 공간 free$
212 ^| | | | |free(pass[l]);$
213 ^| | | | |free(key[l]);$
214 ^| | | | |}$
215 ^| | | | |free(pass);$
216 ^| | | | |free(key);$
217 ^| | | | |free(plain);$
218 ^| | | | |free(cipher);$
219 $
220 ^| | | | |return 0; | | | |// 빠른 실행 종료를 위해 바로 return$
221 ^| | | | |}$
222 ^| | | | |k++;$
223 ^| | | | |}$
224 ^| | | | |}$
225 $
226 | | | | |$
```

위에서 최소한으로 설정하였던 diff값의 기준을 통과하였다면, x array에 암호화에 사용하였던 key 값을 차례로 입력한 후 fwrite를 통해 'Keys.txt'에 입력하게 해주었습니다. 이후 모든 과정이 끝났음으로 for문을 계속 진행할 필요가 없기에 malloc으로 할당해주었던 공간들을 모두 free해준 후 return을 통해 main문이 바로 종료되게 해주었습니다.

```
230 ^| | | | |fclose(fpOut); | | | |//만약 key를 찾지 못했어도 할당한 공간 free 후 종료$
231 | | | | |for(i=0; i<184389; i++){ $
232 ^| | | | |free(pass[i]);$
233 ^| | | | |free(key[i]);$
234 ^| | | | |}$
235 ^| | | | |free(pass);$
236 ^| | | | |free(key);$
237 ^| | | | |free(plain);$
238 ^| | | | |free(cipher);$
239 $
240 ^| | | | |return 0;$
241 | | | | |}$
```

만약 for문을 모두 진행했음에도 찾지 못하면, 그대로 할당 공간들을 free한 후 종료하였습니다.

```

1 000099656c16f8e84f50d10a65055c3d coders$
2 0001245350b5eb0a1548fc6d27d3b4d1 piwtf$
3 000132637f568134a2b61b1055994f6c miguel10$
4 00023aa1be6d219fd5125d6787f782b0 r0tflgsa$
5 0002cf33af394c5bc6e51638f05528ee carbonmegat
6 0002f9e500c69f3a093bbc7d592c5399 music1$
7 0003fac10771961b4122f63b235deca1 402760$
8 000445f1e822fa2b6229cd57b4f9bbc4 termopan$
9 00046fd5d95fe096d9a0b9c42008e661 supervegeta
10 00048bd75ceb26c6f9c8b35d5cd1652a 11291985$
11 00049e4d33f0a395c387652cf13840de jamie123$
12 0004c6d780086405acefad97c01be4ea 25741662$
13 0004cb8f65f65690624d4b697f93ea98 netto1$
14 0004d0b59e19461ff126e3a08a814c33 1970$
15 000500fbd0fcb5e905a5f84c49d683a3 Hyperfx$
16 0005bda5f81683870a3e69f5f6498da2 ZBNgASLo$
17 00063c3ed24c556a43e100f2ac2bf3d1 jkex90$
18 00067c58d0ab6f3853e0f588363b1559 496328$
19 0006f80da1dab7b03fcb338dbe1a4017 daedelus9$
20 00075afb16c7a466b856ecc064eb569d 8218361$
21 0007a80c985385bb86b7a780862d2b25 presov$
22 0007b5b98c145c835d3350814cf49ab5a prizrak$
23 0007c363cd7b17d23443f12d4a0ffc26 tazeem$
24 0008b17e5903bfd465823a1e3f5b7f0 fraudster$
25 0008b267e8ddde331730bdf9b9ce74a icywind$
26 0008c0dab2ff4c2f7e84fcab7eb8db6a peropero$
27 0009111dct5c059fbb7a49d525f66acb ninap994$
28 00093a31c97e99bd1cd899785ac13dc0 nokia7110$
29 00097f5db717ccc9b8892b12f69029e4 _april11a$
30 0009d6fa1234c3f5dd5c4ef3e8c0ba1e 3D4YFBV3$
31 0009fba66fb053a8efabb9d74ceab52b koshan$
32 000ab0c3aa0703646a3a13d81678f6bf lam1root$
33 000b3577d16ef5c30bcc627e91469da7 007squall$
34 000bef5dec4bd0749344ff9184f21063 Gandalf02$
35 000c689f901d37703104303efcbafba0 londone$
36 000cbcf85ca3583bf820c61bb4186c3 pureza$
37 000ce7cbd549ab3ac00d382bf582f348 hammed$
38 000d2ef9c5c431f16712f3bc0fa8f1ea 255YMOQP$

```

passwords.txt

현재 'passwords.txt'에 왼쪽과 같이 저장 되어있고

'PlaintextCiphertext.txt'에는 아래와 같이 저장되어 있습니다.

```

1 SKKU is the top university in the world          adsf          adsf
          asdf$
2 CBMsZ223gfHe6AH6I+1IEjpXxjFIupBrGYZ8CDYYr9WJj4j0cMuL8uAA/Yxr9pNK          asdfwefsav
          asdvwrf avsadf          waefsadg$
~
PlaintextCiphertext.txt          1,1

```

다음으로 code를 compile한 후 실행하면 다음과 같이 keys.txt가 생성되는 것이 확인 가능합니다.

```

saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ ls
2016310936_hw2.c      Assignment_02.pdf      new                    passwords.txt
2016310936_hw2.docx  PlaintextCiphertext.txt  openssl-1.0.1g       '-$16310936_hw2.docx'
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ vi passwords.txt
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ vi PlaintextCiphertext.txt
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ gcc -Wall 2016310936_hw2.c -o 2016310
936_hw2 -lcrypto
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ ./2016310936_hw2
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ ls
2016310936_hw2      Assignment_02.pdf      new                    passwords.txt
2016310936_hw2.c    PlaintextCiphertext.txt  openssl-1.0.1g       '-$16310936_hw2.docx'
2016310936_hw2.docx  keys.txt              passwords.txt
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$ vi keys.txt
saumsung@DESKTOP-IC1E19F:/mnt/c/Users/wsm42/Desktop/4-1/정보개/2$

```

keys.txt를 보면 아래와 같이 key가 출력되는 것을 확인할 수 있습니다.

```

1 coders$
2 piwtf$
~
~
~
~
~
~
keys.txt

```