密码学基础实验报告

東北大學

实验名称	DES 的编程实现
班　　级	软信 2201
学　　号	20226694
姓　　名	魏董帅
日　　期	2024 年 6 月 19 日
成　　绩	
评 阅 人	

软件学院

一、 实验目的与意义

理解加密算法原理:

通过编程实现 DES 算法,可以深入了解其加密原理和工作机制,包括初始置换、16 轮 Feistel 网络结构、S 盒和 P 盒等关键步骤。这有助于加深对对称加密算法的理解。

实践加密技术:

编程实现 DES 可以将理论知识转化为实践操作,培养学生的编程能力和解决实际问题的能力。这对于未来从事密码学或信息安全相关工作很有帮助。

性能分析和优化:

通过测试和分析 DES 算法的运行效率,可以探讨算法的性能瓶颈,并尝试进行优化,如采用并行计算、查找表等方式提高速度。这有助于理解算法实现的细节和权衡。

二、 实验环境

操作系统：Windows 操作系统
调试软件：Code::Blocks
版本号：20.03
上机地点：信息 B405

三、 实验的预习内容

对称加密算法：了解对称加密算法的基本原理，包括加密和解密过程、密钥管理等方面的知识。
DES 算法原理：深入了解 DES 算法的工作原理，包括 Feistel 结构、轮函数、S 盒、P 置换等基本概念。
密钥生成：学习 DES 算法中密钥生成的过程，包括从 64 位密钥中生成 16 个子密钥的方法。
加密和解密过程：了解 DES 算法中的加密和解密过程，包括初始置换、轮函数的应用、逆初始置换等步骤。

四、 实验的步骤与调试方法

确定实验目标：明确你的实验目标，例如实现 DES 算法的加密和解密功能。

学习算法原理：深入了解 DES 算法的原理和流程，包括 Feistel 结构、轮函数、S 盒、P 置换等。

编写代码：使用你选择的编程语言，根据 DES 算法的原理编写加密和解密的代码。

调试代码：测试你的代码，确保加密和解密的功能正常。如果遇到问题，可以使用以下调试方法。

确定实验目标：明确你的实验目标，例如实现 DES 算法的加密和解密功能。
学习算法原理：深入了解 DES 算法的原理和流程，包括 Feistel 结构、轮函数、S 盒、P 置换等。
编写代码：使用你选择的编程语言，根据 DES 算法的原理编写加密和解密的代码。
调试代码：测试你的代码，确保加密和解密的功能正常。如果遇到问题，可以使

用以下调试方法。

五、 实验数据与实验结果

需要加密的文本：HELLOWORD
加密后的文本：8A8B5C3D9FC6F1A
解密后的文本：HELLOWORD

六、 实验用程序清单（要有注释）

```
#include "stdio.h"
#include "memory.h"
#include "time.h"
#include "stdlib.h"

#define PLAIN_FILE_OPEN_ERROR -1
#define KEY_FILE_OPEN_ERROR -2
#define CIPHER_FILE_OPEN_ERROR -3
#define OK 1


typedef char ElemType;


 /*初始置换表 IP*/
int IP_Table[64] = {  57,49,41,33,25,17,9,1,
59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5,
63,55,47,39,31,23,15,7,
56,48,40,32,24,16,8,0,
58,50,42,34,26,18,10,2,
60,52,44,36,28,20,12,4,
62,54,46,38,30,22,14,6};
/*逆初始置换表 IP^-1*/
int IP_1_Table[64] = {39,7,47,15,55,23,63,31,
  38,6,46,14,54,22,62,30,
  37,5,45,13,53,21,61,29,
  36,4,44,12,52,20,60,28,
  35,3,43,11,51,19,59,27,
  34,2,42,10,50,18,58,26,
  33,1,41,9,49,17,57,25,
  32,0,40,8,48,16,56,24};

/*扩充置换表 E*/
int E_Table[48] = {31, 0, 1, 2, 3, 4,
```

```
3,  4,  5,  6,  7,  8,
7,  8,9,10,11,12,
11,12,13,14,15,16,
15,16,17,18,19,20,
19,20,21,22,23,24,
23,24,25,26,27,28,
27,28,29,30,31,  0};

/*置换函数 P*/
int P_Table[32] = {15,6,19,20,28,11,27,16,
0,14,22,25,4,17,30,9,
1,7,23,13,31,26,2,8,
18,12,29,5,21,10,3,24};

/*S 盒*/
int S[8][4][16] =
/*S1*/
{{{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7},
  {0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8},
 {4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0},
 {15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13}},
 /*S2*/
  {{15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10},
  {3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5},
  {0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15},
  {13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9}},
  /*S3*/
  {{10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8},
  {13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1},
 {13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7},
  {1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12}},
  /*S4*/
  {{7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15},
  {13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9},
  {10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4},
  {3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14}},
  /*S5*/
  {{2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9},
  {14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6},
  {4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14},
  {11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3}},
  /*S6*/
  {{12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11},
  {10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8},
```

```
  {9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6},
  {4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13}},
  /*S7*/
  {{4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1},
  {13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6},
  {1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2},
  {6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12}},
  /*S8*/
  {{13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7},
  {1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2},
  {7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8},
  {2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}}};
/*置换选择 1*/
int PC_1[56] = {56,48,40,32,24,16,8,
  0,57,49,41,33,25,17,
  9,1,58,50,42,34,26,
  18,10,2,59,51,43,35,
  62,54,46,38,30,22,14,
  6,61,53,45,37,29,21,
  13,5,60,52,44,36,28,
  20,12,4,27,19,11,3};

/*置换选择 2*/
int PC_2[48] = {13,16,10,23,0,4,2,27,
  14,5,20,9,22,18,11,3,
  25,7,15,6,26,19,12,1,
  40,51,30,36,46,54,29,39,
  50,44,32,46,43,48,38,55,
  33,52,45,41,49,35,28,31};

/*对左移次数的规定*/
int MOVE_TIMES[16] = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};



int ByteToBit(ElemType ch,ElemType bit[8]);
int BitToByte(ElemType bit[8],ElemType *ch);
int Char8ToBit64(ElemType ch[8],ElemType bit[64]);
int Bit64ToChar8(ElemType bit[64],ElemType ch[8]);
int DES_MakeSubKeys(ElemType key[64],ElemType subKeys[16][48]);
int DES_PC1_Transform(ElemType key[64], ElemType tempbts[56]);
int DES_PC2_Transform(ElemType key[56], ElemType tempbts[48]);
int DES_ROL(ElemType data[56], int time);
int DES_IP_Transform(ElemType data[64]);
int DES_IP_1_Transform(ElemType data[64]);
```

```
int DES_E_Transform(ElemType data[48]);
int DES_P_Transform(ElemType data[32]);
int DES_SBOX(ElemType data[48]);
int DES_XOR(ElemType R[48], ElemType L[48],int count);
int DES_Swap(ElemType left[32],ElemType right[32]);
int DES_EncryptBlock(ElemType plainBlock[8], ElemType subKeys[16][48],
ElemType cipherBlock[8]);
int DES_DecryptBlock(ElemType cipherBlock[8], ElemType subKeys[16][48],
ElemType plainBlock[8]);
int DES_Encrypt(char *plainFile, char *keyStr,char *cipherFile);
int DES_Decrypt(char *cipherFile, char *keyStr,char *plainFile);


/*字节转换成二进制*/
int ByteToBit(ElemType ch, ElemType bit[8]){
 int cnt;
 for(cnt = 0;cnt < 8; cnt++){
  *(bit+cnt) = (ch>>cnt)&1;
 }
 return 0;
}

/*二进制转换成字节*/
int BitToByte(ElemType bit[8],ElemType *ch){
 int cnt;
 for(cnt = 0;cnt < 8; cnt++){
  *ch |= *(bit + cnt)<<cnt;
 }
 return 0;
}

/*将长度为 8 的字符串转为二进制位串*/
int Char8ToBit64(ElemType ch[8],ElemType bit[64]){
 int cnt;
 for(cnt = 0; cnt < 8; cnt++){
  ByteToBit(*(ch+cnt),bit+(cnt<<3));
 }
 return 0;
}

/*将二进制位串转为长度为 8 的字符串*/
int Bit64ToChar8(ElemType bit[64],ElemType ch[8]){
 int cnt;
 memset(ch,0,8);
```

```
 for(cnt = 0; cnt < 8; cnt++){
  BitToByte(bit+(cnt<<3),ch+cnt);
 }
 return 0;
}

/*生成子密钥*/
int DES_MakeSubKeys(ElemType key[64],ElemType subKeys[16][48]){
 ElemType temp[56];
 int cnt;
 DES_PC1_Transform(key,temp);/*PC1 置换*/
 for(cnt = 0; cnt < 16; cnt++){/*16 轮跌代，产生 16 个子密钥*/
  DES_ROL(temp,MOVE_TIMES[cnt]);/*循环左移*/
  DES_PC2_Transform(temp,subKeys[cnt]);/*PC2 置换，产生子密钥*/
 }
 return 0;
}

/*密钥置换 1*/
int DES_PC1_Transform(ElemType key[64], ElemType tempbts[56]){
 int cnt;
 for(cnt = 0; cnt < 56; cnt++){
  tempbts[cnt] = key[PC_1[cnt]];
 }
 return 0;
}

/*密钥置换 2*/
int DES_PC2_Transform(ElemType key[56], ElemType tempbts[48]){
 int cnt;
 for(cnt = 0; cnt < 48; cnt++){
  tempbts[cnt] = key[PC_2[cnt]];
 }
 return 0;
}

/*循环左移*/
int DES_ROL(ElemType data[56], int time){
 ElemType temp[56];

 /*保存将要循环移动到右边的位*/
 memcpy(temp,data,time);
 memcpy(temp+time,data+28,time);
```

```
 /*前 28 位移动*/
 memcpy(data,data+time,28-time);
 memcpy(data+28-time,temp,time);

 /*后 28 位移动*/
 memcpy(data+28,data+28+time,28-time);
 memcpy(data+56-time,temp+time,time);

 return 0;
}

/*IP 置换*/
int DES_IP_Transform(ElemType data[64]){
 int cnt;
 ElemType temp[64];
 for(cnt = 0; cnt < 64; cnt++){
  temp[cnt] = data[IP_Table[cnt]];
 }
 memcpy(data,temp,64);
 return 0;
}

/*IP 逆置换*/
int DES_IP_1_Transform(ElemType data[64]){
 int cnt;
 ElemType temp[64];
 for(cnt = 0; cnt < 64; cnt++){
  temp[cnt] = data[IP_1_Table[cnt]];
 }
 memcpy(data,temp,64);
 return 0;
}

/*扩展置换*/
int DES_E_Transform(ElemType data[48]){
 int cnt;
 ElemType temp[48];
 for(cnt = 0; cnt < 48; cnt++){
  temp[cnt] = data[E_Table[cnt]];
 }
 memcpy(data,temp,48);
 return 0;
}
```

```
/*P 置换*/
int DES_P_Transform(ElemType data[32]){
 int cnt;
 ElemType temp[32];
 for(cnt = 0; cnt < 32; cnt++){
  temp[cnt] = data[P_Table[cnt]];
 }
 memcpy(data,temp,32);
 return 0;
}

/*异或*/
int DES_XOR(ElemType R[48], ElemType L[48] ,int count){
 int cnt;
 for(cnt = 0; cnt < count; cnt++){
  R[cnt] ^= L[cnt];
 }
 return 0;
}

/*S 盒置换*/
int DES_SBOX(ElemType data[48]){
 int cnt;
 int line,row,output;
 int cur1,cur2;
 for(cnt = 0; cnt < 8; cnt++){
  cur1 = cnt*6;
  cur2 = cnt<<2;

  /*计算在 S 盒中的行与列*/
  line = (data[cur1]<<1) + data[cur1+5];
  row = (data[cur1+1]<<3) + (data[cur1+2]<<2)
+ (data[cur1+3]<<1) + data[cur1+4];
  output = S[cnt][line][row];

  /*化为 2 进制*/
  data[cur2] = (output&0X08)>>3;
  data[cur2+1] = (output&0X04)>>2;
  data[cur2+2] = (output&0X02)>>1;
  data[cur2+3] = output&0x01;
 }
 return 0;
}
```

```
/*交换*/
int DES_Swap(ElemType left[32], ElemType right[32]){
 ElemType temp[32];
 memcpy(temp,left,32);
 memcpy(left,right,32);
 memcpy(right,temp,32);
 return 0;
}

/*加密单个分组*/
int DES_EncryptBlock(ElemType plainBlock[8], ElemType subKeys[16][48],
ElemType cipherBlock[8]){
 ElemType plainBits[64];
 ElemType copyRight[48];
 int cnt;

 Char8ToBit64(plainBlock,plainBits);
 /*初始置换（IP 置换）*/
 DES_IP_Transform(plainBits);

 /*16 轮迭代*/
 for(cnt = 0; cnt < 16; cnt++){
  memcpy(copyRight,plainBits+32,32);
  /*将右半部分进行扩展置换，从 32 位扩展到 48 位*/
  DES_E_Transform(copyRight);
  /*将右半部分与子密钥进行异或操作*/
  DES_XOR(copyRight,subKeys[cnt],48);
  /*异或结果进入 S 盒，输出 32 位结果*/
  DES_SBOX(copyRight);
  /*P 置换*/
  DES_P_Transform(copyRight);
  /*将明文左半部分与右半部分进行异或*/
  DES_XOR(plainBits,copyRight,32);
  if(cnt != 15){
/*最终完成左右部的交换*/
DES_Swap(plainBits,plainBits+32);
  }
 }
 /*逆初始置换（IP^1 置换）*/
 DES_IP_1_Transform(plainBits);
 Bit64ToChar8(plainBits,cipherBlock);
 return 0;
}
```

```
/*解密单个分组*/
int DES_DecryptBlock(ElemType cipherBlock[8], ElemType
subKeys[16][48],ElemType plainBlock[8]){
 ElemType cipherBits[64];
 ElemType copyRight[48];
 int cnt;

 Char8ToBit64(cipherBlock,cipherBits);
 /*初始置换（IP 置换）*/
 DES_IP_Transform(cipherBits);

 /*16 轮迭代*/
 for(cnt = 15; cnt >= 0; cnt--){
  memcpy(copyRight,cipherBits+32,32);
  /*将右半部分进行扩展置换，从 32 位扩展到 48 位*/
  DES_E_Transform(copyRight);
  /*将右半部分与子密钥进行异或操作*/
  DES_XOR(copyRight,subKeys[cnt],48);
  /*异或结果进入 S 盒，输出 32 位结果*/
  DES_SBOX(copyRight);
  /*P 置换*/
  DES_P_Transform(copyRight);
  /*将明文左半部分与右半部分进行异或*/
  DES_XOR(cipherBits,copyRight,32);
  if(cnt != 0){
/*最终完成左右部的交换*/
DES_Swap(cipherBits,cipherBits+32);
  }
 }
 /*逆初始置换（IP^1 置换）*/
 DES_IP_1_Transform(cipherBits);
 Bit64ToChar8(cipherBits,plainBlock);
 return 0;
}

/*加密文件*/
int DES_Encrypt(char *plainFile, char *keyStr,char *cipherFile){
 FILE *plain,*cipher;
 int count;
 ElemType plainBlock[8],cipherBlock[8],keyBlock[8];
 ElemType bKey[64];
 ElemType subKeys[16][48];
 if((plain = fopen(plainFile,"rb")) == NULL){
  return PLAIN_FILE_OPEN_ERROR;
```

```
 }
 if((cipher = fopen(cipherFile,"wb")) == NULL){
  return CIPHER_FILE_OPEN_ERROR;
 }
 /*设置密钥*/
 memcpy(keyBlock,keyStr,8);
 /*将密钥转换为二进制流*/
 Char8ToBit64(keyBlock,bKey);
 /*生成子密钥*/
 DES_MakeSubKeys(bKey,subKeys);

 while(!feof(plain)){
  /*每次读 8 个字节，并返回成功读取的字节数*/
  if((count = fread(plainBlock,sizeof(char),8,plain)) == 8){
DES_EncryptBlock(plainBlock,subKeys,cipherBlock);
fwrite(cipherBlock,sizeof(char),8,cipher);
  }
 }
 if(count){
  /*填充*/
  memset(plainBlock + count,'\0',7 - count);
  /*最后一个字符保存包括最后一个字符在内的所填充的字符数量*/
  plainBlock[7] = 8 - count;
  DES_EncryptBlock(plainBlock,subKeys,cipherBlock);
  fwrite(cipherBlock,sizeof(char),8,cipher);
 }
 fclose(plain);
 fclose(cipher);
 return OK;
}

/*解密文件*/
int DES_Decrypt(char *cipherFile, char *keyStr,char *plainFile){
 FILE *plain, *cipher;
 int count,times = 0;
 long fileLen;
 ElemType plainBlock[8],cipherBlock[8],keyBlock[8];
 ElemType bKey[64];
 ElemType subKeys[16][48];
 if((cipher = fopen(cipherFile,"rb")) == NULL){
  return CIPHER_FILE_OPEN_ERROR;
 }
 if((plain = fopen(plainFile,"wb")) == NULL){
  return PLAIN_FILE_OPEN_ERROR;
```

```
 }

 /*设置密钥*/
 memcpy(keyBlock,keyStr,8);
 /*将密钥转换为二进制流*/
 Char8ToBit64(keyBlock,bKey);
 /*生成子密钥*/
 DES_MakeSubKeys(bKey,subKeys);

 /*取文件长度 */
 fseek(cipher,0,SEEK_END);/*将文件指针置尾*/
 fileLen = ftell(cipher); /*取文件指针当前位置*/
 rewind(cipher); /*将文件指针重指向文件头*/
 while(1){
  /*密文的字节数一定是 8 的整数倍*/
  fread(cipherBlock,sizeof(char),8,cipher);
  DES_DecryptBlock(cipherBlock,subKeys,plainBlock);
  times += 8;
  if(times < fileLen){
fwrite(plainBlock,sizeof(char),8,plain);
  }
  else{
break;
  }
 }
 /*判断末尾是否被填充*/
 if(plainBlock[7] < 8){
  for(count = 8 - plainBlock[7]; count < 7; count++){
if(plainBlock[count] != '\0'){
 break;
}
  }
 }
 if(count == 7){/*有填充*/
  fwrite(plainBlock,sizeof(char),8 - plainBlock[7],plain);
 }
 else{/*无填充*/
  fwrite(plainBlock,sizeof(char),8,plain);
 }

 fclose(plain);
 fclose(cipher);
 return OK;
}
```

```
int main()
{
 clock_t a,b;
 a = clock();
 DES_Encrypt("1.txt","key.txt","2.txt");
 b = clock();
 printf("加密（请按回车）");
 getchar();


 system("pause");
 a = clock();
 DES_Decrypt("2.txt","key.txt","3.txt");
 b = clock();
 printf("解密\n");
 getchar();
 return 0;
}
```

七、 思考题（必需回答）

1. DES 的原理是什么？
   初始置换：对输入的 64 位明文进行初始置换，打乱明文的顺序。
   加密轮：DES 算法使用 16 个加密轮来对数据进行处理。每个加密轮包括以下步骤：
   a. 将 64 位明文分成左右两部分，每部分 32 位。
   b. 将右半部分作为输入，经过扩展置换得到 48 位的数据。
   c. 将 48 位数据与轮密钥进行异或操作。
   d. 将异或结果分成 8 个 6 位的数据块，每个数据块作为 S 盒的输入，经过 S 盒替换得到 32 位数据。
   e. 经过置换操作，将左半部分和 S 盒输出结果进行异或操作，得到新的右半部分。
   f. 将原来的右半部分作为新的左半部分。
   最终置换：经过 16 个加密轮之后，将左右两部分进行交换，并进行最终的置换操作，得到 64 位的密文。
2. DES 使用多少位密钥？
   56 位，尽管实际上密钥是 64 位的，但是每个第 8、16、24、32、40、48、56 位是校验位，因此实际用于加密和解密的位数是 56 位。
3. DES 对明文分块的单位是多少？
   DES 算法对明文分块的单位是 64 位，即 8 个字节。这意味着明文被分成 64 位的块，然后分别进行加密处理。如果明文长度不是 64 位的倍数，通常会使用填充的方式将其填充到 64 位的倍数。
4. DES 对每一个数据块加密的轮次是多少？
   DES 算法对每一个 64 位的数据块进行 16 轮加密。
5. 简单描述 EBox 的操作过程。

输入是 32 位的数据块，这些数据位被标记为 32 个位，从左到右编号为 1 到 32。
E 盒中包含 48 个输出位，被标记为 1 到 48。
E 盒的作用是将输入的 32 位数据进行扩展和置换，生成 48 位的输出数据。
扩展的过程是通过复制和置换来实现的。具体地，输入的 32 位数据中的某些位会被复制到输出的 48 位中，并且经过一定的置换规则。
扩展后的 48 位数据会作为轮函数中的一部分，与轮子密钥进行异或运算。

6. 简单描述 SBox 的操作过程。
输入是 48 位的数据块，这些数据位被分成 8 组，每组 6 位。
DES 算法中有 8 个不同的 S 盒，每个 S 盒都有自己的置换规则。
每组 6 位数据作为 S 盒的输入，通过 S 盒的置换规则，被映射为 4 位的输出数据。
S 盒的置换规则是固定的，通过查表的方式进行，将每组 6 位数据映射为 4 位输出数据。
最终将 8 个 S 盒的输出数据合并为 32 位输出数据。

7. 简单描述 PBox 的操作过程。
输入是 32 位的数据块，这些数据位被标记为 32 个位，从左到右编号为 1 到 32。
P 盒中包含 32 个输出位，被标记为 1 到 32。
P 盒的作用是对输入的 32 位数据进行置换，生成 32 位的输出数据。
置换的过程是通过重排输入数据位来实现的。具体地，输入的 32 位数据中的某些位会被重新排列，生成输出的 32 位数据。
P 盒的置换规则是固定的，是 DES 算法中的一个重要部分。
P 盒的作用是增加加密算法的复杂性，使得加密后的数据更难以被破解。

八、 结束语

通过本次实验，我深入了解了 DES 算法的原理和实现过程，掌握了对称加密算法的基本知识，提高了编程实现和调试的能力。

自己的认识：DES 算法作为一种经典的对称加密算法，其原理复杂而且涉及到许多细节，通过实际编程实现，我对其原理有了更深入的理解，也认识到了加密算法在信息安全领域的重要性。

九、 参考文献

1.Richard J.Spillman:《CLASSICAL AND CONTEMPORARY CRYPTOLOGY》, 清华大学出版社, 2005-7.
2.William Stallings. Cryptography and Network Security: Principles and Practice. 8th ed. Prentice Hall, 2020.
3. Ling Dong, Kefei Chen. Cryptographic protocol. Security analysis based on trusted freshness. 高等教育出版社, 2011-10.
4. 胡向东.《应用密码学：第 4 版》, 电子工业出版社, 2019-6.

实验成绩

考查内容	分数	得分
做好实验内容的预习，写出预习报告	10	
了解实验题目的调试方法	10	
按实验要求预先设计好程序	10	
认真记录实验数据并分析实验结果	10	
实验后按要求书写实验报告，记录实验用数据及运行结果	30	
创新能力强，在实验中设计的程序有一定的通用性，算法优化	20	
实验过程中，具有严谨的学习态度，认真、踏实、一丝不苟的科学作风	10	