**实 验 报 告 五**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **课 程** | 物联网信息感知技术 | **实验项目** | RFID写卡实验 | |
| **专业班级** | 物联网工程2015级 | | 实验日期 | 2017-05-27 星期六 |
| **报告撰写** | **石华** | **学 号** | **201531060570** |  |
| **同组人1** | **孔维坤** | **学 号** | **201531060562** |  |
| **同组人2** | **冉驰昊** | **学 号** | **201531060557** |  |

【**实验目的**】

1. 熟悉RFID写卡的基本原理。
2. 了解RFID中的数据加密的基本原理。
3. 掌握数据加密与界面。

**【实验步骤】**

1阅读教师提供的【**pdf格式参考论文**】文件夹下的论文，了解RFID系统中，加密的必要性，加密的基本原理。

2使用任何一种编程语言，实现如下功能：假定要加密的信息是**你的学号**(例如201531026666)，以及你往一卡通上**充值的金额**(1.00-500.00)之间的任何数(例如1.01，1.02，....499.98，499.99，500.00)，对这两项数据，选取你认为合理的数据类型格式存储(即定义变量或数组，直接为其赋值)。用一种**你已经掌握的简单的加密算法**，进行加密。在加密之后，尝试对**内存中的数据**进行解密，查看解密后是否与原始的数据完全相同。

代码：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#define KEY 131311

using namespace std;

int encrypt(int original, int key) {

return original^key;

}

int decrypt(int encryption, int key) {

return encryption^key;

}

int main() {

setlocale(LC\_ALL, "");

int money[2];

int money\_en[2];

int result[2];

int StudentID[2];

cout << "请输入学号:";

scanf("%6d%6d", &StudentID[0], &StudentID[1]);

cout << "请输入余额:";

scanf("%d.%2d", &money[0], &money[1]);

cout << "要加密的数据为:" << StudentID[0] << StudentID[1] << " " << money[0] << "." << money[1] << endl;

result[0] = encrypt(StudentID[0], KEY);

result[1] = encrypt(StudentID[1], KEY);

money\_en[0] = encrypt(money[0], KEY);

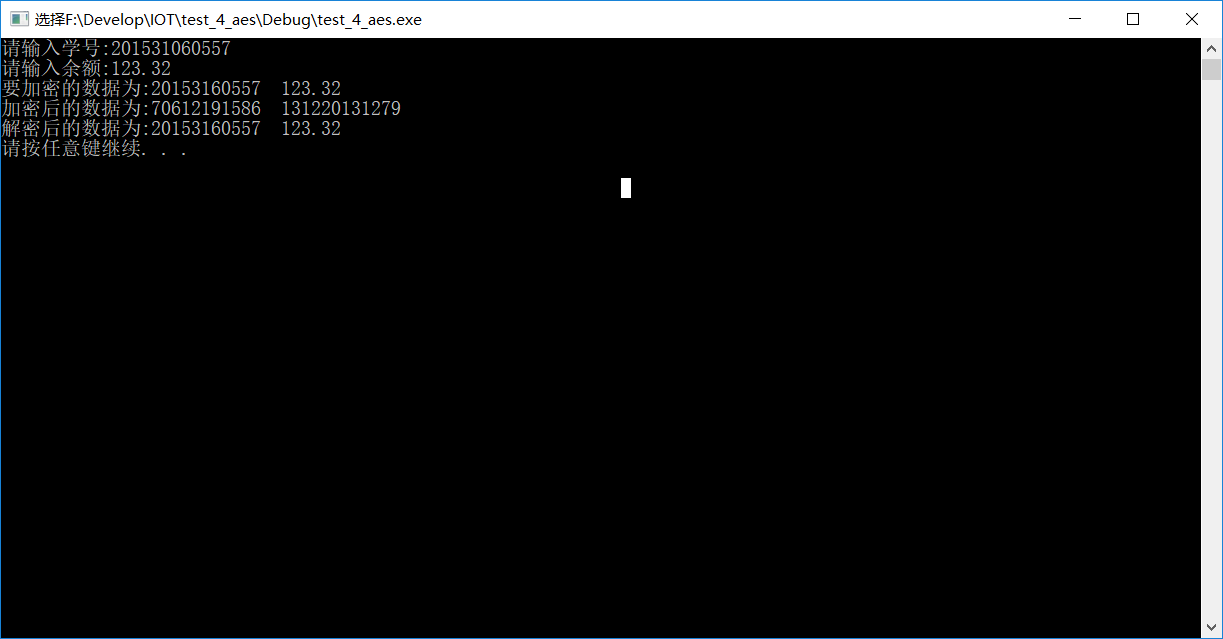
money\_en[1] = encrypt(money[1], KEY);

cout << "加密后的数据为:" << result[0] << result[1] << " " << money\_en[0] << money\_en[1] << endl;

cout << "解密后的数据为:" << decrypt(result[0], KEY) << decrypt(result[1], KEY) << " " << decrypt(money\_en[0], KEY) << "." << decrypt(money\_en[1], KEY) << endl;

system("pause");

}



3在已经完成第二步的基础上，使用**AES加密算法**，重新实现上述的步骤。注意，RFID Gen2标签在芯片中有96字节的存储空间，要确定你的加密后占用的字节数不超过96字节。

代码：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <sstream>

#include <string>

#include <stdio.h>

#include <assert.h>

#include <openssl\evp.h>

#include <openssl\objects.h>

using namespace std;

string aesEncrypt(string &data, string &key)

{

unsigned char\* out = new unsigned char[data.length() + 64];

int len, outl;

len = 0;

EVP\_CIPHER\_CTX ctx;

EVP\_CIPHER\_CTX\_init(&ctx);//初始化ctx

EVP\_EncryptInit\_ex(&ctx, EVP\_aes\_128\_ecb(), NULL, (const unsigned char \*)key.data(), NULL);//初始化加密方式（\*ctx，加密方式，引擎，key，额外选项）

EVP\_EncryptUpdate(&ctx, out, &outl, (unsigned char\*)data.data(), data.length());//加密 (\*ctx，加密后数据，加密长度，要加密的内容，要加密的长度)

len += outl;

EVP\_EncryptFinal\_ex(&ctx, out + len, &outl); //aes加密为64位作为一块，末尾不足的填补后再加密(\*ctx，加密后数据，最后加密的长度)

len += outl;

string result((char\*)out, len);//用的string存数据，用char[]不需要这步

return result;

}

string desDecrypt(string &data, string &key)

{

unsigned char\* out = new unsigned char[data.length() + 64];

int len, outl;

int ret;

len = 0;

EVP\_CIPHER\_CTX ctx;

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_DecryptInit\_ex(&ctx, EVP\_aes\_128\_ecb(), NULL, (const unsigned char \*)key.data(), NULL);//类比加密，下同

EVP\_DecryptUpdate(&ctx, out, &outl, (unsigned char \*)data.data(), data.length());

len += outl;

EVP\_DecryptFinal\_ex(&ctx, out + len, &outl);

len += outl;

string result((char\*)out, len);

return result;

}

int main() {

setlocale(LC\_ALL, "");

string key = "helloaes";

string studentID;

stringstream ss;

double money;

string Money;

string out;

string result[2];

cout << "请输入学号:";

cin >> studentID;

cin.clear();

cin.sync();

cout << "请输入余额:";

scanf("%lf", &money);

fflush(stdin);

ss << fixed << setprecision(2) << money;

Money = ss.str();

cout << "要加密的数据为:" << studentID << " " << Money << endl;

result[0] = aesEncrypt(studentID, key);

result[1] = aesEncrypt(Money, key);

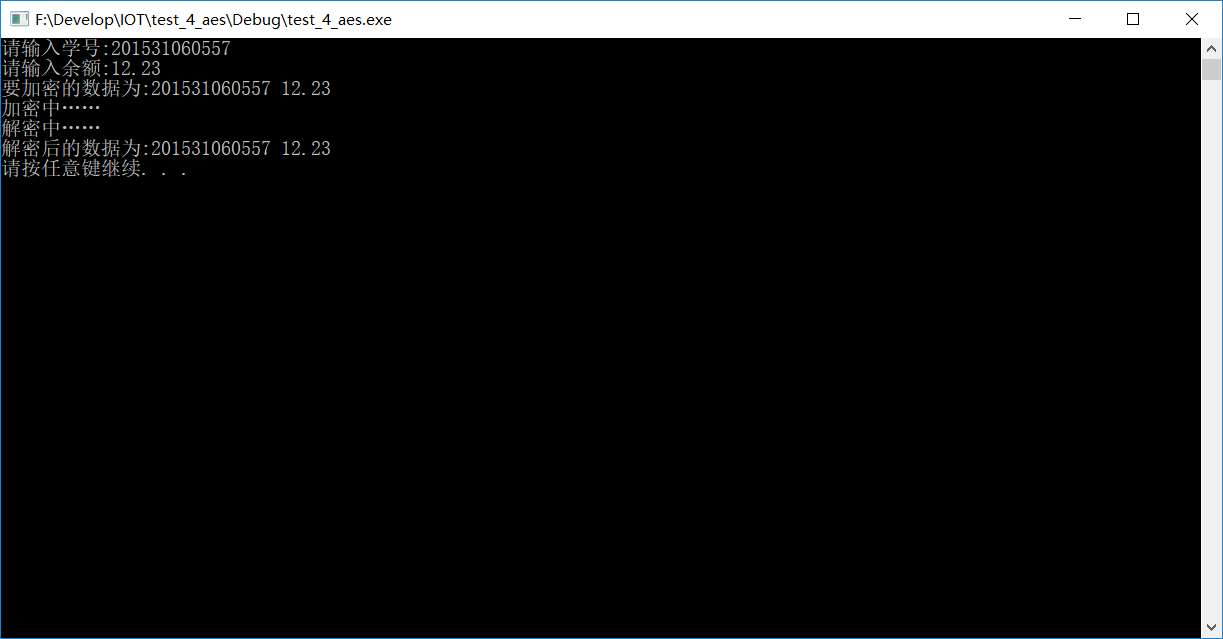
cout << "加密中……" << endl;

cout << "解密中……" << endl;

cout << "解密后的数据为:" << desDecrypt(result[0], key) << " " << desDecrypt(result[1], key) << endl;

system("pause");

}



4在实现第三步的前提下，将原始要进行加密的数据，存储到一个txt文件中。使用AES加密算法进行加密后的数据，存储到一个bin文件中。之后尝试对bin文件进行解密，比较信息是否和原来一致。

(例如，对于input.txt，加密得到了data.bin，再对data.bin解密，得到的文件是output.txt，比较input.txt和output.txt是否完全一致。)

代码：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <sstream>

#include <fstream>

#include <string>

#include <stdio.h>

#include <assert.h>

#include <openssl\evp.h>

#include <openssl\objects.h>

using namespace std;

string aesEncrypt(string &data, string &key)

{

unsigned char\* out = new unsigned char[data.length() + 64];

int len, outl;

len = 0;

EVP\_CIPHER\_CTX ctx;

EVP\_CIPHER\_CTX\_init(&ctx);//初始化ctx

EVP\_EncryptInit\_ex(&ctx, EVP\_aes\_128\_ecb(), NULL, (const unsigned char \*)key.data(), NULL);

EVP\_EncryptUpdate(&ctx, out, &outl, (unsigned char\*)data.data(), data.length());

len += outl;

EVP\_EncryptFinal\_ex(&ctx, out + len, &outl); //aes加密为64位作为一块，末尾不足的填补后再加密

len += outl;

string result((char\*)out, len);//用的string存数据

return result;

}

string desDecrypt(string &data, string &key)

{

unsigned char\* out = new unsigned char[data.length() + 64];

int len, outl;

int ret;

len = 0;

EVP\_CIPHER\_CTX ctx;

EVP\_CIPHER\_CTX\_init(&ctx);

EVP\_DecryptInit\_ex(&ctx, EVP\_aes\_128\_ecb(), NULL, (const unsigned char \*)key.data(), NULL);//类比加密，下同

EVP\_DecryptUpdate(&ctx, out, &outl, (unsigned char \*)data.data(), data.length());

len += outl;

EVP\_DecryptFinal\_ex(&ctx, out + len, &outl);

len += outl;

string result((char\*)out, len);

return result;

}

void write(string fName, string wData) { //写文件函数

ofstream out(fName); //fstream子类

if (out.is\_open()) {

out << wData;

out.close();

}

}

string readF(string fName) { //读文件函数

char buffer[256]; //fstream子类

string result = "";

ifstream in(fName);

if (!in.is\_open())

{

cout << "打开文件失败！";system("pause"); exit(1);

}

while (!in.eof())

{

in.getline(buffer, 100); //getline函数第三个参数为结束符，不填的时候为\n

result.append(buffer);

if (in.eof())break;

result.append("\n"); //为了不舍弃文本中的\n，当一行结束后手动加一个\n字符

}

in.close();

return result; //返回文本内容

}

int main() {

setlocale(LC\_ALL, "");

string key = "hellodes";

cout << readF("F:\\input.txt") << endl; // 更改为自己的目录

write("F:\\data.bin", aesEncrypt(readF("F:\\input.txt"), key));

write("F:\\output.txt", desDecrypt(readF("F:\\data.bin"), key));

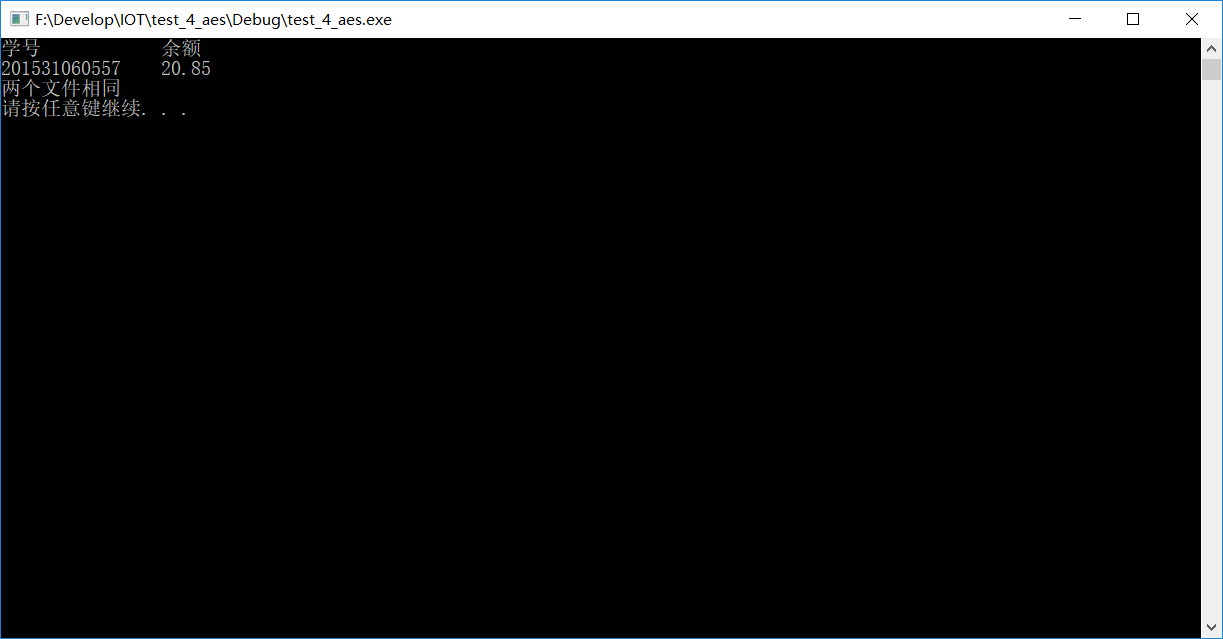
string f1 = readF("F:\\input.txt");

string f2 = readF("F:\\output.txt");

if (!f1.compare(f2))cout << "两个文件相同" << endl;

system("pause");

}



**【实验总结】**

通过本次实验，学习了数据加密的过程和aes加密的原理和用法。了解了简单加密和高级加密分别需要满足的需求。能够使用简单加密和aes进行简单的加密操作