山东大学 软件 学院

信息安全导论 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201500301112 | 姓名： 邵长旭 | | 班级：2015级4班 |
| 实验题目：文件加密系统 | | | |
| 实验学时：2 | | 实验日期： 2017-11-20 | |
| 实验目的：  通过编写文件加密系统，了解对称加密算法的使用，了解如何将libtom等安全工具包应用于自己的程序。 | | | |
| 硬件环境：  Intel core i7-5500U | | | |
| 软件环境：  Window10 | | | |
| 实验步骤与内容：  本实验中采用libtom工具包，libtom工具包相比OpenSSL工具包更易于上手学习。   1. 配置libtom：   先打开libtomcrypt\_VS2008.sln    编译文件，但是发现需要tommath数学包，下载后同样编译tommath数学包，之后将lib文件和各种头文件考到自己的项目中，做一下静态库的链接，就可以开始编写自己的代码了。   1. 编写代码：   在代码初始，声明一些变量，用来接受用户的输入，本程序中提供了两种aes、des两种加密方法，以及cbc、ofb、cfb、ctr四种模式，下面的程序中以aes算法的cbc模式来讲解：    接下来对文件名进行一些操作，加密文件在源文件名后加上.crypt后缀名，对于解密文件，将.crypt后缀名去掉，还原出原始文件。      在正常加解密开始之前，先注册加密方法      我的加解密策略是这样的：  生成函数  **随机生成的密钥(用于本次加密)**  **双方约定好的密钥**  **明文**  加密  加密  **密文**  解密  **双方约定好的密钥**  解密  **本次使用的密钥**  **明文**  加密双方事先约定好一个密钥，然后基于这个密钥随机生成每次加密过程中要使用的密钥，并通过事先约定好的密钥对这个随机密钥加密，保证安全性，然后利用这个随机密钥对明文进行加密，形成传输过程中的密文，还需要注意的是，最好利用以下代码生成一次密钥，这样可以解决弱密钥的问题，mykey就是我们上文中生成的key，调用它的方法再在内部生成一遍密钥可以进一步加强安全性。    我在这里采用的随机生成函数只是在原密钥中随机找一个位置，将之前的字符与之后的交换，虽然这样做的安全性不是特别高，但是我们这里只是用来展示思路，当然，我们也可以采用hash等方法生成更加随机的密钥：    接下来，开始我们的加解密过程：  加密过程：  思路是这样的，先利用生成随机密钥，打开加密文件和明文文件，利用XXX\_start方法绑定密钥，XXX是指工作模式，然后利用cbc\_encrypt加密随机密钥，切换密钥，再去加密明文，明文文件有可能太大而无法一次读入内存，所以我们一块块的读入文件，加密文件最后写入密文文件。最后利用cbc\_done和unregister\_cipher完成收尾工作；  解密过程思路与加密大体一致，就不再讲述、贴代码了。             1. 测试程序：     如上图，我们利用aes算法和cbc模式加密b.que文件，显然，操作系统中是没有.que这种类型文件的，不过，这并不影响我们的加解密过程，实际是我们的b.que是这样的：    加密后：      解密后：    可以看到整个过程正确。  下面我们再测试一个相对较大的文件，一张png图片（c.png）：      加密后：      解密后：    文件能正常打开：    和原始文件一样,证明实现成功. | | | |
| 结论分析与体会：  本次实验采用的是对称加密算法，对称加密指加密和解密使用相同密钥的加密算法。有时又叫传统密码算法，就是加密密钥能够从解密密钥中推算出来，同时解密密钥也可以从加密密钥中推算出来。    在加密算法中，要求密钥一定是随机的，不能约定不变，这样即使攻击者偶然得到了一次加密的密钥，也无法用此密钥继续破解其他的信息。而且本次实验是对文件进行加密，对文件的类型不应该人为的加以限制，考虑问题需要更加全面，比如文件过大无法一次调入内存，所以我们应该分次读入内存。  实验的目的是要让我们加密的文件尽可能的安全，所以我们采用了别人写好的安全工具包，著名的是OpenSSL,但是这个工具包过大，不利用于初学者使用学习，我们采用了相对简单的libtom工具包来完成加解密工作，同样，在我们设计的加密策略的基础上，应当再次调用现成的生成密钥的方法来规避弱密钥等安全问题。 | | | |

代码：

#include"tomcrypt.h"

#include<iostream>

#include<tchar.h>

#include <functional>

using namespace std;

/\*

注册方法，注册加密的方法，如des\aes等

\*/

int regis(int Method) {

if (Method == 1) {

//aes

if (register\_cipher(&aes\_desc) == -1) {

printf("failed to register\_cipher");

return -1;

}

}

else if (Method == 2) {

//des

if (register\_cipher(&des\_desc) == -1) {

printf("failed to register\_cipher");

return -1;

}

}

//成功

return 1;

}

/\*

形成密文和解密文的文件名

\*/

char\* deal\_fileName(char \*deal\_name,int type,char \*pn) {

int last = 0;

for (; pn[last] != '\0'; last++);

int dotlocation = last;

for (; pn[dotlocation] != '.'; dotlocation--);

if (type == 1) {

int i = 0;

for (; i < last; i++) {

deal\_name[i] = pn[i];

}

deal\_name[i++] = '.';

deal\_name[i++] = 'c';

deal\_name[i++] = 'r';

deal\_name[i++] = 'y';

deal\_name[i++] = 'p';

deal\_name[i++] = 't';

deal\_name[i++] = '\0';

}

else if (type == 2) {

int i = 0;

for (; i < dotlocation; i++) {

deal\_name[i] = pn[i];

}

}

return deal\_name;

}

void generateRandomKey(unsigned char \*mykey,unsigned char \*key) {

srand((unsigned)time(NULL));

int b = -1;

for (b = 0; key[b] != '\0';b++);

int index = (rand() % b);

for (int i = 0; i < b-index; i++) {

mykey[i] = key[index+i];

}

for (int i = 0; i < index; i++) {

mykey[i+ b - index] = key[i];

}

mykey[b] = '\0';

}

int main() {

cout << "Welcome to use the program\n";

int type = -1;//加密1 解密2

int method = 0;//aes des

int mode = 0;//cbc,ofb,cfb,ctr

//加解密文件名

char crypt\_name[100] = "", plain\_name[100] = "", decrypt\_name[100] = "";

char \*Method[] = {"aes","des"};

char \*Mode[] = {"CBC","OFB","CFB","CTR"};

int BlockSize[2] = {16,8};

int keySize[2] = {16,7};

int error = 0;//返回值

int cipher = 0;//加解密索引

unsigned char key[16] = "scx199702scx199";

const unsigned char IV[16] = "oTvdDgf3mSab7Lv";//初始向量

FILE \*plaintText = 0, \*cryptText = 0, \*decryptText = 0;

symmetric\_CBC cbc = { 0 };

symmetric\_OFB ofb = { 0 };

symmetric\_CFB cfb = { 0 };

symmetric\_CTR ctr = { 0 };

size\_t len = 0;

cout << "would you like to cyprt or decrypt?1 to crypt,2 to decrypt\n";

cin >> type;

int MethodLen = sizeof(Method) / sizeof(Method[0]);

int ModeLen = sizeof(Mode) / sizeof(Mode[0]);

for (int i = 0; i < MethodLen; i++) {

cout << i + 1 << "\t" << Method[i] << "\n";

}

cout << "please choose one method:\n";

cin >> method;

unsigned char ct[16] = "";

unsigned char pt[16] = "";

for (int i = 0; i < ModeLen; i++) {

cout << i + 1 << "\t" <<Mode[i] << "\n";

}

cout << "please choose one mode:\n";

cin >> mode;

cout << "please input the location of the file:\n";

cin >> plain\_name;

if (type == 1) {

deal\_fileName(&crypt\_name[0],type,plain\_name);

}

else if (type == 2) {

deal\_fileName(&decrypt\_name[0], type,plain\_name);

}

//cout << "please input the key:(the key length use 8)\n";

//cin >> key;//双方事先约定的口令，如scx199702

/\*cout << "please input the IV:\n";

cin >> IV;\*/

//deal the key

unsigned char mykey[16];

//注册加密方法

if (regis(method) == -1) {

return -1;

}

cipher = find\_cipher(Method[method-1]);

if (cipher == -1) {

printf("find\_cipher error");

goto Del;

}

switch (mode)

{

case 1:

error = cbc\_start(cipher,IV,mykey,keySize[method-1],0,&cbc);

break;

case 2:

error = ofb\_start(cipher, IV, mykey, keySize[method - 1], 0, &ofb);

break;

case 3:

error = cfb\_start(cipher, IV, mykey, keySize[method - 1], 0, &cfb);

break;

case 4:

error = ctr\_start(cipher, IV, mykey, keySize[method - 1], 0,0, &ctr);

break;

default:

break;

}

if (error!=CRYPT\_OK) {

printf("failed to start", error\_to\_string(error));

goto Del;

}

if (type == 1) {

generateRandomKey(&mykey[0], key);

//加密

fopen\_s(&plaintText, plain\_name, "rb");

fopen\_s(&cryptText, crypt\_name, "wb");

////加密密钥

switch (mode)

{

case 1:

error = cbc\_start(cipher, IV, key, keySize[method - 1], 0, &cbc);

break;

case 2:

error = ofb\_start(cipher, IV, key, keySize[method - 1], 0, &ofb);

break;

case 3:

error = cfb\_start(cipher, IV, key, keySize[method - 1], 0, &cfb);

break;

case 4:

error = ctr\_start(cipher, IV, key, keySize[method - 1], 0, 0, &ctr);

break;

default:

break;

}

if (error != CRYPT\_OK) {

printf("failed to start", error\_to\_string(error));

goto Del;

}

switch (mode)

{

case 1:

error = cbc\_encrypt(mykey, ct, 16, &cbc);

break;

case 2:

error = ofb\_encrypt(mykey, ct, 16, &ofb);

break;

case 3:

error = cfb\_encrypt(mykey, ct, 16, &cfb);

break;

case 4:

error = ctr\_encrypt(mykey, ct, 16, &ctr);

break;

default:

break;

}

if (error != CRYPT\_OK) {

printf("failed to crypt key", error\_to\_string(error));

}

fwrite(ct, sizeof(ct[0]), 16, cryptText);

//加密明文

switch (mode)

{

case 1:

cbc\_done(&cbc);

error = cbc\_start(cipher, IV, mykey, keySize[method - 1], 0, &cbc);

break;

case 2:

error = ofb\_start(cipher, IV, mykey, keySize[method - 1], 0, &ofb);

break;

case 3:

error = cfb\_start(cipher, IV, mykey, keySize[method - 1], 0, &cfb);

break;

case 4:

error = ctr\_start(cipher, IV, mykey, keySize[method - 1], 0, 0, &ctr);

break;

default:

break;

}

if (error != CRYPT\_OK) {

printf("failed to start", error\_to\_string(error));

goto Del;

}

fseek(plaintText, 0L, SEEK\_END);

long end = ftell(plaintText);

fseek(plaintText, 0L, SEEK\_SET);

long start = ftell(plaintText);

while (start!=end) {

memset(pt,0,BlockSize[method-1]);

memset(ct, 0, BlockSize[method - 1]);

len = fread(pt, sizeof(pt[0]), BlockSize[method-1], plaintText);

start = ftell(plaintText);

if (len < 1) {

break;

}

switch (mode)

{

case 1:

error = cbc\_encrypt(pt,ct,16,&cbc);

break;

case 2:

error = ofb\_encrypt(pt, ct, 16, &ofb);

break;

case 3:

error = cfb\_encrypt(pt, ct, 16, &cfb);

break;

case 4:

error = ctr\_encrypt(pt, ct, 16, &ctr);

break;

default:

break;

}

fwrite(ct,sizeof(ct[0]),16,cryptText);

}

printf("Success Crypt");

fclose(plaintText);

fclose(cryptText);

goto Del;

}

else if (type == 2) {

//解密

fopen\_s(&decryptText, decrypt\_name, "wb");

fopen\_s(&cryptText, plain\_name, "rb");

////解密密钥

switch (mode)

{

case 1:

error = cbc\_start(cipher, IV, key, keySize[method - 1], 0, &cbc);

break;

case 2:

error = ofb\_start(cipher, IV, key, keySize[method - 1], 0, &ofb);

break;

case 3:

error = cfb\_start(cipher, IV, key, keySize[method - 1], 0, &cfb);

break;

case 4:

error = ctr\_start(cipher, IV, key, keySize[method - 1], 0, 0, &ctr);

break;

default:

break;

}

bool first = true;

fseek(cryptText, 0L, SEEK\_END);

long end = ftell(cryptText);

fseek(cryptText, 0L, SEEK\_SET);

long start = ftell(cryptText);

while (!feof(cryptText)) {

memset(pt, 0, sizeof(pt));

memset(ct, 0, sizeof(ct));

len = fread(ct, sizeof(ct[0]), BlockSize[method - 1], cryptText);

//start = ftell(cryptText);

if (len < 1) {

break;

}

switch (mode)

{

case 1:

error = cbc\_decrypt(ct, pt, 16, &cbc);

break;

case 2:

error = ofb\_decrypt(ct, pt, 16, &ofb);

break;

case 3:

error = cfb\_decrypt(ct, pt, 16, &cfb);

break;

case 4:

error = ctr\_decrypt(ct, pt, 16, &ctr);

break;

default:

break;

}

if (error != CRYPT\_OK) {

printf("failed to decrypt", error\_to\_string(error));

break;

}

//cout << pt<<"\n";

//fwrite(pt, sizeof(pt[0]), 16, decryptText);

if (first) {

first = false;

switch (mode)

{

case 1:

error = cbc\_start(cipher, IV, pt, keySize[method - 1], 0, &cbc);

break;

case 2:

error = ofb\_start(cipher, IV, pt, keySize[method - 1], 0, &ofb);

break;

case 3:

error = cfb\_start(cipher, IV, pt, keySize[method - 1], 0, &cfb);

break;

case 4:

error = ctr\_start(cipher, IV, pt, keySize[method - 1], 0, 0, &ctr);

break;

default:

break;

}

}

else {

fwrite(pt, sizeof(pt[0]), 16, decryptText);

}

}

printf("Success decrypt");

fclose(decryptText);

fclose(cryptText);

goto Del;

}

Del:

switch (mode)

{

case 1:

cbc\_done(&cbc);

break;

case 2:

ofb\_done(&ofb);

break;

case 3:

cfb\_done(&cfb);

break;

case 4:

ctr\_done(&ctr);

break;

default:

break;

}

switch (method)

{

case 1:

error = unregister\_cipher(&aes\_desc);

break;

case 2:

error = unregister\_cipher(&des\_desc);

break;

default:

break;

}

if (error != CRYPT\_OK) {

printf(("unregister error"));

return -1;

}

return 0;

}