实验项目名称： AES密码算法 实验学时： 8

同组学生姓名： 无 实验地点： 科技楼4-1204

实验日期： 2023.11.20 实验成绩：

批改教师： 黄丹丹 批改时间： 2023.11.29

实验4 AES密码算法

一、基于OBE模式的实验目的和要求

1、了解分组密码的起源与涵义。

2、掌握AES密码的加解密原理。

3、编程设计AES密码算法。

二、实验仪器和设备

Visual C/C++

三、实验过程

1、2000年10月2日美国商务部部长Norman Y. Mineta宣布经过三年来世界著名密码专家之间的竞争,比利时密码专家 Joan Daemen 和 Vincent Riimen 提出的“ Rijndael 数据加密算法”最终获胜。2001年11月26日，NIST正式公布高级加密标准AES，并于2002年5月26日正式生效，成为美国的官方政府标准。AES被设计成三个密钥长度128/192/256比特用于加密长度为128比特的分组，相应的轮数为10/12/14。AES算法是SPN结构的代表。

2、算法细节

**明文**是128bit序列，划分为16个字节，并把字节数据块按a00, a10, a20, a30, a01, a11, a21, a31, a02, a12, a22, a32, a03, a13, a23, a33 顺序映射为状态字节矩阵。

**密钥长度为128比特**的AES加密流程如下：

（1）密钥编排：由密钥扩展算法将128比特的初始密钥扩展为11个128比特的轮密钥K0, K1…K10, 每一个轮密钥同样表示为字节矩阵。

（2）密钥白化（初始轮）：将明文状态矩阵与第一个轮密钥K0 异或加运算。

（3）执行9轮完全相同的轮变换，轮函数分为四步（中间轮）：字节代换、行移位、列混合、轮密钥加。

（4）执行最后一轮轮变换（最后轮）：字节代换、行移位、轮密钥加。

（5）步骤（4）输出的字节矩阵按照顺序排列即为密文。

详见图1：

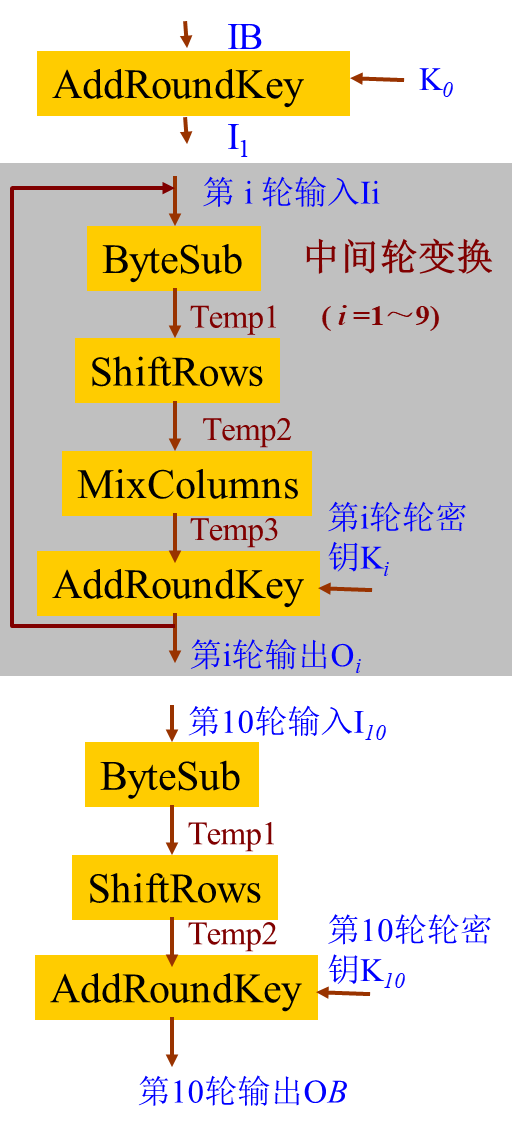


图1 AES加密

解密算法与加密算法的计算网络都相同，只是将各计算部件换为对应的逆部件，解密算法的轮密钥依次为：K10, InvMixColumns(K9), …, InvMixColumns(K1), K0。列混合变换中乘法使用的4字节向量为 ( 03 01 01 02)，逆列混合变换中乘法使用的4字节向量为 ( 0B 0D 09 0E)。

密钥扩展算法将128比特的初始密钥扩展为44个字W0, W1…W43, 每一个字为32比特，算法流程如图2：

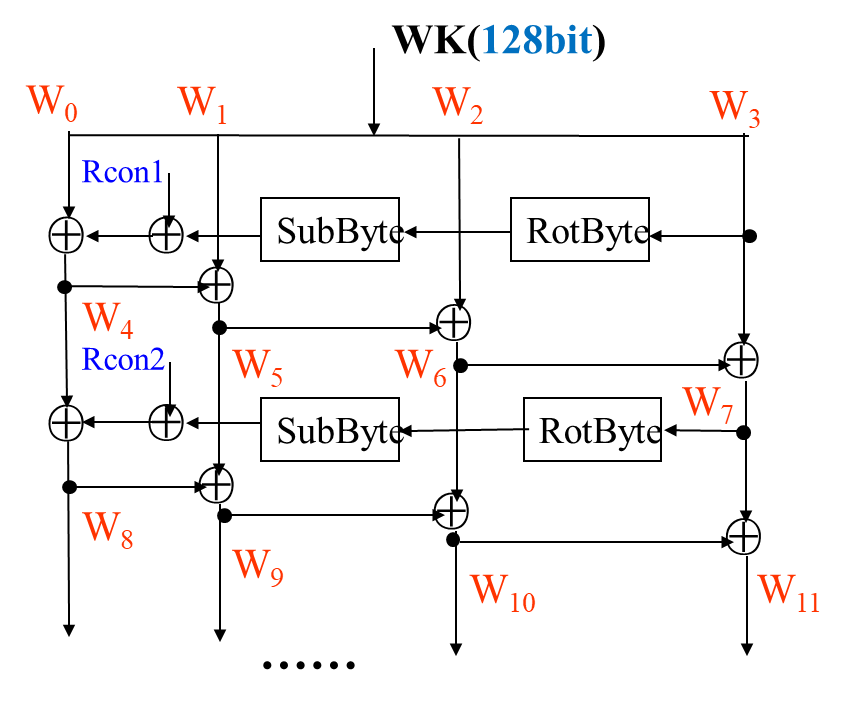


图2 AES密钥编排

3、编写设计AES密码算法

（1）输入128比特明文和密钥，利用AES密码对明文加密并输出密文。

（2）输入AES密码加密的128比特密文和密钥，对密文进行解密。

（3）要求有对应的程序调试记录和验证记录。

提示点：

1. 使用模块化编程，将算法实现的功能写成子函数后供上一层函数调用。
2. 编写函数时多输出中间变量查看结果。
3. 首先需要定义AES密码算法中的基本函数，例如字节代换、行移位、列混合、轮密钥加、密钥扩展算法、有限域上的乘法运算（字节乘法）等。部分函数参考代码如下：

void AddRoundKey( unsigned char \*a , unsigned char \*Key ) { // 轮密钥加

for( int i = 0 ; i < 16 ; i ++ )

a[i] ^= Key[i] ;

}

void SubBytes( unsigned char \*input ) { // S盒字节代换

for( int i = 0 ; i < 16 ; i ++ )

input[i] = S[input[i]] ;

}

void InvSubBytes( unsigned char \*input ) { // 逆S盒字节代换

for( int i = 0 ; i < 16 ; i ++ )

input[i] = IS[input[i]] ;

}

void ShiftRows( unsigned char \*a ) { // 行移位—矩阵按列展开

unsigned char b[16] ;

b[ 0] = a[ 0] ; b[ 4] = a[ 4] ; b[ 8] = a[ 8] ; b[12] = a[12] ;

b[ 1] = a[ 5] ; b[ 5] = a[ 9] ; b[ 9] = a[13] ; b[13] = a[ 1] ;

b[ 2] = a[10] ; b[ 6] = a[14] ; b[10] = a[ 2] ; b[14] = a[ 6] ;

b[ 3] = a[15] ; b[ 7] = a[ 3] ; b[11] = a[ 7] ; b[15] = a[11] ;

for( int i = 0 ; i < 16 ; i ++ )

a[i] = b[i] ;

}

void InvShiftRows( unsigned char \*a ) { // 逆行移位变换

unsigned char b[16] ;

b[ 0] = a[ 0] ; b[ 4] = a[ 4] ; b[ 8] = a[ 8] ; b[12] = a[12] ;

b[ 1] = a[13] ; b[ 5] = a[ 1] ; b[ 9] = a[ 5] ; b[13] = a[ 9] ;

b[ 2] = a[10] ; b[ 6] = a[14] ; b[10] = a[ 2] ; b[14] = a[ 6] ;

b[ 3] = a[ 7] ; b[ 7] = a[11] ; b[11] = a[15] ; b[15] = a[ 3] ;

for( int i = 0 ; i < 16 ; i ++ )

a[i] = b[i] ;

}

void MixColumns( unsigned char \*a ) { // 列混合

unsigned char b[16] ;

b[ 0] = Mul(0x02,a[0]) ^ Mul(0x03,a[1]) ^ a[2] ^ a[3];

b[ 1] = Mul(0x02,a[1]) ^ Mul(0x03,a[2]) ^ a[3] ^ a[0];

b[ 2] = Mul(0x02,a[2]) ^ Mul(0x03,a[3]) ^ a[0] ^ a[1];

b[ 3] = Mul(0x02,a[3]) ^ Mul(0x03,a[0]) ^ a[1] ^ a[2];

b[ 4] = Mul(0x02,a[4]) ^ Mul(0x03,a[5]) ^ a[6] ^ a[7];

b[ 5] = Mul(0x02,a[5]) ^ Mul(0x03,a[6]) ^ a[7] ^ a[4];

b[ 6] = Mul(0x02,a[6]) ^ Mul(0x03,a[7]) ^ a[4] ^ a[5];

b[ 7] = Mul(0x02,a[7]) ^ Mul(0x03,a[4]) ^ a[5] ^ a[6];

b[ 8] = Mul(0x02,a[8]) ^ Mul(0x03,a[9]) ^ a[10] ^ a[11];

b[ 9] = Mul(0x02,a[9]) ^ Mul(0x03,a[10]) ^ a[11] ^ a[8];

b[10] = Mul(0x02,a[10]) ^ Mul(0x03,a[11]) ^ a[8] ^ a[9];

b[11] = Mul(0x02,a[11]) ^ Mul(0x03,a[8]) ^ a[9] ^ a[10];

b[12] = Mul(0x02,a[12]) ^ Mul(0x03,a[13]) ^ a[14] ^ a[15];

b[13] = Mul(0x02,a[13]) ^ Mul(0x03,a[14]) ^ a[15] ^ a[12];

b[14] = Mul(0x02,a[14]) ^ Mul(0x03,a[15]) ^ a[12] ^ a[13];

b[15] = Mul(0x02,a[15]) ^ Mul(0x03,a[12]) ^ a[13] ^ a[14];

for( int i = 0 ; i < 16 ; i ++ )

a[i] = b[i] ;

}

1. 所使用参数的初始定义

unsigned char S[256] = { // S盒--按行展开为一元数组0x63,0x7C,0x77,0x7B,0xF2,0x6B,0x6F,0xC5,0x30,0x01,0x67,0x2B,0xFE,0xD7,0xAB,0x76,

0xCA,0x82,0xC9,0x7D,0xFA,0x59,0x47,0xF0,0xAD,0xD4,0xA2,0xAF,0x9C,0xA4,0x72,0xC0,

0xB7,0xFD,0x93,0x26,0x36,0x3F,0xF7,0xCC,0x34,0xA5,0xE5,0xF1,0x71,0xD8,0x31,0x15, 0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x12,0x80,0xE2,0xEB,0x27,0xB2,0x75, 0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3B,0xD6,0xB3,0x29,0xE3,0x2F,0x84, 0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xCB,0xBE,0x39,0x4A,0x4C,0x58,0xCF, 0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF9,0x02,0x7F,0x50,0x3C,0x9F,0xA8, 0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB6,0xDA,0x21,0x10,0xFF,0xF3,0xD2, 0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA7,0x7E,0x3D,0x64,0x5D,0x19,0x73, 0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xEE,0xB8,0x14,0xDE,0x5E,0x0B,0xDB, 0xE0,0x32,0x3A,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79, 0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,0xA9,0x6C,0x56,0xF4,0xEA,0x65,0x7A,0xAE,0x08, 0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xDD,0x74,0x1F,0x4B,0xBD,0x8B,0x8A, 0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x35,0x57,0xB9,0x86,0xC1,0x1D,0x9E, 0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1E,0x87,0xE9,0xCE,0x55,0x28,0xDF, 0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x99,0x2D,0x0F,0xB0,0x54,0xBB,0x16};

unsigned char IS[256] = { // 逆S盒

0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb,

0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb,

0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0x0b,0x42,0xfa,0xc3,0x4e,

0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,0x6d,0x8b,0xd1,0x25,

0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92,

0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84,

0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,0x05,0xb8,0xb3,0x45,0x06,

0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,0x03,0x01,0x13,0x8a,0x6b,

0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,0xe6,0x73,

0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e,

0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0x0e,0xaa,0x18,0xbe,0x1b,

0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4,

0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,0x59,0x27,0x80,0xec,0x5f,

0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9c,0xef,

0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61,

0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0x0c,0x7d} ;

4、程序代码：

#include <stdio.h># include <stdlib.h># include <string.h>unsigned char S[256] = {    0x63,0x7C,0x77,0x7B,0xF2,0x6B,0x6F,0xC5,0x30,0x01,0x67,0x2B,0xFE,0xD7,0xAB,0x76,    0xCA,0x82,0xC9,0x7D,0xFA,0x59,0x47,0xF0,0xAD,0xD4,0xA2,0xAF,0x9C,0xA4,0x72,0xC0,      
    0xB7,0xFD,0x93,0x26,0x36,0x3F,0xF7,0xCC,0x34,0xA5,0xE5,0xF1,0x71,0xD8,0x31,0x15,      
    0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x12,0x80,0xE2,0xEB,0x27,0xB2,0x75,      
    0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3B,0xD6,0xB3,0x29,0xE3,0x2F,0x84,      
    0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xCB,0xBE,0x39,0x4A,0x4C,0x58,0xCF,      
    0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF9,0x02,0x7F,0x50,0x3C,0x9F,0xA8,      
    0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB6,0xDA,0x21,0x10,0xFF,0xF3,0xD2,      
    0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA7,0x7E,0x3D,0x64,0x5D,0x19,0x73,      
    0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xEE,0xB8,0x14,0xDE,0x5E,0x0B,0xDB,      
    0xE0,0x32,0x3A,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79,      
    0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,0xA9,0x6C,0x56,0xF4,0xEA,0x65,0x7A,0xAE,0x08,      
    0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xDD,0x74,0x1F,0x4B,0xBD,0x8B,0x8A,      
    0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x35,0x57,0xB9,0x86,0xC1,0x1D,0x9E,      
    0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1E,0x87,0xE9,0xCE,0x55,0x28,0xDF,     
    0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x99,0x2D,0x0F,0xB0,0x54,0xBB,0x16  
};unsigned char IS[256] = {    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D  
};void SubBytes( unsigned char \*input )   {      
    for( int i = 0 ; i < 16 ; i ++ )        input[i] = S[input[i]] ;}void InvSubBytes(unsigned char \*input) {    for (int i = 0; i < 16; i++) {        input[i] = IS[input[i]];    }}void ShiftRows( unsigned char \*a )     {    unsigned char b[16] ;    b[ 0] = a[ 0] ;    b[ 4] = a[ 4] ;    b[ 8] = a[ 8] ;    b[12] = a[12] ;    b[ 1] = a[ 5] ;    b[ 5] = a[ 9] ;    b[ 9] = a[13] ;    b[13] = a[ 1] ;    b[ 2] = a[10] ;    b[ 6] = a[14] ;    b[10] = a[ 2] ;    b[14] = a[ 6] ;    b[ 3] = a[15] ;    b[ 7] = a[ 3] ;    b[11] = a[ 7] ;    b[15] = a[11] ;    for( int i = 0 ; i < 16 ; i ++ )        a[i] = b[i] ;}void InvShiftRows(unsigned char \*a) {    unsigned char b[16];    b[ 0] = a[ 0];   b[ 4] = a[ 4];   b[ 8] = a[ 8];   b[12] = a[12];    b[ 1] = a[13];   b[ 5] = a[ 1];   b[ 9] = a[ 5];   b[13] = a[ 9];    b[ 2] = a[10];   b[ 6] = a[14];   b[10] = a[ 2];   b[14] = a[ 6];    b[ 3] = a[ 7];   b[ 7] = a[11];   b[11] = a[15];   b[15] = a[ 3];    for (int i = 0; i < 16; i++) {        a[i] = b[i];    }}unsigned char Mul(unsigned char a, unsigned char b) {    unsigned char result = 0;    while (b > 0) {        if (b & 1) {            result ^= a;        }        if (a & 0x80) {            a = (a << 1) ^ 0x1b;        } else {            a <<= 1;        }        b >>= 1;    }    return result;}static void MixColumns(unsigned char \*a){    unsigned char b[16] ;    b[ 0] = Mul(0x02,a[0]) ^ Mul(0x03,a[1]) ^ a[2] ^ a[3];    b[ 1] = Mul(0x02,a[1]) ^ Mul(0x03,a[2]) ^ a[3] ^ a[0];    b[ 2] = Mul(0x02,a[2]) ^ Mul(0x03,a[3]) ^ a[0] ^ a[1];    b[ 3] = Mul(0x02,a[3]) ^ Mul(0x03,a[0]) ^ a[1] ^ a[2];    b[ 4] = Mul(0x02,a[4]) ^ Mul(0x03,a[5]) ^ a[6] ^ a[7];    b[ 5] = Mul(0x02,a[5]) ^ Mul(0x03,a[6]) ^ a[7] ^ a[4];    b[ 6] = Mul(0x02,a[6]) ^ Mul(0x03,a[7]) ^ a[4] ^ a[5];    b[ 7] = Mul(0x02,a[7]) ^ Mul(0x03,a[4]) ^ a[5] ^ a[6];    b[ 8] = Mul(0x02,a[8]) ^ Mul(0x03,a[9]) ^ a[10] ^ a[11];    b[ 9] = Mul(0x02,a[9]) ^ Mul(0x03,a[10]) ^ a[11] ^ a[8];    b[10] = Mul(0x02,a[10]) ^ Mul(0x03,a[11]) ^ a[8] ^ a[9];    b[11] = Mul(0x02,a[11]) ^ Mul(0x03,a[8]) ^ a[9] ^ a[10];    b[12] = Mul(0x02,a[12]) ^ Mul(0x03,a[13]) ^ a[14] ^ a[15];    b[13] = Mul(0x02,a[13]) ^ Mul(0x03,a[14]) ^ a[15] ^ a[12];    b[14] = Mul(0x02,a[14]) ^ Mul(0x03,a[15]) ^ a[12] ^ a[13];    b[15] = Mul(0x02,a[15]) ^ Mul(0x03,a[12]) ^ a[13] ^ a[14];    for( int i = 0 ; i < 16 ; i ++ )        a[i] = b[i] ;      
}static void InvMixColumns(unsigned char \*a){    unsigned char b[16];    b[0] = Mul(0x0E, a[0]) ^ Mul(0x0B, a[1]) ^ Mul(0x0D, a[2]) ^ Mul(0x09, a[3]);    b[1] = Mul(0x09, a[0]) ^ Mul(0x0E, a[1]) ^ Mul(0x0B, a[2]) ^ Mul(0x0D, a[3]);    b[2] = Mul(0x0D, a[0]) ^ Mul(0x09, a[1]) ^ Mul(0x0E, a[2]) ^ Mul(0x0B, a[3]);    b[3] = Mul(0x0B, a[0]) ^ Mul(0x0D, a[1]) ^ Mul(0x09, a[2]) ^ Mul(0x0E, a[3]);    b[4] = Mul(0x0E, a[4]) ^ Mul(0x0B, a[5]) ^ Mul(0x0D, a[6]) ^ Mul(0x09, a[7]);    b[5] = Mul(0x09, a[4]) ^ Mul(0x0E, a[5]) ^ Mul(0x0B, a[6]) ^ Mul(0x0D, a[7]);    b[6] = Mul(0x0D, a[4]) ^ Mul(0x09, a[5]) ^ Mul(0x0E, a[6]) ^ Mul(0x0B, a[7]);    b[7] = Mul(0x0B, a[4]) ^ Mul(0x0D, a[5]) ^ Mul(0x09, a[6]) ^ Mul(0x0E, a[7]);    b[8] = Mul(0x0E, a[8]) ^ Mul(0x0B, a[9]) ^ Mul(0x0D, a[10]) ^ Mul(0x09, a[11]);    b[9] = Mul(0x09, a[8]) ^ Mul(0x0E, a[9]) ^ Mul(0x0B, a[10]) ^ Mul(0x0D, a[11]);    b[10] = Mul(0x0D, a[8]) ^ Mul(0x09, a[9]) ^ Mul(0x0E, a[10]) ^ Mul(0x0B, a[11]);    b[11] = Mul(0x0B, a[8]) ^ Mul(0x0D, a[9]) ^ Mul(0x09, a[10]) ^ Mul(0x0E, a[11]);    b[12] = Mul(0x0E, a[12]) ^ Mul(0x0B, a[13]) ^ Mul(0x0D, a[14]) ^ Mul(0x09, a[15]);    b[13] = Mul(0x09, a[12]) ^ Mul(0x0E, a[13]) ^ Mul(0x0B, a[14]) ^ Mul(0x0D, a[15]);    b[14] = Mul(0x0D, a[12]) ^ Mul(0x09, a[13]) ^ Mul(0x0E, a[14]) ^ Mul(0x0B, a[15]);    b[15] = Mul(0x0B, a[12]) ^ Mul(0x0D, a[13]) ^ Mul(0x09, a[14]) ^ Mul(0x0E, a[15]);    for (int i = 0; i < 16; i++)        a[i] = b[i];}void AddRoundKey( unsigned char \*a , unsigned char \*Key )  {  //  轮密钥加      for( int i = 0 ; i < 16 ; i ++ )        a[i] ^= Key[i] ;}void AesEncrypt(unsigned char \*blk, unsigned char \*key, int Nr){    int round;    AddRoundKey(blk,key);    for(round=1;round<=(Nr-1);round++)      
    {        SubBytes(blk);          
        ShiftRows(blk);          
        MixColumns(blk);      
        AddRoundKey(blk,key);    }    SubBytes(blk);    ShiftRows(blk);    AddRoundKey(blk,key);}void ScheduleKey(unsigned char \*inkey,unsigned char \*outkey, int Nk, int Nr){    unsigned char temp[4],t;    int m,n;      
    for(n=0;n<(4\*Nk);n++){        outkey[n]=inkey[n];    }    n=Nk;      
    unsigned char RC[10];    RC[0]=1;    for(int i=1;i<10;i++){        RC[i]=Mul(0x02,RC[i-1]);    }      
    while(n<(4\*(Nr+1)))   
    {        for(m=0;m<4;m++)   
            temp[m]=outkey[(4\*(n-1))+m];      
        if(n%Nk==0)        {            t=temp[0];temp[0]=temp[1];temp[1]=temp[2];temp[2]=temp[3];temp[3]=t;            for(m=0;m<4;m++){                temp[m]=S[temp[m]];            }            temp[0]^=RC[(n/Nk)-1];        }        for(m=0;m<4;m++){            outkey[(4\*n)+m]=outkey[(4\*(n-Nk))+m]^temp[m];        }        ++n;    }}void AesDecrypt(unsigned char \*blk, unsigned char \*key, int Nr)//解密方法一 替换步骤顺序和加密不一样{    AddRoundKey(blk, key);    for (int round = Nr - 1; round > 0; round--) {        InvShiftRows(blk);        InvSubBytes(blk);        AddRoundKey(blk, key);        InvMixColumns(blk);    }    InvShiftRows(blk);    InvSubBytes(blk);    AddRoundKey(blk, key);}int main(void){    unsigned char pt[16], key[17];    unsigned char skey[15 \* 16];    int i= 0;    printf("请输入16字节明文：");    scanf("%s", pt);//    getchar();    printf("明文为（128比特）：");    for (i = 0; i < 16; i++) {        printf("%02x", pt[i]);    }    printf("\n");    printf("请输入16字节密钥：");    scanf("%s", key);//    getchar();    ScheduleKey(key, skey, 4, 10);    AesEncrypt(pt, skey, 10);    printf("加密后密文为：");    for (i = 0; i < 16; i++) {        printf("%02x", pt[i]);    }    printf("\n");    AesDecrypt(pt, skey, 10);    printf("解密后明文为：");    for (i = 0; i < 16; i++) {        printf("%c", pt[i]);    }      
    printf("\n解密后明文为（128比特）：");    for (i = 0; i < 16; i++) {        printf("%02x", pt[i]);    }  
    return 0;}

四、实验结果与分析

测试结果截图与分析如下：

图形用户界面, 应用程序

描述已自动生成、

先调用Mul函数运行测试用例，来验证Mul函数正确性。结果与PPT所提到的一致，证明Mul函数正确。

文本

描述已自动生成

用字符串接收明文，将明文输出16进制bit串便于后续比较。再用字符串的形式接收密钥。用16进制输出被处理后的密文比特串。再运行解密程序。同时输出解密后明文的字符串形式和16进制比特串形式，比较处理前后明文，发现一致。

五、基于OBE模式的学生自我评价与体会

（该实验对自身分析、设计、思辨、创新等个人综合能力与素质的影响）

本次实验，通过C语言编程实现AES加密功能。通过本次实验我对AES算法有了更深刻的认识。在实验过程中，需要设计模块和函数来实现AES算法的各个步骤，如密钥扩展、字节替代、行移位和列混淆等，这要求对AES算法有充分的了解，通过本次实验我发现了自己对AES掌握上还有所欠缺。总的来说，不仅练习了我的编程能力，还让我加深了对AES算法的理解，此外还让我知道自己的不足之处。