 ****

**实验报告二**

学生姓名： 龚开宸 学 号：6103120034 专业班级： 20级计科1班

实验类型：□验证 ☑ 综合 □设计□创新 实验日期： 2022.3.123 实验成绩：

1. **实验项目名称**

DES算法的实现。

1. **实验目的**

分组密码体制是在密钥控制下一次变换一个明文数据块（又称为分组）的密码体制。熟悉分组加密算法的原理，了解SPN及Feistel结构。完成DES算法加密及解密过程。

1. **实验基本原理**

在本实验中，DES加密过程中的明文、密钥均为英文字母或汉字构成的信息，比如中英文文章，加密后的密文要从二进制信息用相关编码方法进行解码。

DES 加密原理：

1．概述：

DES算法是一种分组加密算法。它将明文分组为64位，并使用64位密钥位明文进行加密。

在本例程中，均采用一位数组的方式存储明文和密钥。

<code>

1. **public** **class** Message **implements** mess{
2. **final** **char** COMPLEMENT = ' ';
3. // store all info
4. **char**[] data;
5. // store 8 character
6. **char**[] buf;
7. // store bitwise info
8. **private** **byte** [] bitM;
9. // mark location
10. **int** idx;
11. **int** surplus;
12. **int** round;
14. }
16. **public** **class** Key **extends** Message {
18. **private** **byte**[] bitM;
20. **public** Key() {
21. **super**();
22. bitM = **new** **byte**[64];
23. }
24. **public** Key(**char**[] t) {
25. **super**(t);
26. bitM = **new** **byte**[64];
27. }
29. }

</code>

2. 算法过程详解

1.首先对明文进行初始IP置换

1. <code> // 对 size大小的 bits矩阵（实际是一维数组）进行 对应的置换
2. **public** **static** **byte**[] permute(**byte**[] bits, **int** size\_t, **int**[] permuteMatrix) {
3. **int** size = permuteMatrix.length;
4. **byte**[] copy = **new** **byte**[size\_t];
5. System.arraycopy(bits, 0, copy, 0, size\_t);
7. bits = **new** **byte**[size];
8. **for**( **int** i = 0 ; i < size ; i ++) {
9. bits[i] = copy[permuteMatrix[i]-1];
10. }
11. **return** bits;
12. }
14. **static** **final** **int**[] IP = **new** **int**[]{
15. 58, 50, 42, 34, 26, 18, 10, 2,
16. 60, 52, 44, 36, 28, 20, 12, 4,
17. 62, 54, 46, 38, 30, 22, 14, 6,
18. 64, 56, 48, 40, 32, 24, 16, 8,
19. 57, 49, 41, 33, 25, 17, 9, 1,
20. 59, 51, 43, 35, 27, 19, 11, 3,
21. 61, 53, 45, 37, 29, 21, 13, 5,
22. 63, 55, 47, 39, 31, 23, 15, 7,
23. };

</code>

2. 根据输入的密钥生成16轮加密的密钥

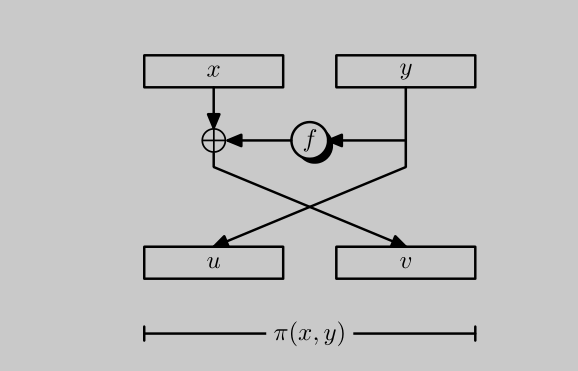
1. // 对密匙进行左移
2. **public** **static** **void** LeftShift(Key k, **int** shift) {
3. **byte**[] m = k.getBitM();
4. **if**( shift == 1 ) {
5. **byte** a0 = m[0], a28 = m[28];
6. System.arraycopy(m, 1, m, 0,27);
7. m[27] = a0;
8. System.arraycopy(m,29,m,28,27);
9. m[55] = a28;
10. }
11. **if**( shift == 2 ) {
12. **byte** a0 = m[0], a1 = m[1];
13. **byte** a28 = m[28], a29 = m[29];
14. System.arraycopy(m, 2, m, 0,26);
15. m[27] = a1; m[26] = a0;
16. System.arraycopy(m,30,m,28,26);
17. m[55] = a29; m[54] = a28;
18. }
19. }
21. // 第round的密匙生成函数
22. **public** **static** **void** generateKey(Key k) {
24. **for**( **int** i = 0; i < 16 ; i++ ) {
25. **int** shift = Data.DesRotations[i];
26. LeftShift(k, shift);
27. Data.deskey[i] = permute(k.getBitM(), k.getBitM().length, Data.PC2);
28. }
29. }

3. 进入Feistel结构进行16轮加密

① 将明文的低32位存入LPT数组(left plain text), 将明文的高32位存入RPT数组。

② 每一轮加密的LPT(低32位)作为 下一轮加密的 RPT(高32位)

每一轮加密的RPT 后作为下一轮加密的LPT.



Feistel 轮结构示意图

1. **public** **static** **void** encrypt(mess m, Key k, **boolean** encrypt) {
2. DesCrypt.generateKey(k);
3. m.setBitM(permute(m.getBitM(), m.getBitM().length, Data.IP));
5. **byte**[] LPT\_old = **new** **byte**[32];
6. **byte**[] RPT\_old = **new** **byte**[32];
7. **byte**[] desKey;
8. System.arraycopy(m.getBitM(), 0, LPT\_old, 0, 32);
9. System.arraycopy(m.getBitM(), 32, RPT\_old, 0, 32);
11. **for** ( **int** i = 0 ; i < 16 ; i++ ) {
12. **if**( encrypt )
13. desKey = Data.deskey[i];
14. **else**
15. desKey = Data.deskey[15-i];
17. **byte**[] LPT\_new = Arrays.copyOf(RPT\_old, RPT\_old.length);
18. **byte**[] RPT\_new = XOR( LPT\_old, F(RPT\_old, desKey));
19. LPT\_old = LPT\_new;
20. RPT\_old = RPT\_new;

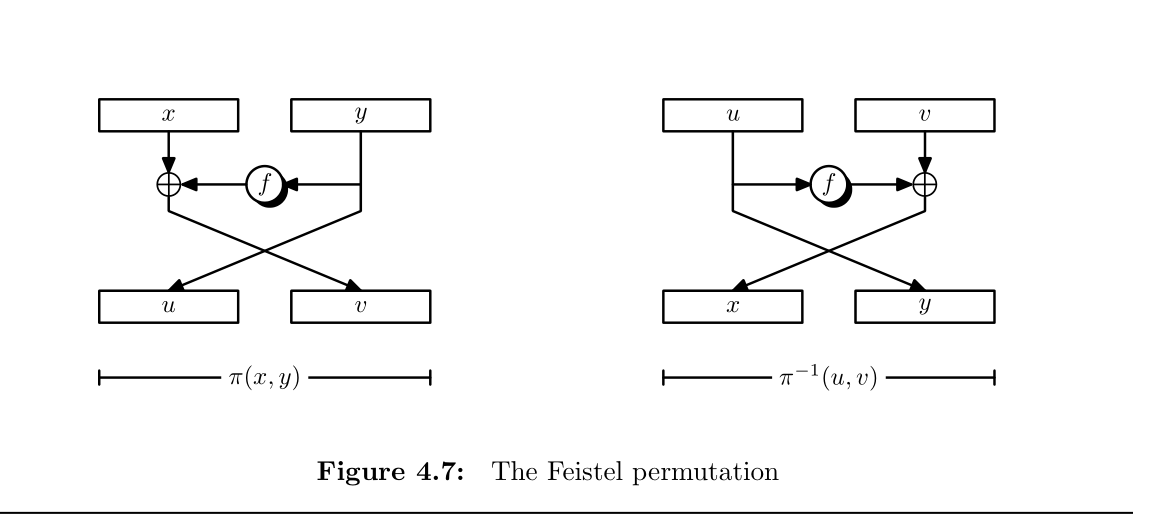
23. }
24. m.setBitM(Merge(RPT\_old, LPT\_old));
25. m.setBitM(permute(m.getBitM(), m.getBitM().length, Data.IPReverse));
26. }

4. 加密完成后，对调左32位和右32位，再进行一次IP逆置换

1. // 加密完成 将 合并 左32位 和右32位
2. **public** **static** **byte**[] Merge( **byte**[] a, **byte**[] b) {
3. **byte** []t = **new** **byte**[64];
4. System.arraycopy(a, 0, t, 0, 32);
5. System.arraycopy(b, 0, t, 32, 32);
6. **return** t;
7. }
8. **static** **final** **int**[] IPReverse = **new** **int**[]{
9. 40, 8, 48, 16, 56, 24, 64, 32,
10. 39, 7, 47, 15, 55, 23, 63, 31,
11. 38, 6, 46, 14, 54, 22, 62, 30,
12. 37, 5, 45, 13, 53, 21, 61, 29,
13. 36, 4, 44, 12, 52, 20, 60, 28,
14. 35, 3, 43, 11, 51, 19, 59, 27,
15. 34, 2, 42, 10, 50, 18, 58, 26,
16. 33, 1, 41, 9, 49, 17, 57, 25
17. };

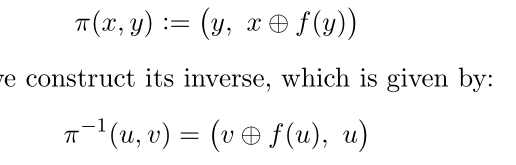
5. 加密结束

6．解密过程和加密过程几乎一样，只是轮密钥使用的顺序与加密使用的正好相反。



3. 加密原理

Feistel轮结构的可逆性



4. DES 介绍

DES加密算法是16轮的feistel型密码，每一轮都采用一个不同的函数

.假设在 第i轮，这个函数定义如下

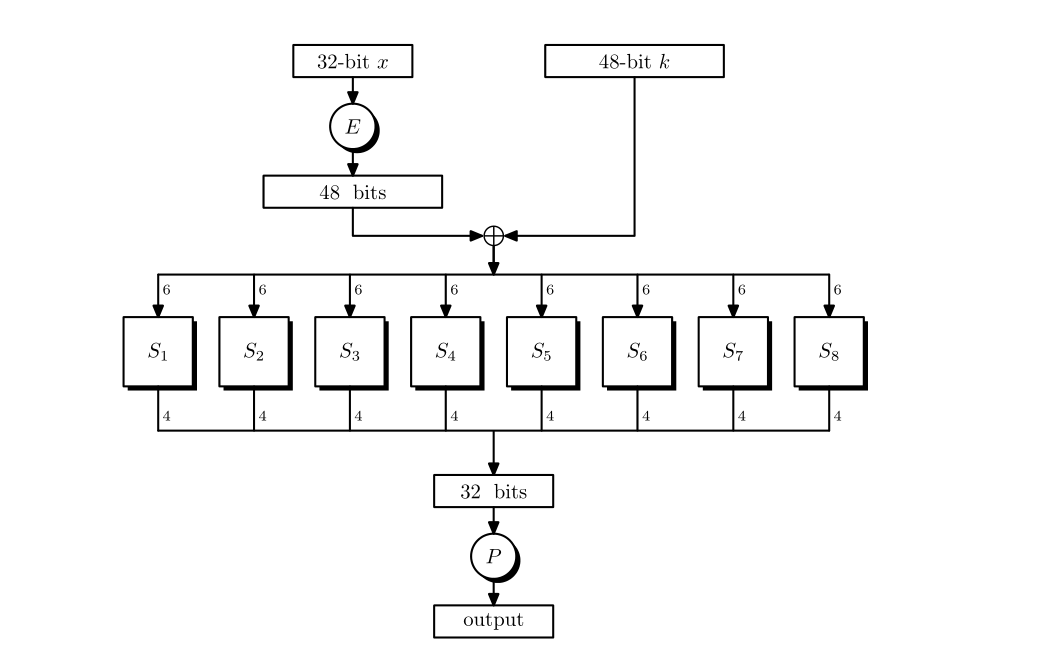


函数是一个固定的函数, 它接受56位密钥和32位的明文。它的实现如下

* 1. 通过E扩展，将32位密文扩展为48位二进制数
  2. 通过PC2置换，将56位密文 缩短为 48位二进制数
  3. 将a,b的输出结构进行异或，得到48位二进制数
  4. 令c得到的48位二进制数 经行 s盒置换，得到32位二进制数
  5. 最后令d中得到32位二进制数进行p置换，以扩散s盒的效应
  6. 输出e得到的32位二进制数

5. S盒介绍

S盒将48位输出分为6组，对于每一组，去最低位和最高位得到行值，取中间四位得到列值，在根据行列值在S盒对应的表中找到对应的16进制数，将其转换为4位的二进制，这样就实现的 6 –> 4 位的变换。



1. <code>// 返回两个整形数组（大小相等）的异或结果
2. **public** **static** **byte**[] XOR(**byte**[] m, **byte**[] DesKey) {
3. **if**( m.length != DesKey.length ) {
4. System.out.println(m.length + ' ' + DesKey.length);
5. **return** **null**;
6. }
7. **int** size = m.length;
8. **byte**[] res = **new** **byte**[size];
9. **for**( **int** i = 0 ; i < size ; i ++ ) {
10. res[i] = (**byte**)(m[i]^DesKey[i]);
11. }
12. **return** res;
13. }
14. // 将48位分成 8 个 6位
15. **public** **static** **int** ToBinary(**byte**[] RPT, **int** round) {
16. **int** sixBit = 0;
17. **for**( **int** i = round\*6 ; i < round\*6 + 6 ; i ++ ) {
18. sixBit = sixBit << 1 | RPT[i];
19. }
20. **return** sixBit;
21. }
22. // 将s盒读取的输入转化为 4位2 进制表示
23. **public** **static** **void** SBoxConvert(**byte**[] RPT\_32, **int** num, **int** round) {
24. **for**( **int** i = 3, j = 4\*round ; i >= 0 ; i --, j++ ) {
25. RPT\_32[j] = (**byte**)((num >> i) & 1);
26. }
27. }
28. // s盒 将48位 -> 32位
29. **public** **static** **byte**[] SBox (**byte**[] RPT) {
31. **byte** []RPT\_32 = **new** **byte**[32];
32. **for**( **int** i = 0 ; i < 8 ; i ++ ) {
33. **int** sixBit = ToBinary(RPT, i);
34. SBoxConvert(RPT\_32, Data.S[i][sixBit], i);
35. }
36. **return** RPT\_32;
37. }
39. // f函数
40. **public** **static** **byte**[] F(**byte**[] RPT, **byte**[] DesKey ) {
42. RPT = permute(RPT, RPT.length, Data.E);
43. RPT = XOR(RPT, DesKey);
44. RPT = SBox(RPT);
45. RPT = permute(RPT, RPT.length, Data.P\_Box);
46. **return** RPT;
48. }

</code>

1. **主要仪器设备及耗材**

编程语言：Java

操作系统：Windows 10

IDE: JETBRIAN IDEA

1. **实验步骤**

采用javafx制作图形化界面

1 利用程序加密、解密字符串

A在文本框输入待加密的字符串

B 在输入框输入密钥

C 点击加密

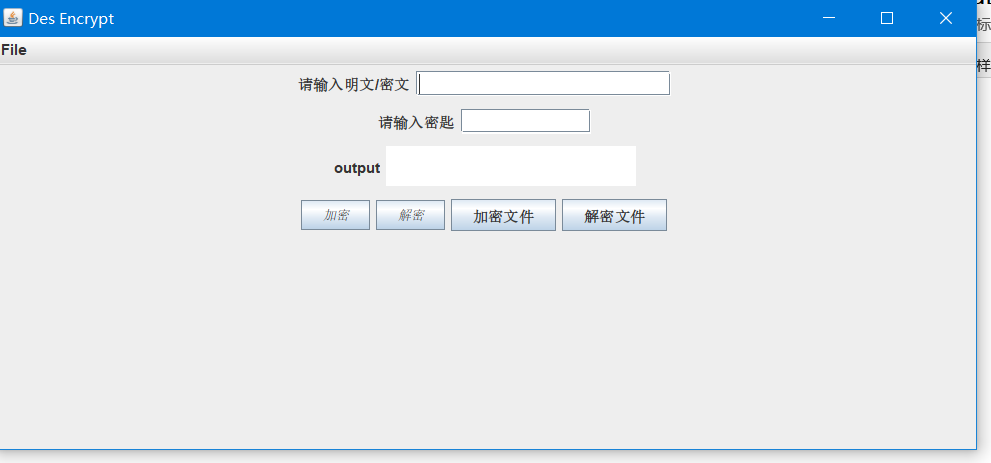
D 将密文复制到输入框，点击解密

2 利用程序加密、解密文件

A 选择文件

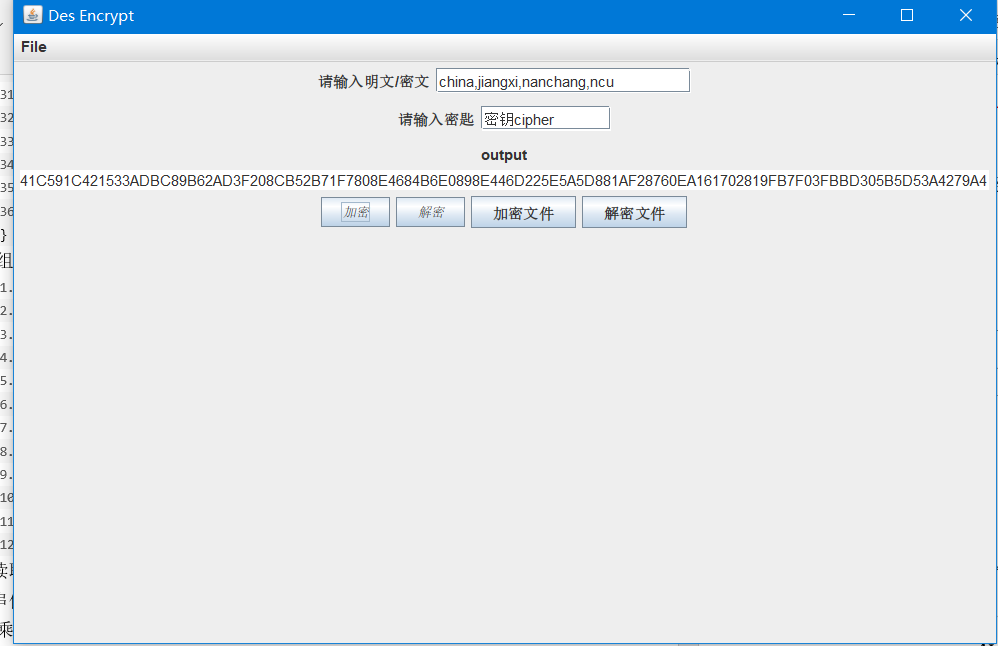
B 点击加密

C 选择加密后的文件, 点击解密.

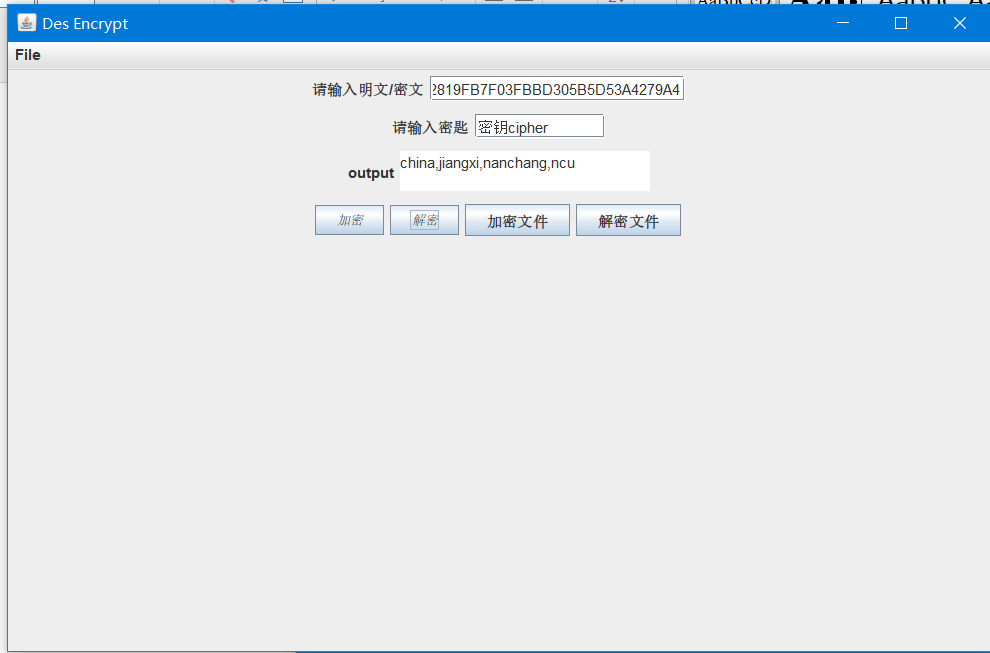


1. **实验数据及处理结果**

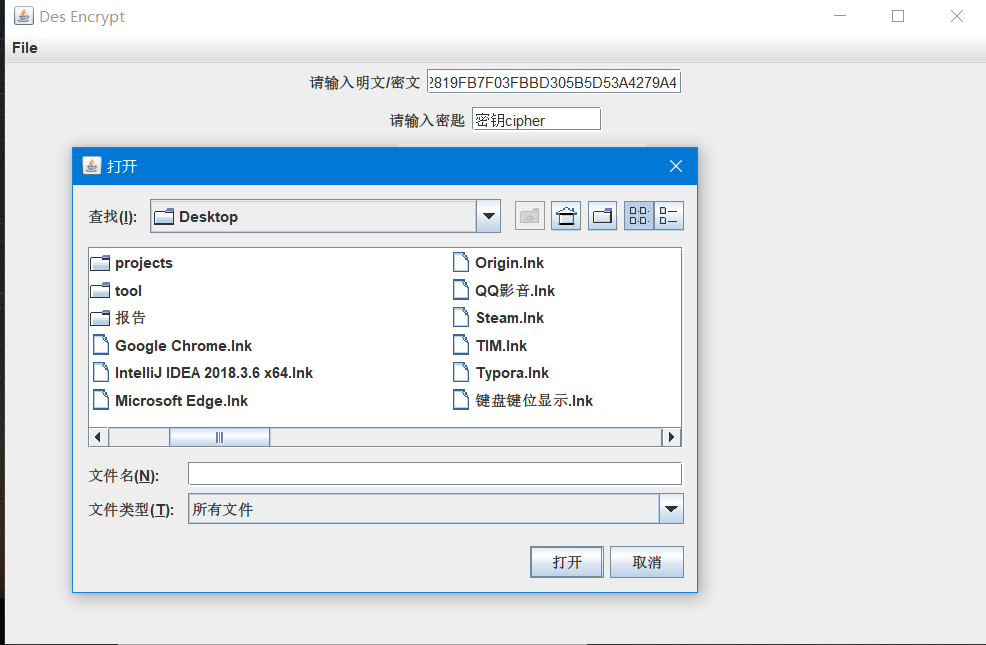
**加密字符串**

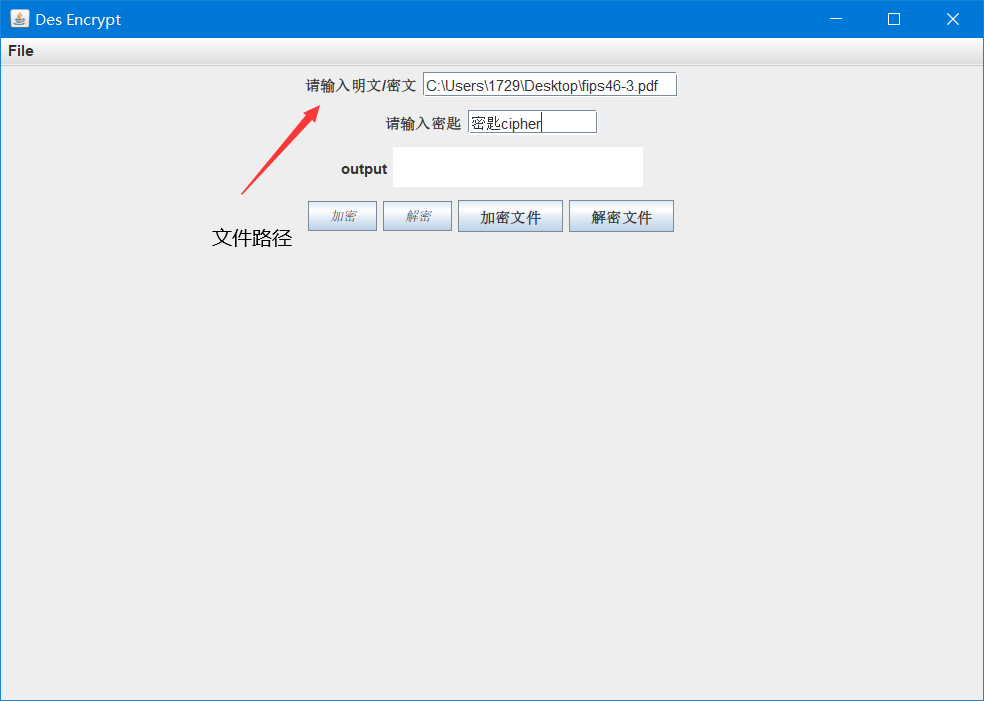


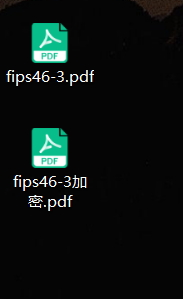
**解密字符串**

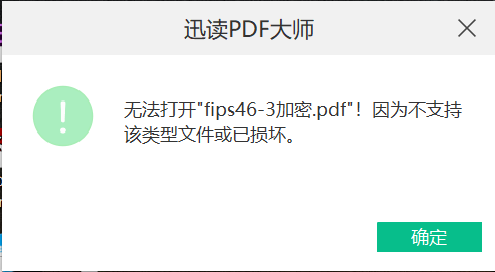


加密文件

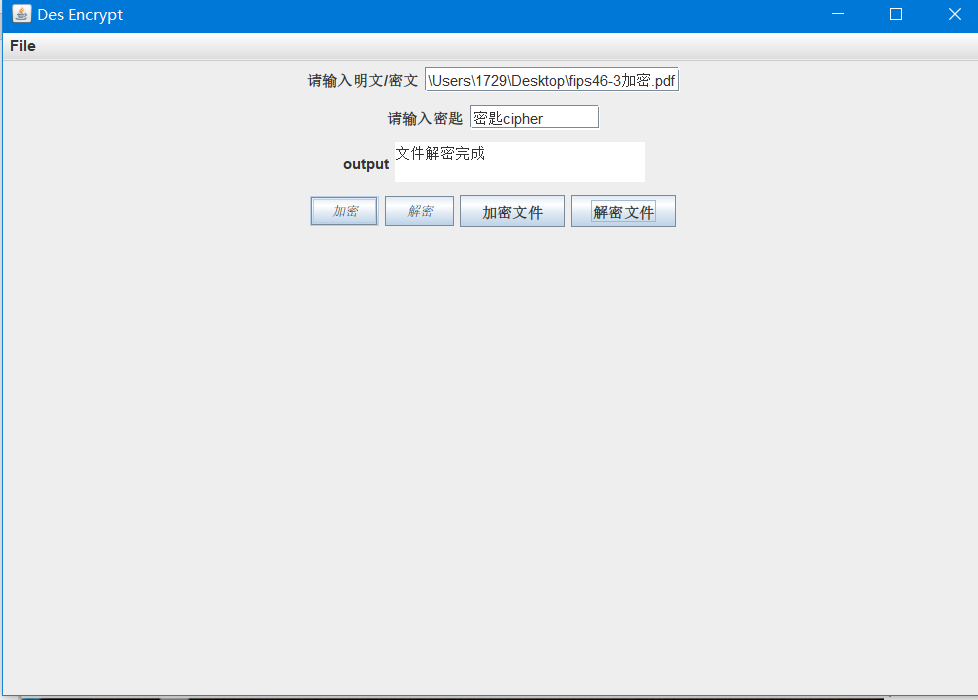


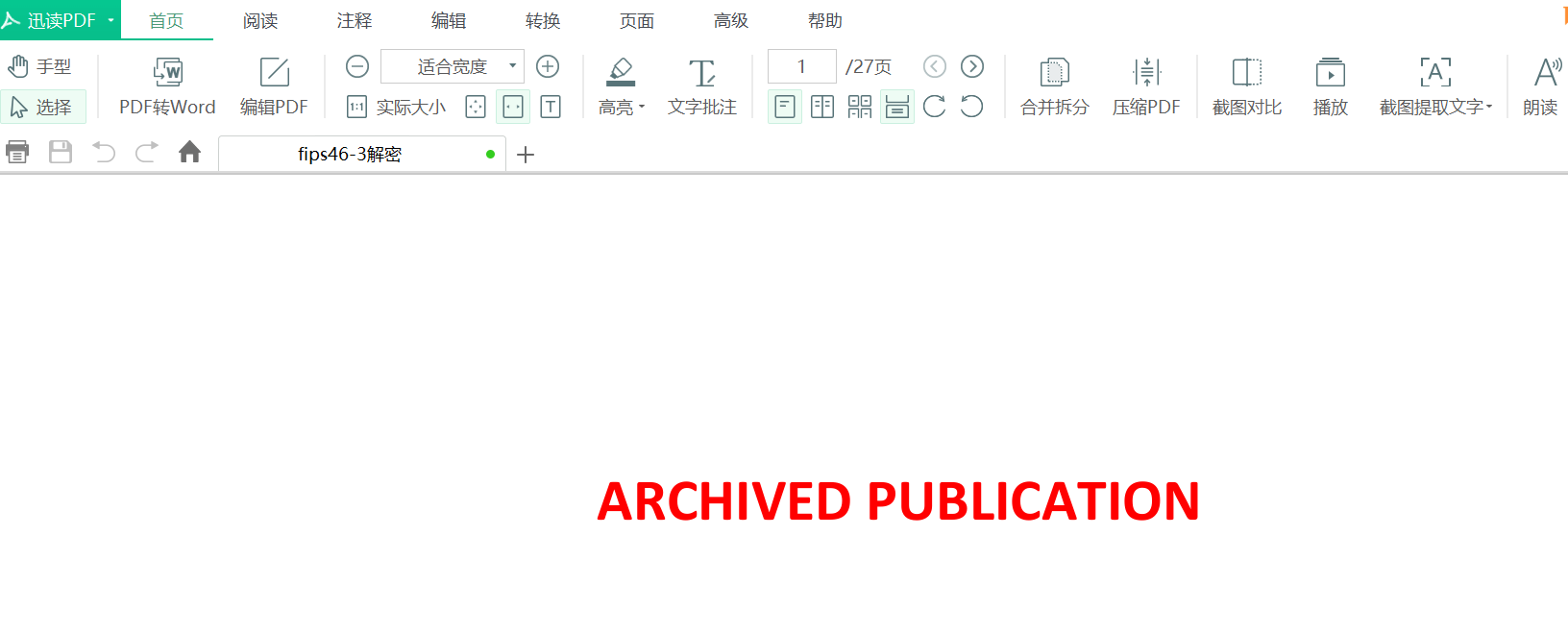




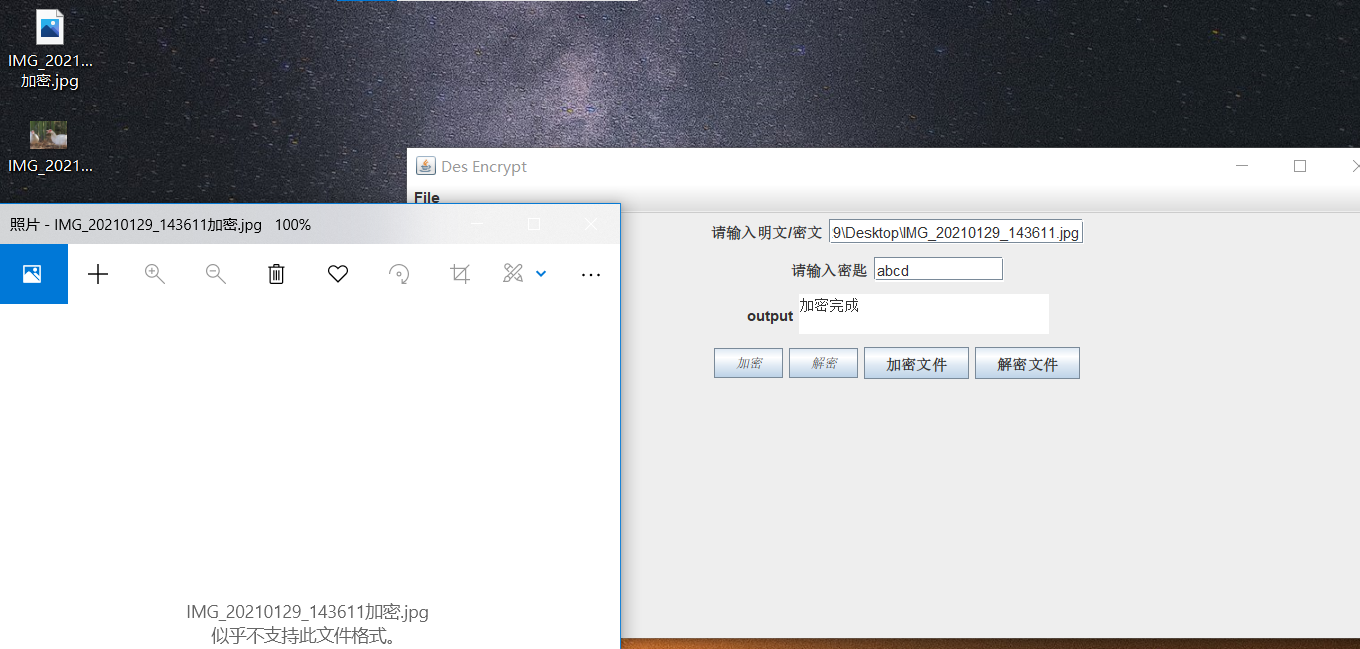


文件解密

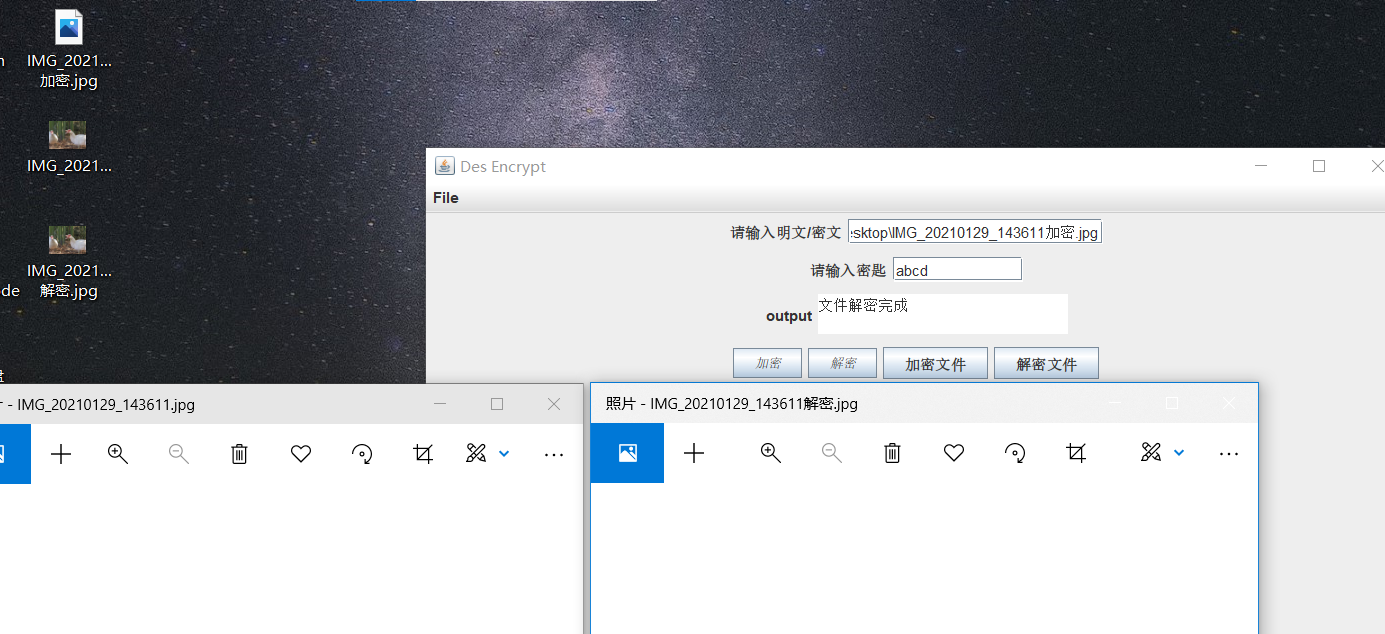




加密图片



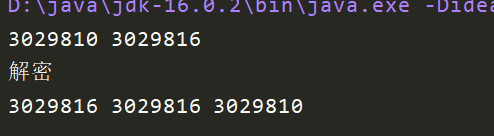
解密图片



1. **思考讨论题或体会或对改进实验的建议**

在加密过程中，常常会碰到明文分组不足4位，或者8位的情况。

为了使得加密正常进行，我们可以将约定，如果少i个字符，那么我就明文的末位补上i个i。在解密过程中，通过读取解密密文的最后一位，将增补的字符删去即可。



1. **参考资料**

1 <应用密码学>

2 [FIPS 46-3: The official document describing the DES standard](http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf) (PDF)