Assignment2

2016310932 배현웅

1. Overview

   A. To save hash value and key value, I made struct data structure. The code reads them from passwords.txt

   B. From PlaintextCiphertext.txt. read plain text and cipher text. (function : read_text())

   C. First of all, from line 1st to line 184,389th, code encrypt by DES and AES, encrypted text would be converted to base64 encoding using into_base64() function. Then, check whether it is same as cipher text given. I use Brute-force attack, Big O would be square of n

2. Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/aes.h>
#include <openssl/des.h>
#include <openssl/bio.h>
#include <openssl/buffer.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>
#include <openssl/evp.h>

#define line_num 184389
#define BUFSIZE 64
#define key_len 16
#define MAX_textlen 102401

int file_size;
typedef unsigned char uc;
typedef struct p_ham{
    unsigned char key[key_len];
    unsigned char p[36];
}store;
store* ham;
uc* base64_in;
//brute forece 1002_1832
void read_passwd(FILE* passwd_file);
void read_text(FILE* f,uc* plain,uc* cipher);
void into_base64(uc* in, int len);
```

Line 12 : Since passwords line length is 184,389, to save password & hash key value, line_num is defined

Line 13,14 to store key value& buffer length, they are defined

Line 15 : Maximum plaintext length is 100KB(100*1024 = 102400), Maxtextlen is defined

Line 19-23 : To store password value, I made store struct data structure.

Line 24 : to store base64-encoded value, I use it.

- Read_passwd()

```
131  void read_passwd(FILE* passwd_file){
132      char buf[BUFSIZE+4];
133      unsigned char p[36];
134      unsigned char hash[33];
135      unsigned char key[key_len];
136      int index=0;
137      while(fgets(buf,sizeof(buf),passwd_file) !=NULL){
138          //divide buf -> hash & p
139          strcpy(p,(buf+33));
140          p[strlen(p)-1] = '\0';
141          strcpy(ham[index].p,p);
142
143          buf[32]='\0';
144          strcpy(hash,buf);
145          //printf("hash : %s\n",hash);
146          for(int j=0; j<key_len; ++j){
147              int a[2];
148              for(int k=0; k<2; ++k){
149                  int pos = 2*j +k;
150                  if(hash[pos] >= '0' && hash[pos] <= '9'){
151                      a[k] = hash[pos]-'0';
152                  } else {
153                      a[k] = hash[pos]-'a'+10;
154                  }
155              }
156              key[j] = a[0]<<4;
157              key[j] += a[1];
158          }
159
160          memcpy(ham[index].key,key,key_len);
161          index++;
162      }
163  }
```

In this function, it reads every line of passwords.txt text which is consists of md5 hash value and

password value. Then MD5 hash value which is saved as unsigned char converted into int value every 2 bytes(hash -> int a[2] -> key[]).After divided into md5 value and password value, they are into ham struct which saved for password value and hash value.

- Read_text

```
126  void read_text(FILE* f,uc* plain,uc* cipher){
127      fgets(plain,MAX_textlen,f);
128      fgets(cipher,MAX_textlen+1,f);
129  }
130
```

Read_text function just read PlaintextCiphertext.txt, then divided into plaintext and cipher text and save into plain, cipher data which defined in main function

- Into_base64

```
104  void into_base64(uc* in, int len){
105      BIO *mem, *v_64;
106      BUF_MEM *ptr;
107
108      v_64 = BIO_new(BIO_f_base64());
109      mem = BIO_new(BIO_s_mem());
110      v_64 = BIO_push(v_64, mem);
111
112      BIO_set_flags(v_64,BIO_FLAGS_BASE64_NO_NL);
113      BIO_write(v_64,in,len);
114      BIO_flush(v_64);
115      BIO_get_mem_ptr(v_64,&ptr);
116
117      BIO_set_close(v_64, BIO_NOCLOSE);
118      memcpy(base64_in, ptr->data, ptr->length);
119      base64_in[ptr->length]='\0';
120      BIO_free_all(v_64);
121      BUF_MEM_free(ptr); // free code needed
122
123      return;
124  }
```

To convert AES-128(DES(m)) encrypted value, I made into_base64 function with OPENSSL library bio.h, buffer.h, and etc. With BIO function, the encoding of base64 continues. Not to be divded into 64 bytes, I use BIO_set_close with BIO_NOCLOSE.

- Main.c

```
30  int main(){
31      // buffer for encrypt
32      ham = calloc(line_num,sizeof(store));
33      uc* plain,*cipher;
34      uc des[MAX_textlen], aes[MAX_textlen];
35      plain = calloc(MAX_textlen, sizeof(uc));
36      cipher = calloc(MAX_textlen+1, sizeof(uc));
37      base64_in = calloc(MAX_textlen, sizeof(uc));
```

This section is for dynamic allocation text value for encryption and so on.

```
39      //read password file
40      FILE *passwd_file;
41      passwd_file = fopen("passwords.txt","r");
42      read_passwd(passwd_file);
43      fclose(passwd_file);
44
45      //read plaintextciphertext.txt
46      FILE* input_file;
47      input_file = fopen("PlaintextCiphertext.txt","r");
48      read_text(input_file,plain,cipher);
49      fclose(input_file);
```

Line 40-43 : Read passwords.txt value to store password value into ham structure.

Line 46-49 : read plaintext and ciphertext.

```
51      //DES encrypt
52      int plain_len = strlen(plain)-1;
53      int idx1,idx2; // key for idx
54      idx1 = idx2 = 0;
55      int iter = ((plain_len/16*16 == plain_len))?(plain_len):((plain_len/16+1)* 16);
56      if(plain_len < iter){
57          memset(plain+plain_len,0x0,iter-plain_len);
58          plain[iter] ='\0';
59      }
60      // elements for key setting
61      uc key1[16],key2[16];
62      DES_key_schedule keysched;
63      AES_KEY enc_key_128;
64      unsigned char iv[AES_BLOCK_SIZE];
```

Line 52-59 : to pad the 0 value for unaligned text with 16 bytes. Check plaintext length which is not included with newline. Then pad 0 value with memset.

```
66      while(idx1<line_num){
67          memcpy(key1,ham[idx1].key,key_len);
68          DES_set_key((DES_cblock *)key1, &keysched);
69          DES_set_key_checked((DES_cblock *)key1, &keysched);
70          for(int i=0; i<iter; i+=8){
71              DES_ecb_encrypt((DES_cblock *)(plain+i),(DES_cblock *)(des+i), &keysched, DES_ENCRYPT);
72          }
```

To break cipher text, the code try every values which is in passwords.txt. So, plain text is encrypted with DES.

```
73          idx2=0;
74          while(idx2<line_num){
75              memset(iv, 0x00, AES_BLOCK_SIZE);
76              memcpy(key2,ham[idx2].key,key_len);
77
78              AES_set_encrypt_key(key2, 16*8, &enc_key_128);
79              AES_cbc_encrypt(des,aes, strlen(des) , &enc_key_128, iv, AES_ENCRYPT);
80              into_base64(aes,iter);
81
82              //check whether ciphered plaintext vs ciphertext
83              if(strncmp(base64_in,cipher,strlen(cipher)) == 0) {
84                  goto exit;
85              }
86              idx2++;
87          }
88          idx1++;
89      }
```

Then encrypt DES encrypted text with AES-128 encryption. To AES encryption, I set iv to be 0, and copy possible key value from ham. Then encrypt (line 78-79). After that, this text would be encoded with base64. (line 80)

Finally, compare with ciphertext which is from PlaintextCiphertext.txt. If they are same, brute-force attack is finished (goto exit label)

```
90  exit:
91      free(plain);
92      free(cipher);
93
94      FILE* output = fopen("keys.txt","w");
95      fprintf(output,"%s\n%s",ham[idx1].p,ham[idx2].p);
96      fclose(output);
97
98      free(base64_in);
99      free(ham);
100     return 0;
101 }
```

To finish code, free all dynamic allocated value, and write key value into keys.txt(output file)