

信息安全导论实验2-密码学及其应用

PB18111707 吕瑞

信息安全导论实验2-密码学及其应用

实验要求

实验步骤

编写 encfile.cpp

验证程序正确性

实验要求

修改例程 cryptoDemo.cpp 为 encfile.cpp:

从命令行接受 3 个字符串类型的 参数: 参数 1, 参数 2, 参数 3。

- 参数 1=enc 表示加密、参数 1=dec 表示解密;
- 参数 2 为待加密、解密的文件名;
- 参数 3 为密码。

以文件 cryptoDemo.cpp 为测试文件, 以你的学号 PB18111707 为密码, 验证程序 encfile.cpp 的正确性。

实验步骤

编写 encfile.cpp

1. 修改 testAes 函数的参数, 使符合实验要求

```
1 // void testAes(char flag[], char file_name[], char passwd[], int  
  pwdLen);  
2 testAes(argv[1], argv[2], argv[3], strlen(argv[3]));
```

其中, `char *argv[]` 为从命令行获取的参数数组, `argv[0]` 代表命令本身, `argv[1]` 代表第一个参数, 以此类推, `argv[i]` 代表第 `i` 个参数

2. 实现加密功能, 将加密后的文档存为 `encode.dat`

- a. 二进制打开待加密文档 `cryptoDemo.cpp`; 初始化字符串数组 `enstring` 为 0, 用于暂时存储加密后的所有字符; 二进制打开一个新的文档 `encode.dat` 等待写入加密后的内容。
- b. 以 16 字节为单位, 用 `fread` 语句读取字符串到 `buf`, 当读取的字符串不满 16 字节时, 在 `buf` 的剩余部分用 0 做填充。
- c. 对 `buf` 中的字符加密, 加密后的字符缓存在 `buf2` 中, 显然, `buf2` 中有效字符的大小永远是 16 个。
- d. 将 `buf2` 中的字符赋值到 `enstring` 数组中的对应位置。
- e. 将 `enstring` 以二进制方式写入 `encode.dat`

```
1 // 实现加密功能  
2 if (flag[0] == 'e') {  
3     // 加密输入的字节串
```

```

4     AES_set_encrypt_key(aes_keybuf, 256, &aeskey);
5
6     printf("encoding -----\\n");
7
8     // 加密后的文件存放到 encode.dat 中
9     FILE *encode = fopen("encode.dat", "wb");
10    if (!encode) {
11        printf("the encode.dat can not open!\\n");
12        exit(-1);
13    }
14
15    int enlen = 0; // 已加密的字节数
16    int lenth = 0; // 记录读入的字节个数
17    memset(buf, 0, 16);
18    while (lenth = fread(buf, sizeof(char), 16, fp)) {
19        // 以 16 字节为单位读取字符串, 存放在 buf 中
20        if (lenth % 16) {
21            // lenth < 16 即, 最后不满 16 字节的字符串
22            for (int i = lenth; i < 16; i++) {
23                buf[i] = 0;
24            }
25            printf("the original key is %s,the lenth is %d\\n", buf,
lenth);
26        }
27
28        // 加密后的字符缓存在 buf2 中
29        AES_encrypt(buf, buf2, &aeskey);
30
31        if (lenth % 16) {
32            printf("the new key is %s", buf2);
33        }
34
35        for (int j = 0; j < 16; j++, enlen++) {
36            // you need to save 16 bytes encode characters
37            enString[enlen] = buf2[j];
38        }
39        memset(buf, 0, 16);
40        memset(buf2, 0, 16);
41    }
42    enString[enlen] = 0;
43    fwrite(enString, sizeof(char), enlen, encode);
44 }

```

3. 实现解密功能, 将解密后的文档存为 decode.cpp

打开文件、新建文件等步骤略过。

- 以 16 字节为单位, 用 `fread` 二进制读取加密文档中的字符串到 `buf` 数组中。
- 对 `buf` 中的字符解密, 解密后的字符缓存在 `buf2` 中, 显然, 当 `buf2` 中出现 0 时, 该字符是对源文档的填充, 不应该写入解密后的文件中。

```

1     // 密文串的解密
2     else if (flag[0] == 'd') {
3         printf("decoding-----\\n");
4         // 解密后的文件放在 decode.dat 中
5         FILE *decode = fopen("decode.cpp", "wb");

```

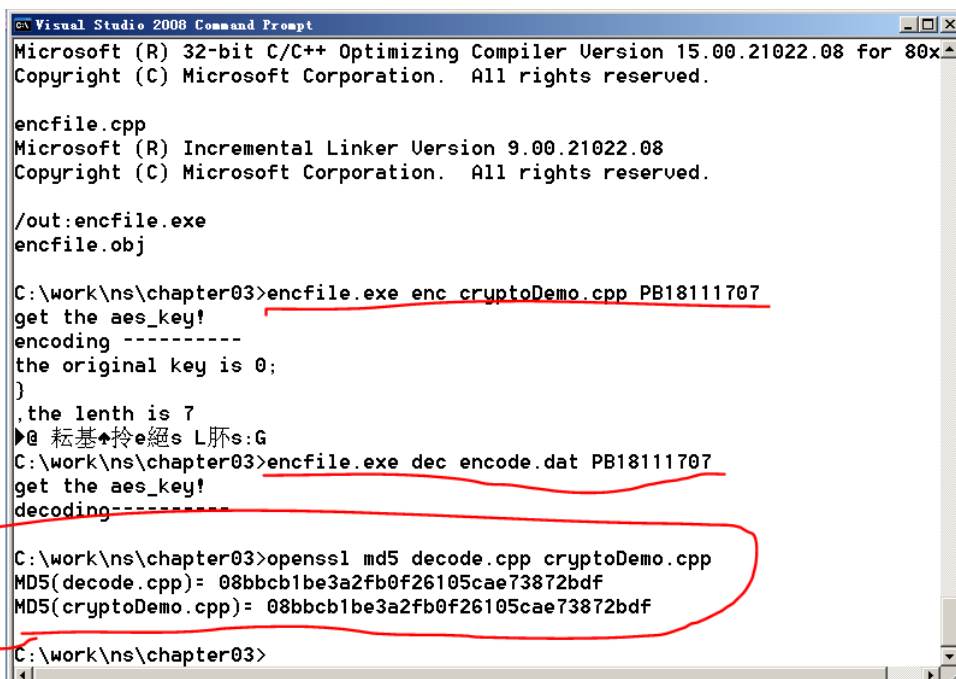
```

6      if (!decode) {
7          printf("the decode.cpp can not open!\n");
8          exit(-1);
9      }
10
11     AES_set_decrypt_key(aes_keybuf, 256, &aeskey);
12
13     int delen = 0; // 解密出来的字符串
14     int lenth = 0; // 实际读取出来的字符个数
15     int res = 0; // record the last res
16
17     memset(buf, 0, 16);
18     while (lenth = fread(buf, sizeof(char), 16, fp)) {
19         // 解密后的字符串存储缓存在 buf2 中
20         AES_decrypt(buf, buf2, &aeskey);
21
22         for (int j = 0; j < 16; j++, delen++) {
23             if (buf2[j] == 0) {
24                 res++;
25             }
26             deString[delen] = buf2[j];
27         }
28
29         memset(buf, 0, 16);
30         memset(buf2, 0, 16);
31     }
32     deString[delen] = 0;
33     fwrite(deString, sizeof(char), delen - res, decode);
34 }

```

验证程序正确性

用 openssl 的 md5 命令验证程序的正确性，即验证解密后的文件 decode.cpp 和原始文件 cryptoDemo.cpp 是否相同



```

C:\work\ns\chapter03>encfile.exe enc cryptoDemo.cpp PB18111707
get the aes_key!
encoding -----
the original key is 0;
]
the lenth is 7
> @ 耘基+拎e絕s L胚s:G
C:\work\ns\chapter03>encfile.exe dec encode.dat PB18111707
get the aes_key!
decoding-----
C:\work\ns\chapter03>openssl md5 decode.cpp cryptoDemo.cpp
MD5(decode.cpp)= 08bbcb1be3a2fb0f26105cae73872bdf
MD5(cryptoDemo.cpp)= 08bbcb1be3a2fb0f26105cae73872bdf
C:\work\ns\chapter03>

```

