****

FUZHOU TECHNOLOGY AND BUSINESS UNIVERSITY



**实验报告**

**课程名称： 网络信息安全技术**

**系 别： 工学院 年 级： 2018级**

**专 业： 软件工程**

**班 级： 1班 学 号： 1850304048**

**姓　　名： 汤郭**

**成 绩：**

**任课教师： 林晖**

**2021年 5 月 24 日**

目录

[一、3DES密码算法 1](#_Toc73292836)

[**1.实验目的** 1](#_Toc73292837)

[**2.实验原理** 1](#_Toc73292838)

[**3.实验环境** 4](#_Toc73292839)

[**4.实验步骤** 4](#_Toc73292840)

[㈠查看3DES算法运行结果并学习核心算法 4](#_Toc73292841)

[二、AES密码算法 6](#_Toc73292842)

[**1.实验目的** 6](#_Toc73292843)

[**2.实验原理** 6](#_Toc73292844)

[**3.实验环境** 10](#_Toc73292845)

[**4.实验步骤** 10](#_Toc73292846)

[㈠字符串加解密 10](#_Toc73292847)

[㈡文件文件加解密 13](#_Toc73292848)

[㈢源码阅读 16](#_Toc73292849)

[三、RSA密码算法 18](#_Toc73292850)

[**1.实验目的** 18](#_Toc73292851)

[**2.实验原理** 18](#_Toc73292852)

[**3.实验环境** 22](#_Toc73292853)

[**4.实验步骤** 23](#_Toc73292854)

[㈠查看RSA算法运行结果并学习核心算法 23](#_Toc73292855)

**实验一 网络数据报分析**

# 一、3DES密码算法

**1.实验目的**

1）了解3DES的算法原理

2）学习3DES算法的编程实现

**2.实验原理**

**(1)算法原理**

(a)DES算法的密钥长度已经被证明不能满足当前安全的要求，但为了充分利用现有的DES软件和硬件资源，人们开始提出针对DES的各种改进方案，最简单的方案是使用多重DES，使用多个不同的DES密钥利用DES加密算法对明文进行多次加密。这样可以增加密钥量，从而大大提高抵抗对密钥的穷举搜索攻击的能力。而最常用的多主要是三重DES算法。

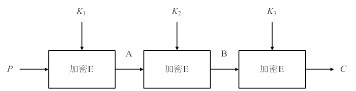
(b)三重DES有4种模式。如下图一所示

Ø  DES-EEE3 模式：该模式中共使用3个不同密钥，顺序使用3次DES加密算法。

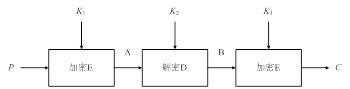
Ø  DES-EDE3 模式：该模式中共使用3个不同密钥，依次用加密—解密—加密。

Ø  DES-EEE2 模式：该模式中共使用2个不同密钥，顺序使用3次DES加密算法，其中第一次和第三次加密使用的密钥相同。

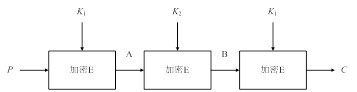
Ø  DES-EDE2 模式：该模式中共使用2个不同密钥，依次用加密—解密—加密，其中加密算法使用的密钥相同。



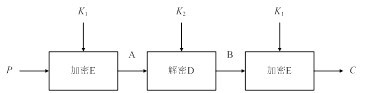
（a） DES-EEE3模式



（b） DES-EDE3模式



（c） DES-EEE2模式



（d） DES-EDE2模式

图一

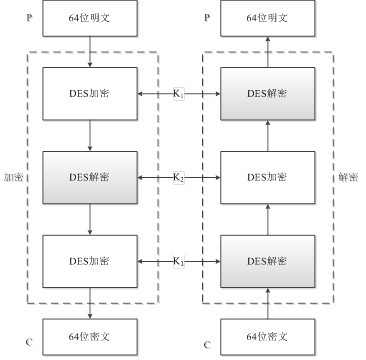
Ø   前两种模式使用3个不同的密钥，每个密钥长度为56位，因此三重DES总的密钥长度达到168位。后2种模式使用2个不同的密钥，总的密钥长度为112位。以前两种模式为例，设http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_38365.gif，http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_49645.gif，http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_19857.gif是三个长度为56位的密钥。给定明文http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_35chapter     = 515.gif，则密文为

http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_71816.gif

  给定密文http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_62224.gif，则明文为

http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_10044.gif

Ø   在1999年10月发布的DES标准报告中推荐使用的三重DES是http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_48340.gif的情形，既两个密钥的三重DES算法。这是一种比较受欢迎的DES的替代方案。银行业已大量采用这种算法。http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_96567.gif，http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_60259.gif，http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_55995.gif互不相同的三DES在实际应用中也经常被采用。下面我们给出http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_15542.gif模式的三重DES加密和解密内部流程。



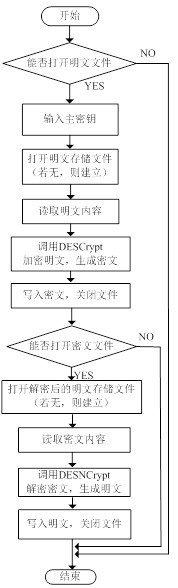
图二

**(2)算法参数**

在三重DES算法中，其参数与DES是保持一致的，密钥总长度为64位。其中有效长度为56位，并附加上8位的奇偶校验位。分组长度为64位，初始置换IP，逆初始置换http://192.168.1.205/Public/editor/attached/image/20130724/20130724231056_81534.gif，E盒，S盒均在DES算法中予以说明，这里就不再赘述了。

**(3)算法流程**

根据在上文中叙述的有关算法的运行原理及加解密流程，我们给出了其用C语言实现的源代码，相关的程序流程图如下。由于三重DES算法的代码以单重DES算法为基础，除了中间需要一步逆向的解密过程外，其他均是DES的单重加密。所以我们这里仅给出单重DES的加密流程图。如图三所示



图三

**3.实验环境**

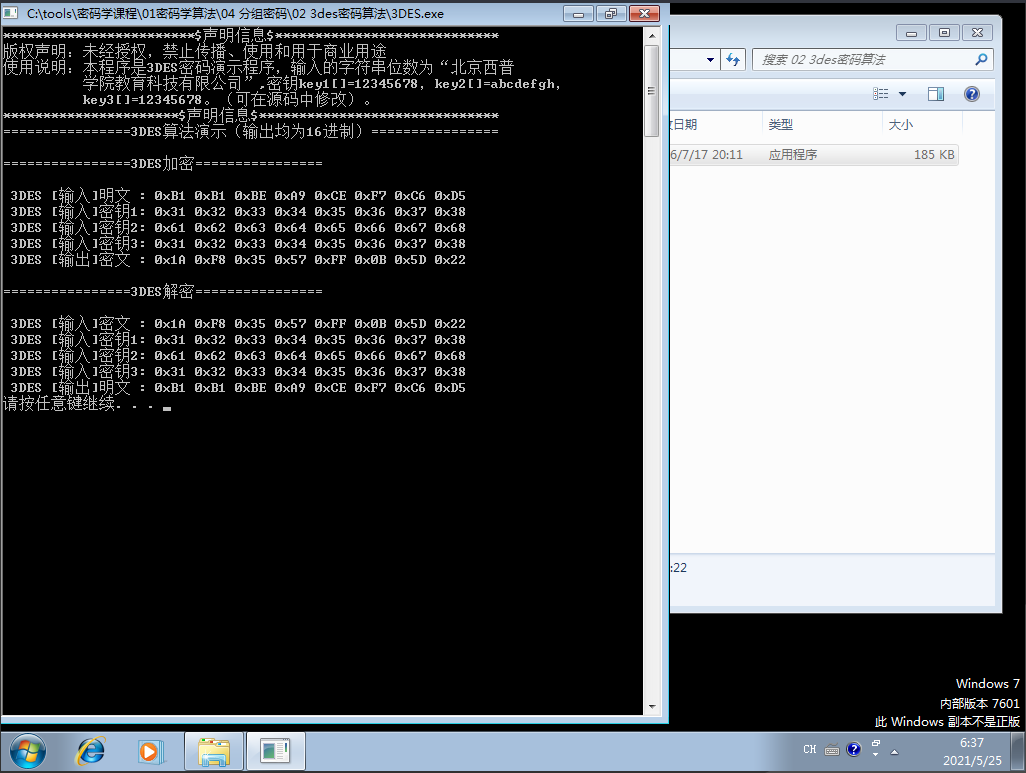
Windows 7

工具：C:\tools\密码学课程\01密码学算法\04 分组密码\02 3des密码算法。

**4.实验步骤**

㈠查看3DES算法运行结果并学习核心算法

1.1运行程序，即可得到3DES加密算法的结果。



1.2 3DES核心密码算法如下。

#include "stdio.h"

#include "des.h"

#include "pub.h"

void main() {

unsigned char key1\[\] = "12345678";

unsigned char key2\[\] = "abcdefgh";

unsigned char key3\[\] = "12345678";? //如果只需要两组密钥，则本组密钥可以和密钥1一样。

unsigned char en\\_data\[\] = "Cryptosystem lab";

unsigned char en\\_out\\_put\[8\];

printf("\\r\\n++++++++3DES加密示例++++++++");

print\\_hex("3DES \[输入\]明文 ", en\\_data, 8);

print\\_hex("3DES \[输入\]密钥1", key1, 8);

print\\_hex("3DES \[输入\]密钥2", key2, 8);

print\\_hex("3DES \[输入\]密钥3", key3, 8);

//3DES 加密

des(en\\_data, key1, en\\_out\\_put, DES\\_ENCRYPT);

des(en\\_out\\_put, key2, en\\_data, DES\\_DECRYPT);

des(en\\_data, key3, en\\_out\\_put, DES\\_ENCRYPT);

print\\_hex("3DES \[输出\]密文 ", en\\_out\\_put, 8);

printf("\\r\\n");

printf("\\r\\n++++++++3DES解密示例++++++++");

print\\_hex("3DES \[输入\]密文 ", en\\_out\\_put, 8);

print\\_hex("3DES \[输入\]密钥1", key1, 8);

print\\_hex("3DES \[输入\]密钥2", key2, 8);

print\\_hex("3DES \[输入\]密钥3", key3, 8);

//3DES 加密

des(en\\_out\\_put, key3, en\\_data, DES\\_DECRYPT);

des(en\\_data, key2, en\\_out\\_put, DES\\_ENCRYPT);

des(en\\_out\\_put, key1, en\\_data, DES\\_DECRYPT);

print\\_hex("3DES \[输出\]明文 ", en\\_data, 8);

printf("\\r\\n");

return;

}

# 二、AES密码算法

**1.实验目的**

1） 了解3DES的算法原理

2） 学习3DES算法的编程实现

**2.实验原理**

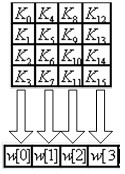
1) 算法原理

a) 1997年美国国家标准技术研究所（NIST）向全球发起征集保护敏感联邦信息的对称密钥加密算法。1999年8月上旬公布了对它们的第一轮评估结果，并选定了入围第二轮评估的五个算法(MARS, RC6, Rijndael,SERPENT and Twofish)。这五个算法最终角逐，其中Rijndael于2000年9月被定为21世纪美国的高级加密标准（Advanced Encryption Standard—AES）。

b) AES的加解密过程见流程图，下面简述各个过程的具体实现。

c) 密钥产生

Ø  AES首先将初始密钥按照列优先的顺序，输入到一个4\*4矩阵中。如下图一表示



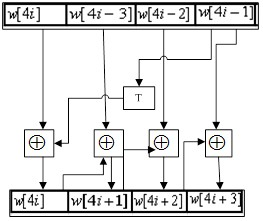
图一 密钥扩展算法部分流程图

Ø  这个4\*4矩阵的每一列的4个字节组成一个字，矩阵4列的4个字一次命名为w[0]，w[1]，w[2]，w[3]。它们构成了一个以字为单位的数组w。

Ø  接着，对数组w扩充40个新列，构成总共44列的扩展密钥数组。新列按照一下的递归方式产生：

如果i不是4的倍数，那么第i列由如下等式确定：w[i]=w[i-4]⨁w[i-1]；

如果i是4的倍数，那么第i列由如下等式确定：w[i]=w[i-4]⨁T（w[i-1]），其中，T是一个复杂函数。这一过程如下。如图二所示



图二 AES密钥扩展图

Ø  函数T由3部分组成：字循环、字节代换和轮常量异或，这3部分的作用分别如下：

  字循环：将1个字中的4个字节循环左移一个字节，即将输入字[b0,b1,b2,b3]变换成[b1,b2,b3,b0]。

  字节代换：对字循环的结果使用AES的S盒进行字节代换。   
  轮常量异或：将前两步的结果同轮常量Rcon[j]进行异或，其中j表示轮数。

Ø  字节代换

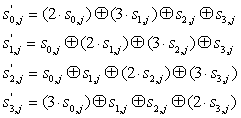
  字节代换操作和字节代换逆操作。AES的字节代换操作可以简化成一个简单的查表操作。AES定义了一个S盒和一个逆S盒。S盒用于加密查表，逆S盒用于解密查表。他们都是由16\*16字节组成的矩阵，即矩阵共有256元素，每个元素的内容是一个字节（8bit）的值，且每元素各不相同。

Ø  行移位

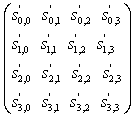
  行移位操作：此次操作中，只是将字节矩阵通过简单的左循环位移操作。当密钥长度为128bit时，状态矩阵的第Vi行左移i个字节，其中i=0,1,2,3。这使得列完全进行了重排，即在移位后的每列中，都包含有未移位前每列的一个字节。   
  行移位逆变换：行移位逆变换是将状态矩阵的每一列执行相反的移位操作，例如AES-128中，状态矩阵的第i行右移i个字节。

Ø  列混合

  列混合操作：列混合变换是通过矩阵相乘来实现的，经行移位后的状态矩阵与固定矩阵相乘，得到混淆后的状态矩阵，状态矩阵中的第j列（j=0,1,2,3）的列混合可以表示为：



  列混合逆运算：逆向列混合变换可由下式的矩阵乘法定义：



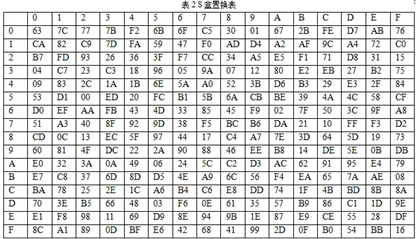
  显然，逆变换矩阵同正交变换矩阵的乘积恰好为单位矩阵。

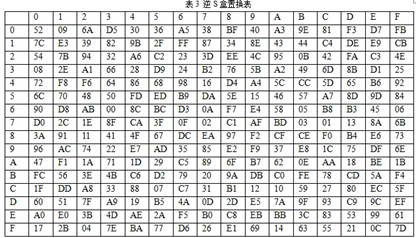
Ø   轮密钥加：轮密钥加是将128位轮密钥Ki同状态中的数据进行逐位异或操作。该过程可以看成是字逐位异或的结果，也可以看成字节级别或者位级别的操作。

2) 算法参数

  在执行AES算法操作时，主要用到的算法参数如下表1—表3所示，其中表1为轮常数值表，用于加解密时的密钥扩展；表2、表3为S盒置换表和逆S盒置换表，用于加解密过程中的字节代换。

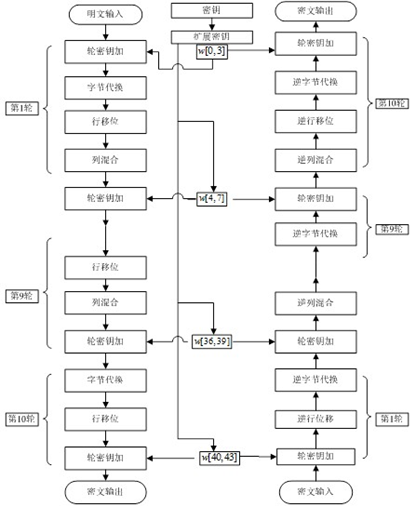






3) 算法流程

  AES算法加解密的主要流程如下。如图三所示



图三 AES算法加解密流程图

**3.实验环境**

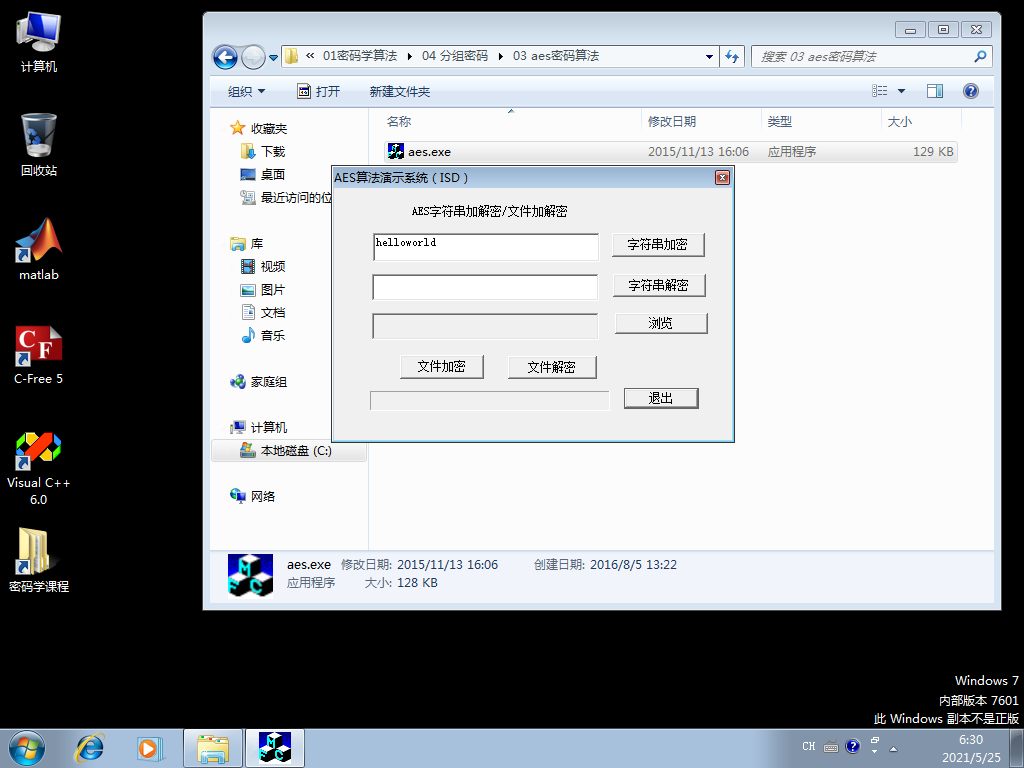
Windows 7

工具：C:\tools\密码学课程\01密码学算法\04 分组密码\03 aes密码算法

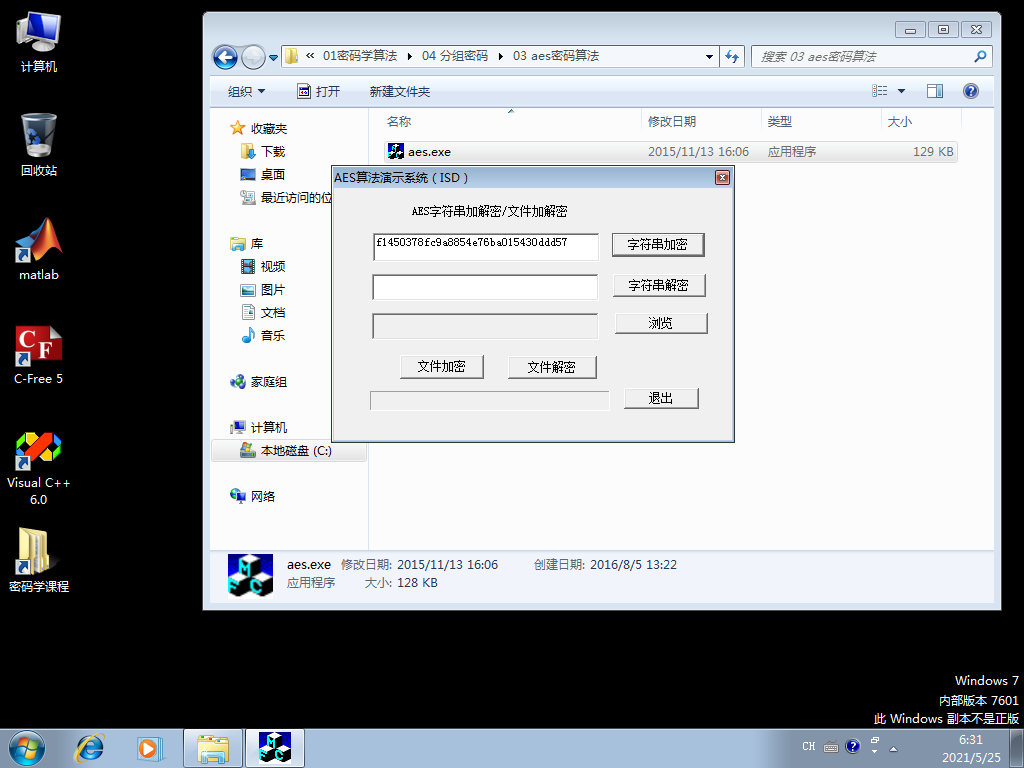
**4.实验步骤**

㈠字符串加解密

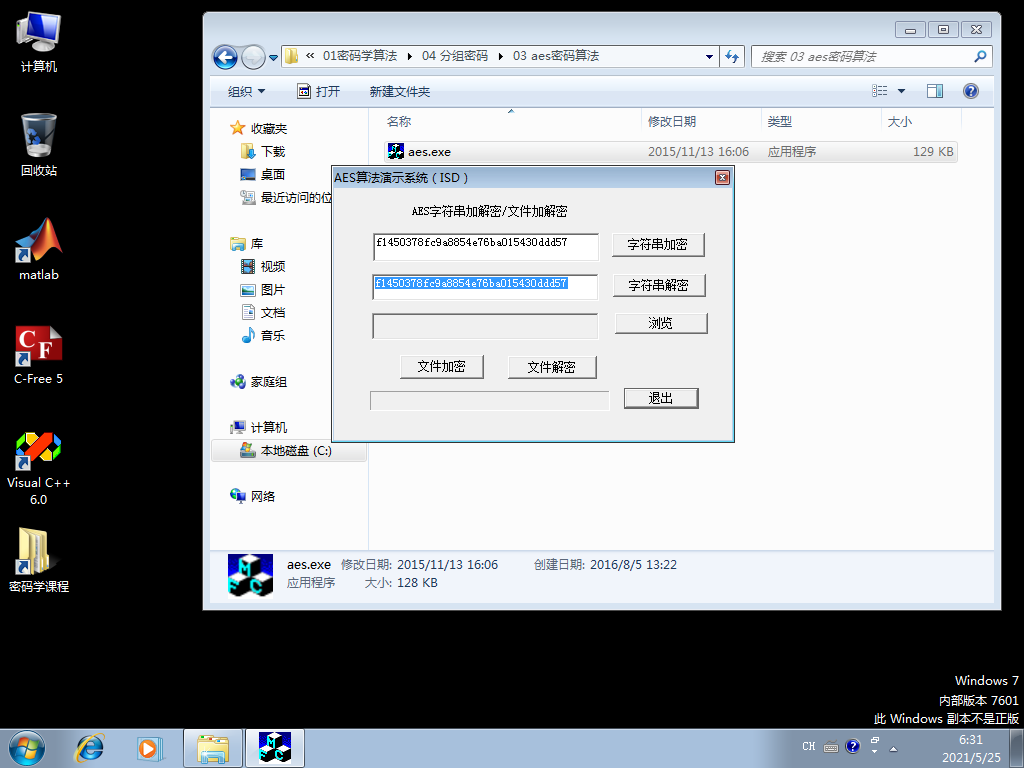
1.1运行文件【AES.exe1】，程序运行界面。



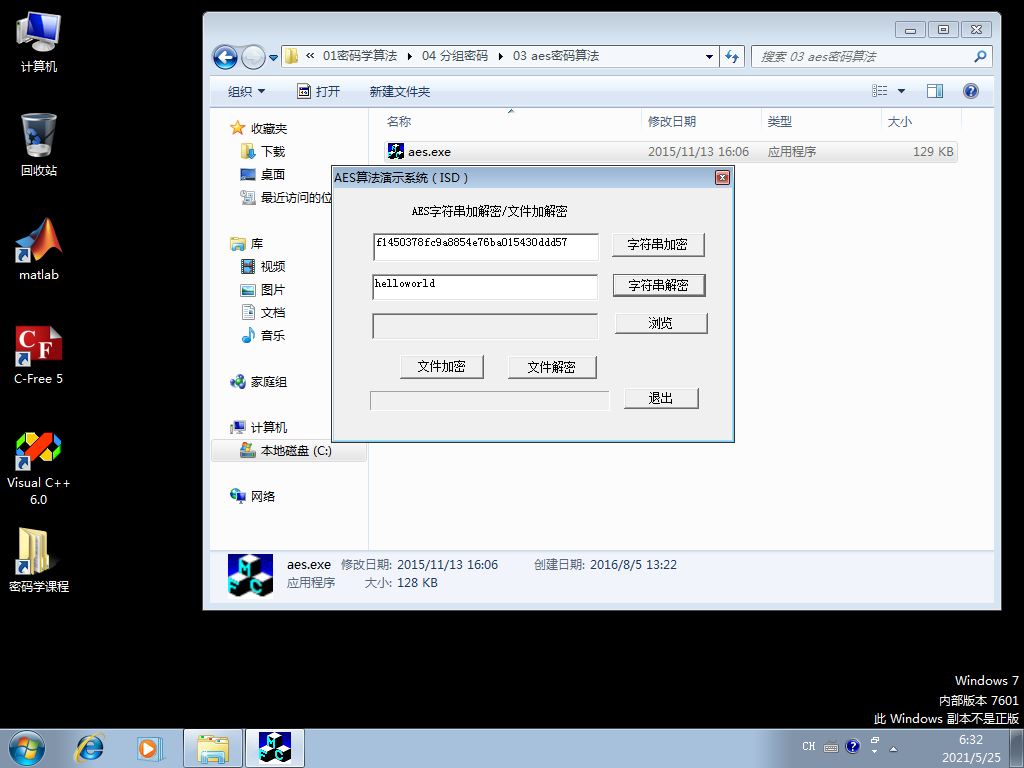
1.2在第一个框格中输入【he11owor1d】，点击【字符串加密】，即可在第一个框格中输入密文。



1.3将得到的密文字符串拷贝到第二个框格中。

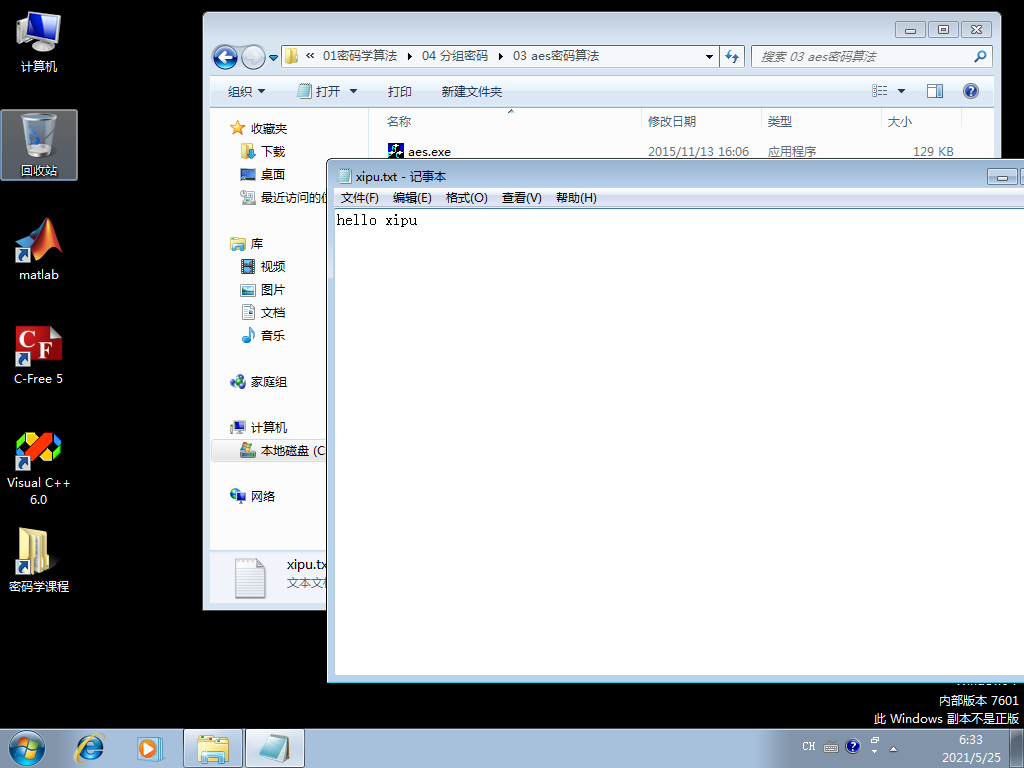


1.4点击【字符串解密】，即可得到明文。

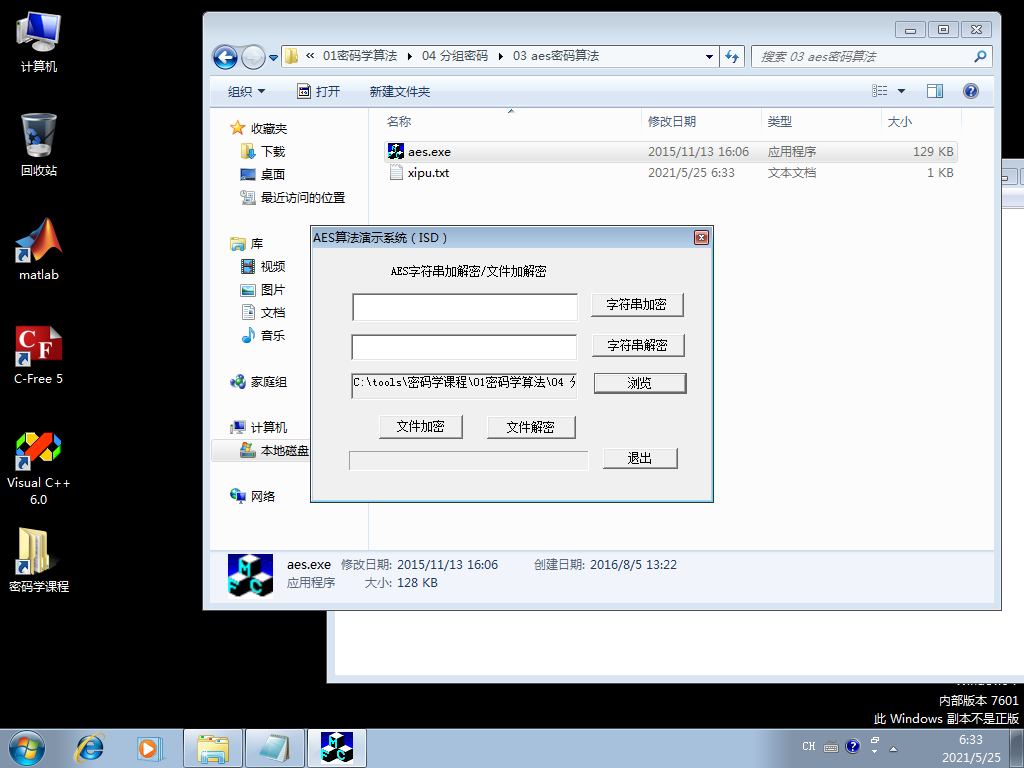


㈡文件文件加解密

2.1新建xipu.txt，其文件内容如下。



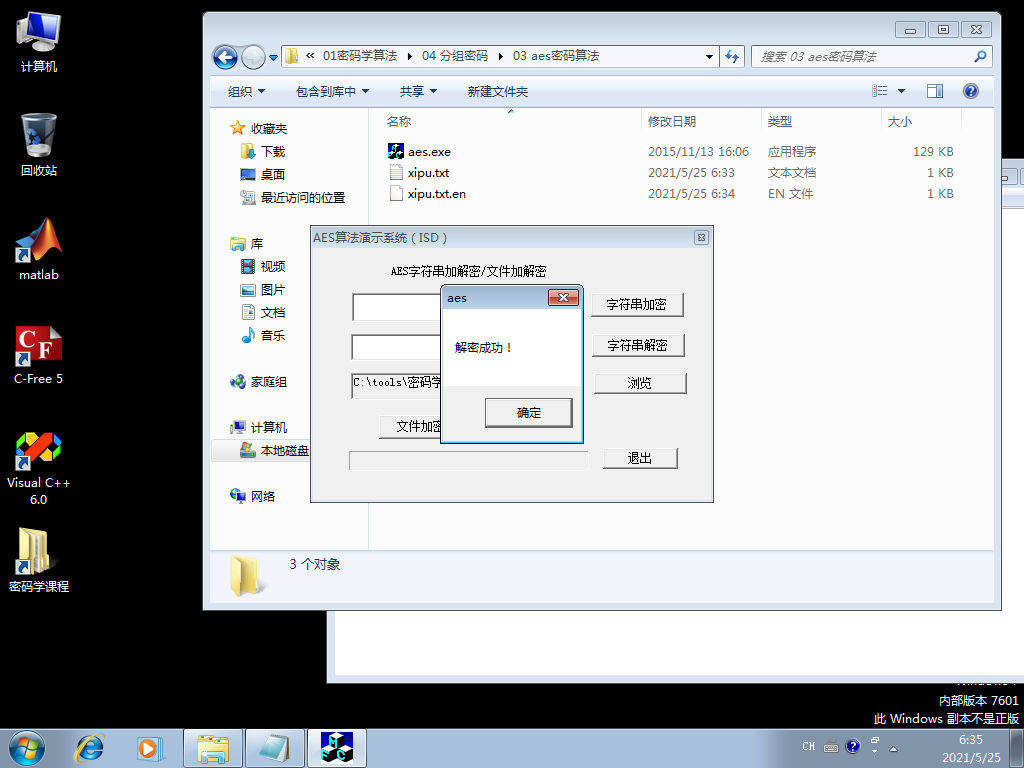
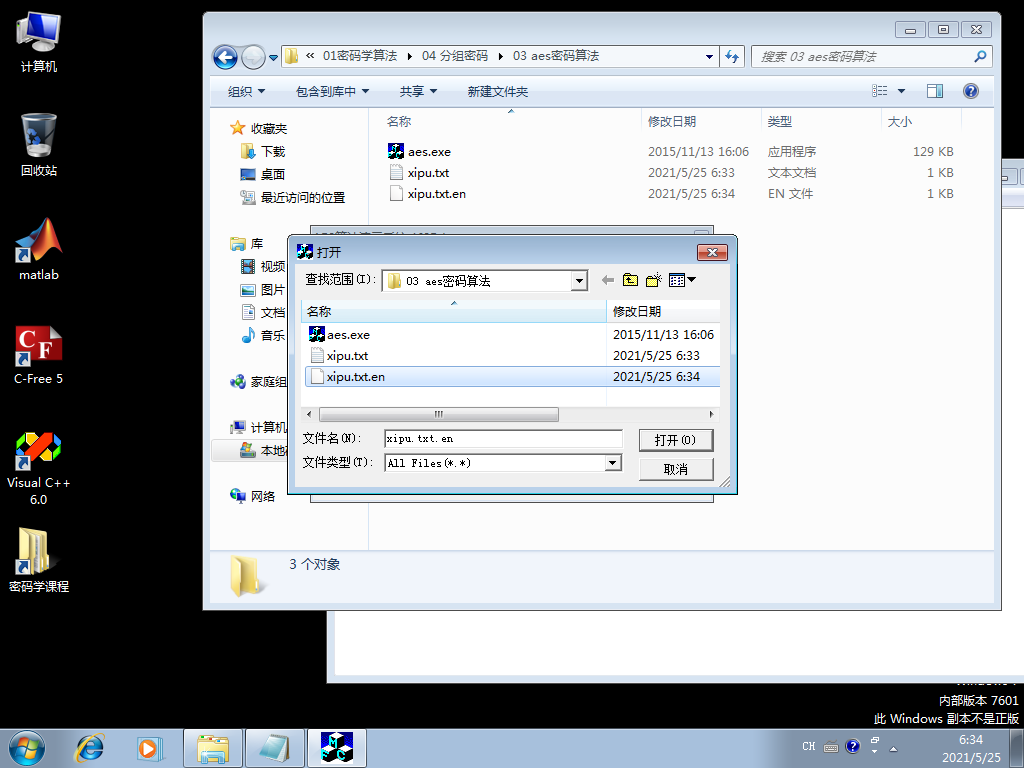
2.2点击【浏览】，选择要加密的文件【xipu.txt】。



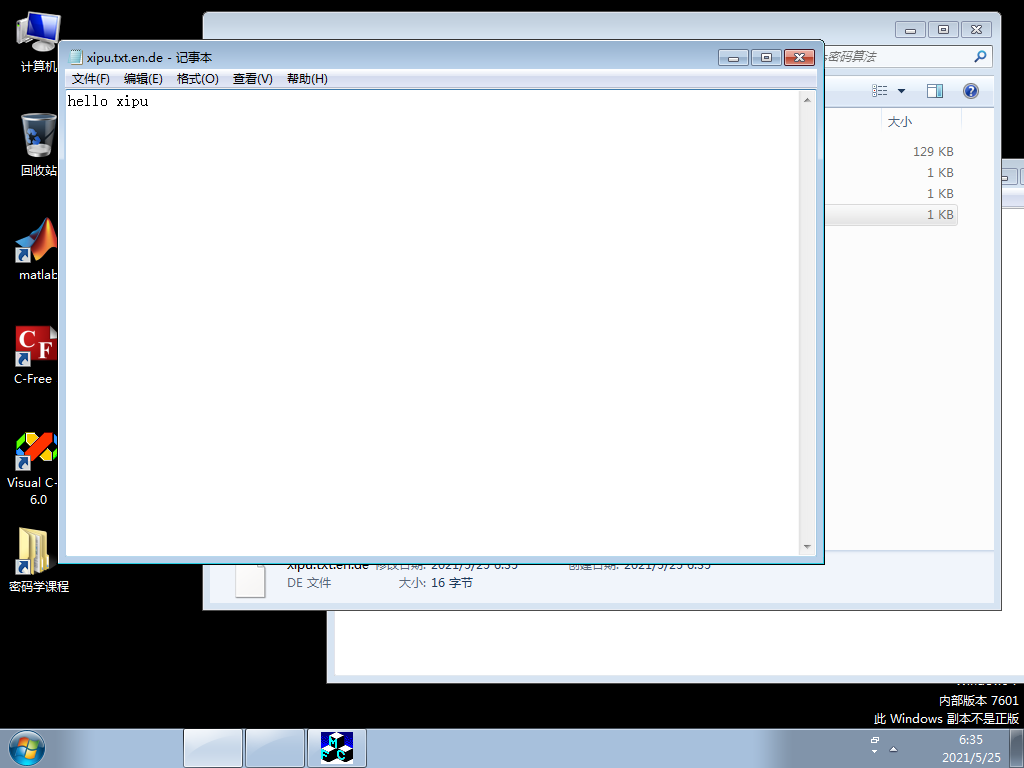
2.3点击【文件加密】，会提示加密成功，并生成一个以【.en】，为后缀名的文件。



2.4点击【浏览】，选择加密后的文件，点击【文件解密】，即可生成一个以.de】为后缀名的文件。



2.5用记事本打开文件【.de】，即可查看被解密的明文文件。



㈢源码阅读

3.1 下面是AES算法关键环节的源码。

ISD\_Aes::ISD\_KeyExpansion() //密钥扩展

{

memset(w,0,16\*15);

for(int row=0;row<Nk;row++) //拷贝seed 密钥

{

w[4\*row+0] = key[4\*row];

w[4\*row+1] = key[4\*row+1];

w[4\*row+2] = key[4\*row+2];

w[4\*row+3] = key[4\*row+3];

}

byte\* temp = new byte[4];

for(row=Nk;row<4\*(Nr+1);row++)

{

temp[0]=w[4\*row-4]; //当前列的前一列

temp[1]=w[4\*row-3];

temp[2]=w[4\*row-2];

temp[3]=w[4\*row-1];

if(row%Nk==0) //逢nk时，对当前列的前一列作特殊处理

{

temp=ISD\_SubWord(ISD\_RotWord(temp)); //先移位，再代换，最后和轮常量异或

temp[0] = (byte)( (int)temp[0] ^ (int) AesRcon[4\*(row/Nk)+0] );

temp[1] = (byte)( (int)temp[1] ^ (int) AesRcon[4\*(row/Nk)+1] );

temp[2] = (byte)( (int)temp[2] ^ (int) AesRcon[4\*(row/Nk)+2] );

temp[3] = (byte)( (int)temp[3] ^ (int) AesRcon[4\*(row/Nk)+3] );

}

else if ( Nk > 6 && (row % Nk == 4) )

{

temp= ISD\_SubWord(temp);

}

w[4\*row+0] = (byte) ( (int) w[4\*(row-Nk)+0] ^ (int)temp[0] );

w[4\*row+1] = (byte) ( (int) w[4\*(row-Nk)+1] ^ (int)temp[1] );

w[4\*row+2] = (byte) ( (int) w[4\*(row-Nk)+2] ^ (int)temp[2] );

w[4\*row+3] = (byte) ( (int) w[4\*(row-Nk)+3] ^ (int)temp[3] );

} }

ISD\_Aes::ISD\_MixColumns() //列混合

{

unsigned char temp[4\*4];

int i,j;

for(j=0;j<4;j++) //2 3 1 1 列混淆矩阵

{ //12 3 1

for(i=0;i<4;i++) //11 2 3

{ //31 1 2

temp[4\*i+j]=State[i][j];

}

}

for(j=0;j<4;j++)

{

State[0][j] = (unsigned char) ( (int)ISD\_gfmultby02(temp[0+j]) ^ (int)ISD\_gfmultby03(temp[4\*1+j]) ^

(int)ISD\_gfmultby01(temp[4\*2+j]) ^ (int)ISD\_gfmultby01(temp[4\*3+j]) );

State[1][j] = (unsigned char) ( (int)ISD\_gfmultby01(temp[0+j]) ^ (int)ISD\_gfmultby02(temp[4\*1+j]) ^

(int)ISD\_gfmultby03(temp[4\*2+j]) ^ (int)ISD\_gfmultby01(temp[4\*3+j]) );

State[2][j] = (unsigned char) ( (int)ISD\_gfmultby01(temp[0+j]) ^ (int)ISD\_gfmultby01(temp[4\*1+j]) ^

(int)ISD\_gfmultby02(temp[4\*2+j]) ^ (int)ISD\_gfmultby03(temp[4\*3+j]) );

State[3][j] = (unsigned char) ( (int)ISD\_gfmultby03(temp[0+j]) ^ (int)ISD\_gfmultby01(temp[4\*1+j]) ^

(int)ISD\_gfmultby01(temp[4\*2+j]) ^ (int)ISD\_gfmultby02(temp[4\*3+j]) );

}

}

ISD\_Aes::ISD\_ShiftRows()

{

unsigned char temp[4\*4]; //行移位

int i,j;

for(j=0;j<4;j++)

{

for(i=0;i<4;i++)

{

temp[4\*i+j]=State[i][j];

}

}

for(i=1;i<4;i++)

{

for(j=0;j<4;j++)

{

if(i==1)State[i][j]=temp[4\*i+(j+1)%4]; //第一行左移1位

else if(i==2)State[i][j]=temp[4\*i+(j+2)%4]; //第二行左移2位

else if(i==3)State[i][j]=temp[4\*i+(j+3)%4]; //第三行左移3位

}

}

}

# 三、RSA密码算法

**1.实验目的**

1） 学习RSA密码算法的原理

2） 学习RSA密码算法的编程实现

**2.实验原理**

**1) 算法原理**

a) RSA公钥加密算法是1977年由Ron Rivest、Adi Shamirh和LenAdleman在美国麻省理工学院开发的。RSA算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但那时想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥。

b) RSA算法是一种非对称密码算法，所谓非对称，就是指该算法需要一对密钥，使用其中一个加密，则需要用另一个才能解密。

c) 为了产生RSA公私钥对，需要进行以下几个步骤:

Ø  选取两个大素数，*p*和*q*（目前两个数的长度都接近512bit是安全的），并计算他们的乘积得n=pq。

Ø  根据欧拉函数，不大于*n*且与*n*互质的整数个数为说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_39589.gif。

Ø  随机选择整数说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_68044.gif作为公钥，要求满足说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_59196.gif，即说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_21689.gif互素。

Ø  用欧几里德扩展算法计算私钥*d*，以满足说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_77331.gif，即说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_11035.gif。则*e*和*n*是公钥，*d*是私钥。

d) 加密过程

  使用加密算法说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_50771.gif，计算出密文说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_31225.gif；  
e) 解密过程   
Ø  接收方收到密文*C*，并把密文*C*按长度为*L*分组得说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_69760.gif；加密时首先将明文比特串分组，使得每个分组对应的十进制小于*n*，即分组长度小于说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_22393.gif，然后对每个明文分组说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_10412.gif作加密运算，具体过程分为如下几步：

Ø  获得接收方公钥说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_57799.gif；

Ø  把消息M分组为长度为说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231841_73388.gif 的消息分组说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_77852.gif；

Ø  将密文C发送给接收方。

Ø  使用私钥*d*和解密算法说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_60448.gif，计算说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_43581.gif；

Ø  得到明文消息说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_40585.gif。

**2) 算法参数**

  参数主要包括：*p、q、e、d、n*。*p*和*q*为大素数，说明: http://192.168.1.205/Public/editor/attached/image/20130723/20130723231842_99492.gif为公钥，*d*为私钥，*n*为*p*和*q*的乘积。*e*和*n*应公开，两个素数 *p*和*q*不再需要，可销毁，但绝不能泄露。另外，为了防止低指数攻击，*e*不能选取太小的数。

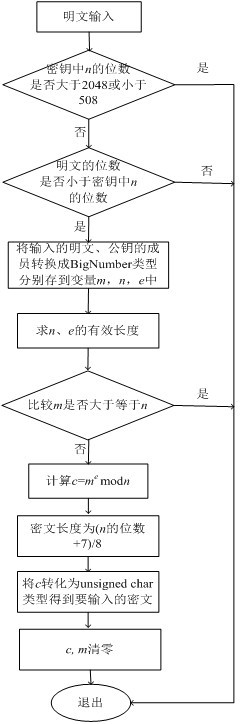
**3) 算法流程**

a）RSA密钥生成部分代码流程图如下。如图一所示



图一  RSA密钥生成流程图

b）RSA加密部分流程图如下。如图二所示

  
图二  RSA加密部分流程图

c）RSA解密部分流程图如下。如图三所示



图三  RSA解密部分流程图

**3.实验环境**

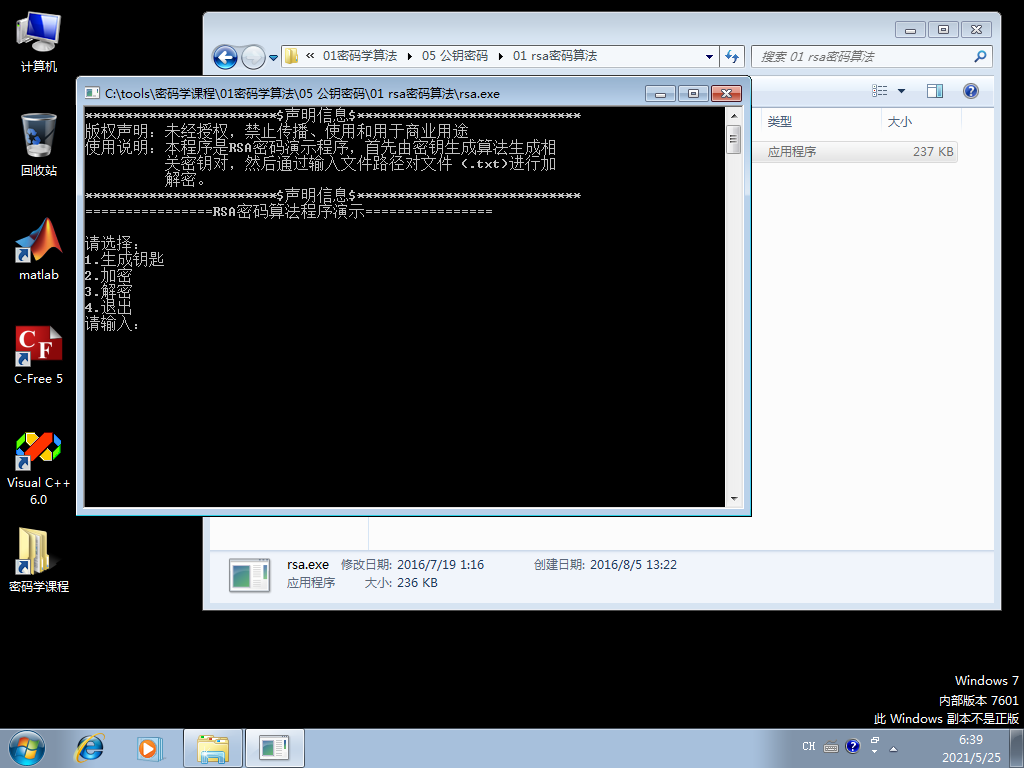
Windows 7

工具：C:\tools\密码学课程\01密码学算法\05 公钥密码\01 rsa密码算法

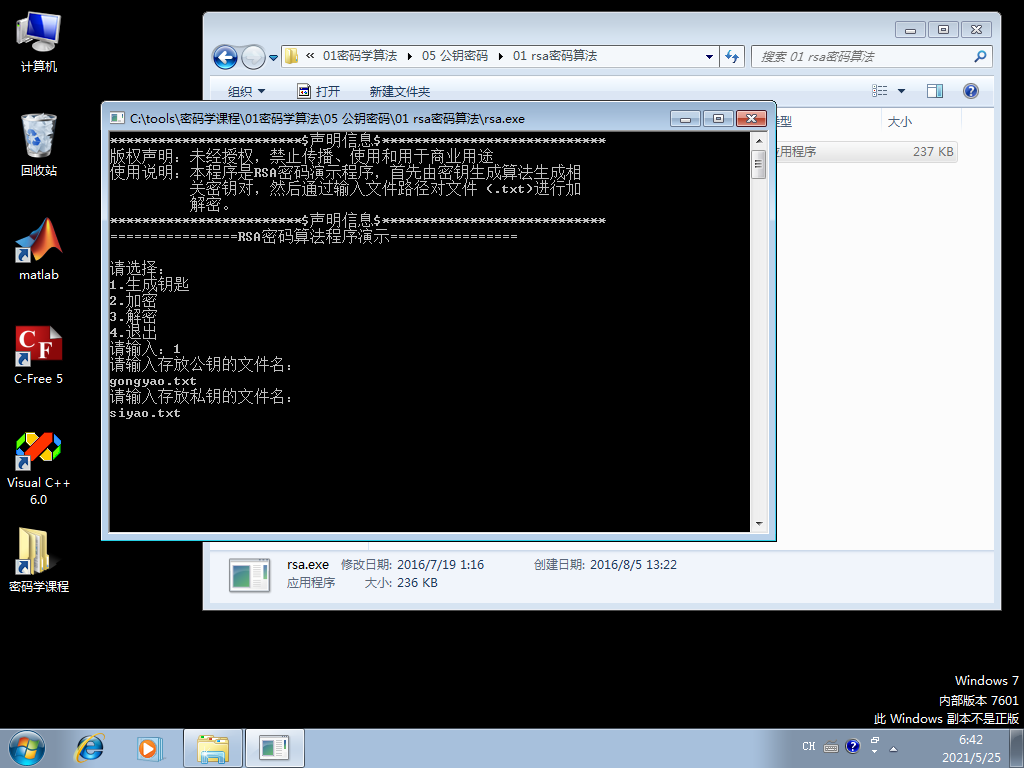
**4.实验步骤**

㈠查看RSA算法运行结果并学习核心算法

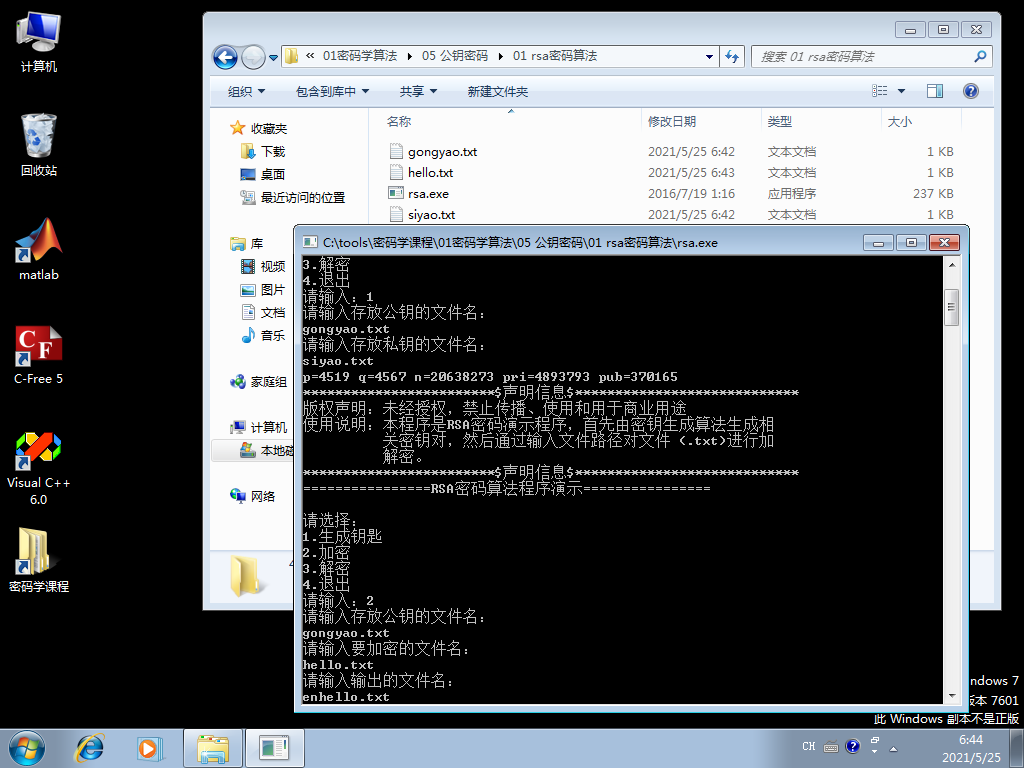
1.1运行rsa.exe程序，结果如下。



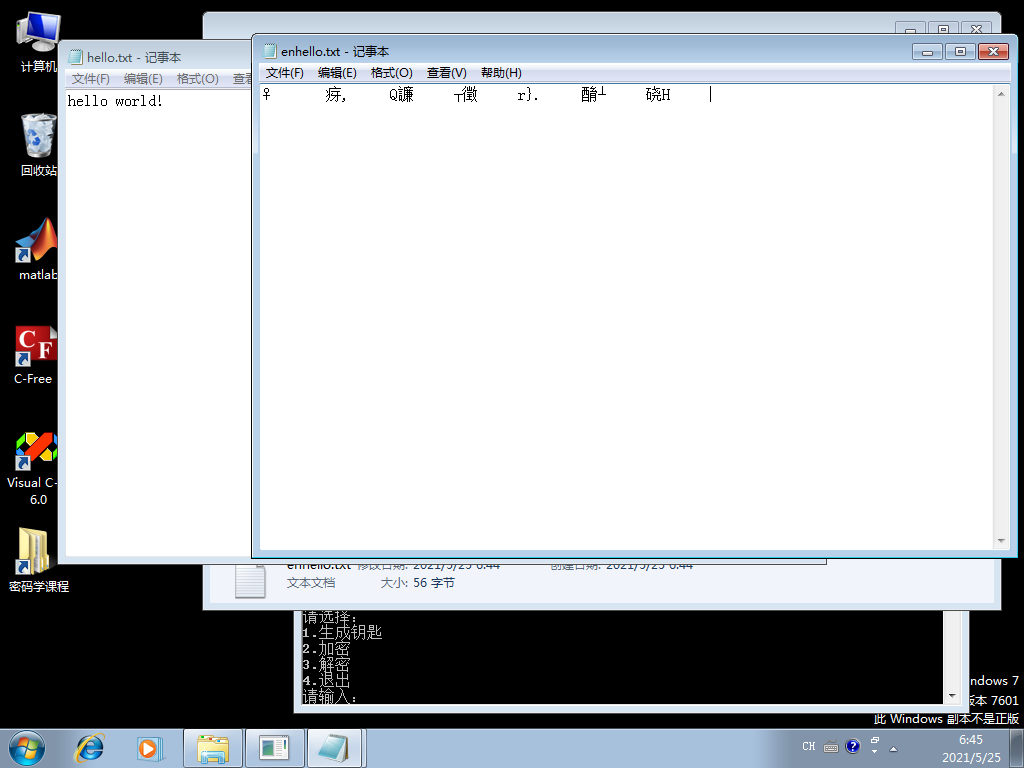
1.2输入1，输入存放公钥和私钥的文件名，即可在程序同级目录下生成相应文件。



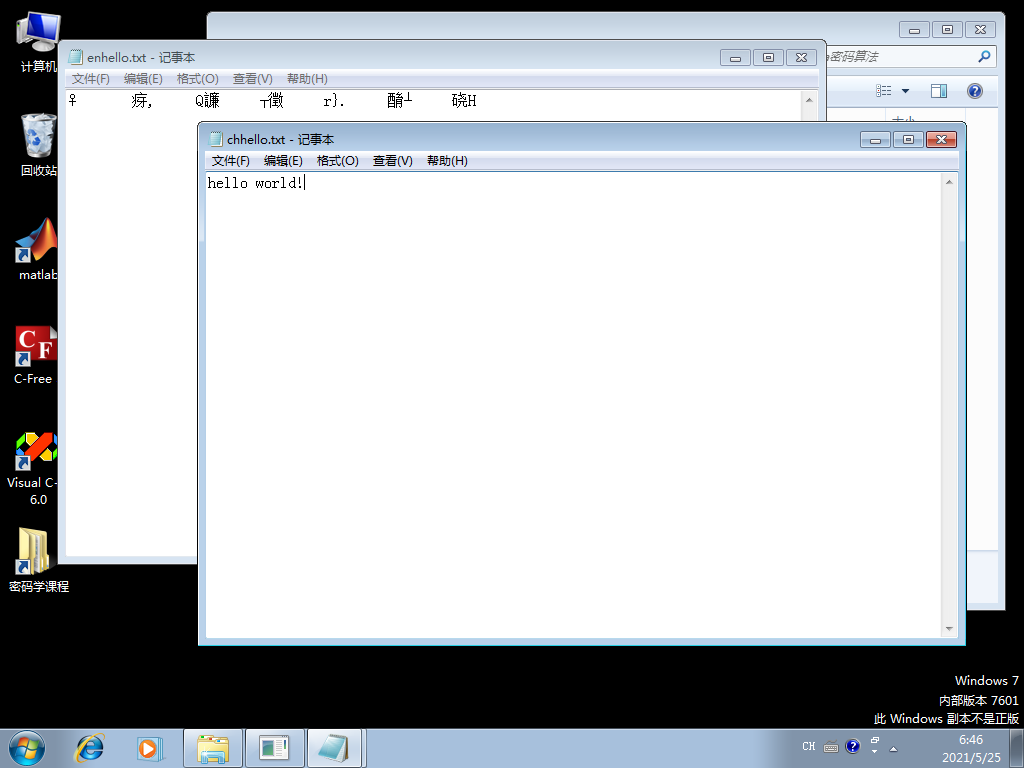
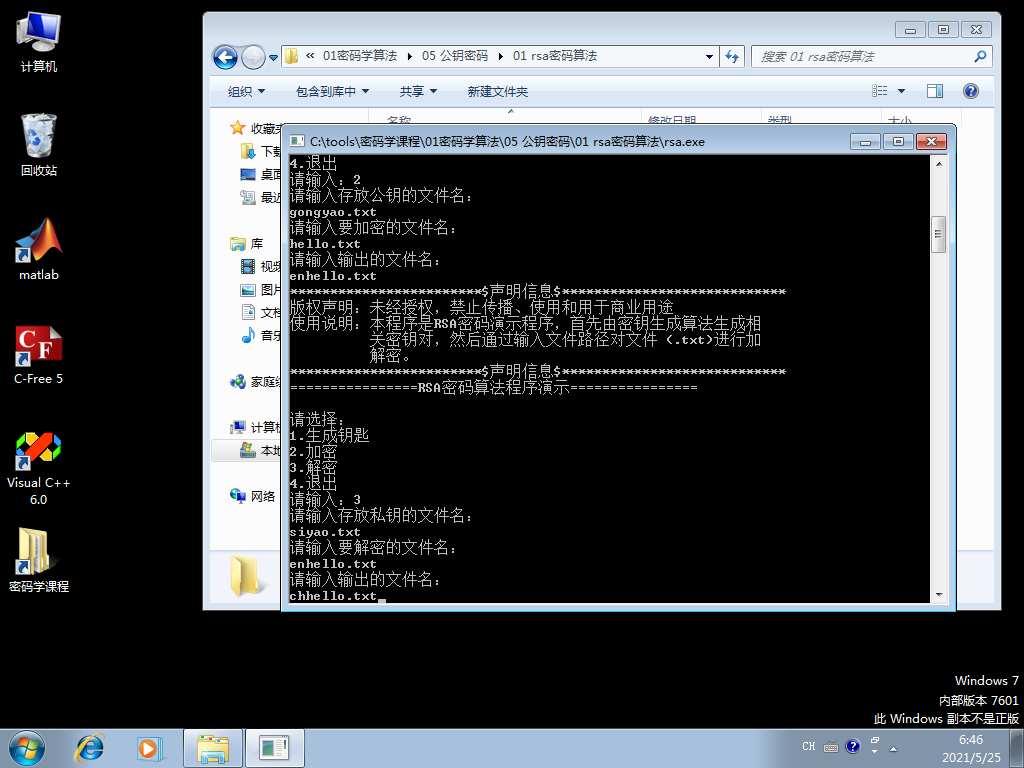
1.3选择2，输入存放公钥的文件名和需要加密的文件【he11o.txt】(若无该文件，可先行在软件目录下创建，内容为he11o Wor1d!）和需要输出的文件名【enhe11o.txt】。



1.4对比明文文件内容和密文文件内容。



1.5解密文件,进行类似的操作即可。



1.6源代码如下所示。

class BigNum {

public:

int length; //大数长度

int signal; //大数符号

nsigned long array [len1]; //大数绝对值

BigNum(); //构造函数

BigNum(unsigned \_int64); //构造函数用于赋初始值?

BigNum::BigNum(string s); //读入字符串

BigNum(BigNum const& A); //复制大数，const保护了原对象的属性

~BigNum(); //析构函数

BigNum operator+(BigNum & A); //运算符+重载

BigNum operator-(BigNum & A); //运算符-重载

BigNum operator\\*(BigNum & A); //运算符\\*重载

BigNum operator/(BigNum & A); //运算符/重载

BigNum operator%(BigNum & A); //运算符%重载

BigNum operator-(void); //负号重载

int operator==(BigNum& A); //等于号重载

int operator!=(BigNum& A); //不等号重载

int Compare(BigNum const& A); //比较两大数绝对值大小

void GeneratePrime(void); //产生素数

int Rabin\\_Miller(void); //拉宾米勒算法用于素数测试

void Random(int a); //随机产生一个大数

void Random(BigNum A); //随机产生一个小于A的大数

void print(void); //输出大数

void printS(void); //输出字符串

BigNum power\\_mod(BigNum& A, BigNum& B); //模幂算法计算X^A mod B

BigNum ex\\_euclid(BigNum a,BigNum b); //扩展欧几里德算法

}

//RSA系统初始化部分代码：

RSA::RSA\\_Generated\\_Parameter () {

q.GeneratePrime();

while(p==q) //判断两个素数不等

q.GeneratePrime();

temp=p-I;

t=q-I;

t=t\\*temp;?

cout<<"素数p为："<<endl;???? //输出素数

p.print();

cout<<endl;

cout<<"素数q为："<<endl;

q.print();

cout<<endl;

cout<<"公钥n为："<<endl;

n=p\\*q;

n.print();

cout<<endl;

cout<<"公钥e为："<<endl;

e.print();

cout<<endl;

d.ex\\_euclid(e,t);??????????????//计算私钥d

cout<<"私钥d为："<<endl;

d.print();

cout<<endl;

}

//RSA加解密部分代码：

RSA::RSA\\_Decrypt\\_Crypt (char s) {

a=BigNum(A);

cout<<endl;

b=a.power\\_mod(e,n); //产生密文b

cout<<"加密结果为："<<endl;

b.print();

cout<<endl;

c=b.power\\_mod(d,n);??????????//解密

cout<<"解密后的结果为："<<endl<<endl;

c.printS();

}