정보보호

HW08 : RSA Cipher

제 출 일	2018년 11월 05일
담당교수	류재철
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 코드분석

■ 기본셋팅

```c
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <openssl/bio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define BLOCK_SIZE 128
#define KEY_LENGTH 1024
#define PUB_EXP 3

int main()
{
    char inputFileName[256];//입력 되는 파일 명
    char encoutputFileName[256];//암호화 출력되는 파일
    char decoutputFileName[256];//다시 복호화된 것을 출력하는 파일
    char in_buff[KEY_LENGTH]={0, };
    char out_buff[KEY_LENGTH]={0, };//RSA과정중 입력과 출력으로 사용할 임시 버퍼
    FILE *input_FD;
    FILE *output_FD;
    printf(">> Input file name : ");
    scanf("%s",inputFileName);//입력 받을 파일명 입력

    sprintf(encoutputFileName,"plain.enc");
    sprintf(decoutputFileName,"plain.enc.dec");
    //RSA 암호화 준비 단계
```

블록사이즈는 128로 키길이는 1024로 하여 RSA1024로 동작하도록 하였다.
입력할 파일명을 입력받고 정해진 출력문에 암호문과 복호화문을 저장하는 식으로 구현하였다.

```
RSA *keypair = RSA_generate_key(KEY_LENGTH,PUB_EXP,NULL,NULL);
//앞서 지정한 키 길이로 키 페어 생성
BIO *pri = BIO_new(BIO_s_mem());
BIO *pub = BIO_new(BIO_s_mem());
//BIO라는 객체를 초기화 하는 과정
PEM_write_bio_RSAPrivateKey(pri,keypair,NULL,NULL,0,NULL,NULL);
PEM_write_bio_RSAPublicKey(pub,keypair);

size_t pri_len=BIO_pending(pri);
size_t pub_len=BIO_pending(pub);

char *pri_key=malloc(pri_len+1);
char *pub_key=malloc(pub_len+1);

BIO_read(pri,pri_key,pri_len);
BIO_read(pub,pub_key,pub_len);
//BIO정보를 읽어 문자열배열에 적재
pri_key[pri_len] = '\0';
pub_key[pub_len] = '\0';

FILE *key_file;
key_file = fopen("prkey.pem","w");
fprintf(key_file,"%s",pri_key);
fclose(key_file);
//개인키파일 생성
key_file = fopen("pukey.pem","w");
fprintf(key_file,"%s",pub_key);
fclose(key_file);
//공용키 파일 생성
key_file = fopen("rsa.key","w");
RSA_print_fp(key_file,keypair,0);
fclose(key_file);
//키페어 파일 생성
```

키쌍을 생성 후 개인키 파일과 공용키 파일을 생성하는 과정

■ 암호화 복호화 단계

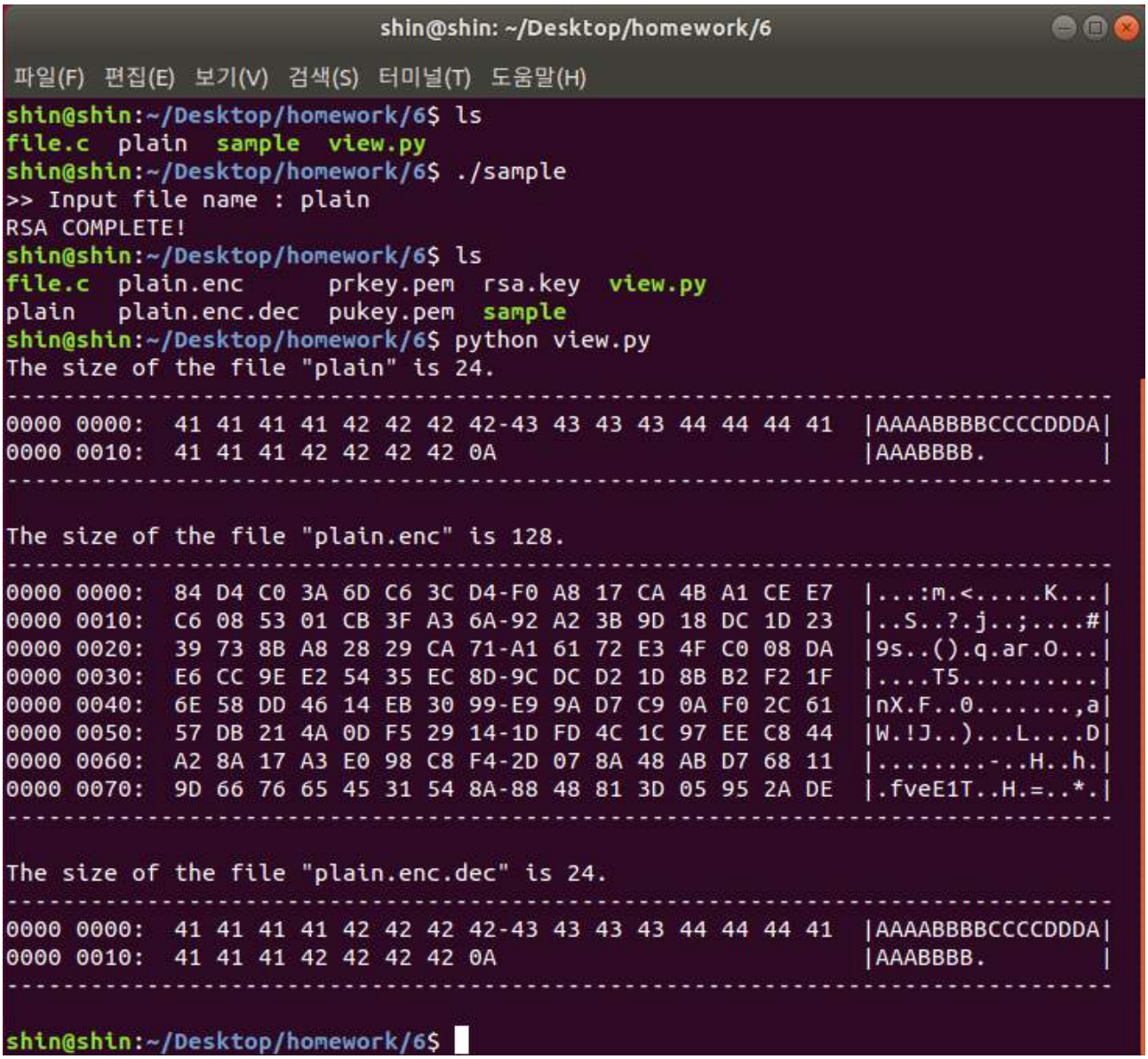
```
    input_FD=fopen(inputFileName,"rb");
    output_FD=fopen(encoutputFileName,"wb");//먼저 암호화부터
    int t=0;//암,복호화후 만들어진 파일의 길이를 저장하기위한 변수
    int k=0;//얼마나 파일을 읽었는지 저장하기위한 변수
    while(0<(k=fread(&in_buff,sizeof(char),117,input_FD))){//암호화시에는 블록사이
즈가 128일때 최대 입력가능한 117단위 만큼 나눠서 읽으면 된다.
    t=RSA_public_encrypt(k,in_buff,out_buff, keypair, RSA_PKCS1_PADDING);
    if(ERR_get_error()){
    printf("RSA_public_encrypt failure: %d ret:%d\n", ERR_get_error(), t);
         return -1;
       }//출력(t값)을 항상 블록사이즈로 나와야하는데 에러시에는 -1를 반환
    fwrite(&out_buff,sizeof(char),t,output_FD);
    }//나온 출력값을 파일에 저장

    memset(in_buff,0,sizeof(char)*BLOCK_SIZE);
    memset(out_buff,0,sizeof(char)*BLOCK_SIZE);
    //버퍼 초기화
    fclose(output_FD);
    fclose(input_FD);

    input_FD=fopen(encoutputFileName,"rb");
    output_FD=fopen(decoutputFileName,"wb");
    //복호화를 위해 다시 파일을 설정
    while(fread(&in_buff,sizeof(char),BLOCK_SIZE,input_FD)){//복호화시에는 무조건
블록사이즈 단위로 읽으면 된다.
    t=RSA_private_decrypt(BLOCK_SIZE,in_buff,out_buff, keypair, RSA_PKCS1_PADDING)

    fwrite(&out_buff,sizeof(char),t,output_FD);
    }//나온 길이 만큼 쓰기를 진행
    fclose(output_FD);
    fclose(input_FD);
    printf("RSA COMPLETE!\n");
    return 0;
}
```

RSA1024이고 RSA_PKCS1_PADDING이라 최대 117문자열까지 입력 후 암호화가 가능하다.
그래서 117개 단위로 읽도록 하였고 읽은 117개 이하로 읽었을 때를 대비해 읽은 수를 k에 저장해서 RSA_public_encrypt할 때 길이를 입력할 때 k를 이용하였다.
RSA_public_encrypt해서 나온 리턴값은 만든 암호문의 길이 임으로 fwrite할 때 사용 하면된다.

사용한 버퍼는 초기화하고 파일을 닫았다가 복호화 하기 위한 셋팅으로 다시 열어준다.
복호화시에는 무조건 블록사이즈 단위로 읽으면 되고 암호화와 마찬가지로
RSA_private_decrypt에서 리턴 되는 수는 복호화 완료한 길이라서 t를 이용하여서 fwrite해준다

■ 실행결과 화면

1. 블록사이즈(128) 이하



중간에 ls를 한 번 더 입력하여 키과 암 복호화 파일이 생성된 걸 확인

2. 블록 사이즈 이상 입력했을때

```
shin@shin: ~/Deskt
파일(F)  편집(E)  보기(V)  검색(S)  터미널(T)  도움말(H)
shin@shin:~/Desktop/homework/6$ ./sample
>> Input file name : plain
RSA COMPLETE!
shin@shin:~/Desktop/homework/6$ python view.py
The size of the file "plain" is 132.
--------------------------------------------------------------------------------
0000 0000:  41 41 41 41 42 42 42 42-43 43 43 43 44 44 44 41  |AAAABBBBCCCCDDDA|
0000 0010:  41 41 41 42 42 42 42 43-43 43 43 44 44 44 41 41  |AAABBBBCCCCDDDAA|
0000 0020:  41 41 41 42 42 42 42 43-43 43 43 44 44 44 41 41  |AAABBBBCCCCDDDAA|
0000 0030:  41 41 42 42 42 42 43 43-43 43 44 44 44 41 46 43  |AABBBBCCCCDDDAFC|
0000 0040:  43 44 44 44 41 41 41 41-42 42 42 43 43 43 43 44  |CDDDAAAABBBCCCCD|
0000 0050:  44 44 41 41 42 42 42 43-43 43 43 44 44 41 44 44  |DDAABBBCCCCDDADD|
0000 0060:  44 41 41 64 64 64 64 41-41 42 42 42 43 64 64 64  |DAAddddAABBBCddd|
0000 0070:  64 64 64 64 61 61 73 64-73 61 7A 78 63 7A 63 73  |ddddaasdsazxczcs|
0000 0080:  71 77 65 0A                                      |qwe.            |
--------------------------------------------------------------------------------

The size of the file "plain.enc" is 256.
--------------------------------------------------------------------------------
0000 0000:  90 9B E8 06 5B AD F5 D5-14 33 93 B9 42 13 40 46  |....[....3..B.@F|
0000 0010:  AB 49 6F 41 45 E1 64 4C-46 E3 D0 5E 09 AD 9F 0D  |.IoAE.dLF..^....|
0000 0020:  B4 3C 4A E4 02 4B 97 C7-33 84 4C 2F 61 8A 98 DE  |.<J..K..3.L/a...|
0000 0030:  B4 45 F9 87 C5 71 CA F7-07 34 86 4B 15 4B 6A 9C  |.E...q...4.K.Kj.|
0000 0040:  2F 37 0A DD 9B 94 F6 64-7F B0 A5 1E 46 07 72 B5  |/7.....d...F.r.|
0000 0050:  68 87 E9 66 D4 CF ED 46-D0 D3 7F F6 1A 68 F3 40  |h..f...F....h.@|
0000 0060:  4C 80 69 93 6F 14 9D A3-DB 16 A6 39 83 72 C8 8F  |L.i.o......9.r..|
0000 0070:  7E E4 C2 71 DF 13 23 4B-1F 7D B0 5A ED AF 17 9C  |~..q..#K.}.Z....|
0000 0080:  9C AD 44 D1 9C BE E8 9C-0E 26 9E A5 DF 73 FA E1  |..D......&...s..|
0000 0090:  E6 75 E4 07 52 F5 B0 AD-E4 73 9D 55 F2 06 88 DB  |.u..R....s.U....|
0000 00A0:  77 3D 57 DD 07 50 9D 05-1D A1 CD 59 A8 B7 3D F4  |w=W..P.....Y..=.|
0000 00B0:  A3 23 02 5F 74 57 6C 28-69 59 44 76 77 3B 23 25  |.#._tWl(iYDvw;#%|
0000 00C0:  CF 46 27 6F FD DB F0 C3-A0 C7 67 56 4B E6 10 66  |.F'o......gVK..f|
0000 00D0:  C4 9E 05 E4 4F E3 52 21-A9 A0 E2 78 50 75 0B 62  |....O.R!...xPu.b|
0000 00E0:  4C C3 C8 94 E1 5A C2 95-1C 75 A0 8C C3 E9 84 A0  |L....Z...u......|
0000 00F0:  02 A6 03 44 F3 CD FC 19-A3 85 4A E4 10 72 32 FD  |...D......J..r2.|
--------------------------------------------------------------------------------

The size of the file "plain.enc.dec" is 132.
--------------------------------------------------------------------------------
0000 0000:  41 41 41 41 42 42 42 42-43 43 43 43 44 44 44 41  |AAAABBBBCCCCDDDA|
0000 0010:  41 41 41 42 42 42 42 43-43 43 43 44 44 44 41 41  |AAABBBBCCCCDDDAA|
0000 0020:  41 41 41 42 42 42 42 43-43 43 43 44 44 44 41 41  |AAABBBBCCCCDDDAA|
0000 0030:  41 41 42 42 42 42 43 43-43 43 44 44 44 41 46 43  |AABBBBCCCCDDDAFC|
0000 0040:  43 44 44 44 41 41 41 41-42 42 42 43 43 43 43 44  |CDDDAAAABBBCCCCD|
0000 0050:  44 44 41 41 42 42 42 43-43 43 43 44 44 41 44 44  |DDAABBBCCCCDDADD|
0000 0060:  44 41 41 64 64 64 64 41-41 42 42 42 43 64 64 64  |DAAddddAABBBCddd|
0000 0070:  64 64 64 64 61 61 73 64-73 61 7A 78 63 7A 63 73  |ddddaasdsazxczcs|
0000 0080:  71 77 65 0A                                      |qwe.            |
--------------------------------------------------------------------------------
```

느낀점 : 블록사이즈와 RSA의 최대입력에 대해서 배워서 재밌었다.