合肥工业大学

# 《信息安全技术》实验报告

# 密码学实验部分

|  |
| --- |
| 姓名 |
| 学号 |
| 专业班级 |

2022年5月

目录

[《信息安全技术》实验报告 1](#_Toc8840)

[密码学实验部分 1](#_Toc1869)

[一、实验原理 1](#_Toc6141)

[1.1 DES加解密 1](#_Toc25827)

[1.2 AES加解密 1](#_Toc3457)

[1.3 RSA加解密 2](#_Toc294)

[二、实验环境 3](#_Toc19304)

[2.1 硬件 3](#_Toc29697)

[2.2 软件 3](#_Toc16206)

[三、实验内容 3](#_Toc15501)

[四、实验结果 3](#_Toc23568)

[4.1 DES加解密 3](#_Toc22968)

[4.1.1 DES加解密字符串 4](#_Toc9821)

[4.1.2 DES加解密图片 4](#_Toc8453)

[4.2 AES加解密 6](#_Toc15095)

[4.2.1 AES加解密字符串 6](#_Toc3153)

[4.2.2 AES加解密文件 9](#_Toc17377)

[4.3 RSA加解密 13](#_Toc16071)

[五、实验心得 15](#_Toc15366)

# 一、实验原理

## 1.1 DES加解密

DES全称为Data Encryption Standard，即数据加密标准，是一种使用密钥加密的块算法，1977年被美国联邦政府的国家标准局确定为联邦资料处理标准（FIPS），并授权在非密级政府通信中使用，随后该算法在国际上广泛流传开来。

对称性：DES是对称的，也就是说它使用同一个密钥来加密和解密数据。与此相对的是RSA加密算法，是一种非对称加密算法

分组性：DES还是一种分组加密算法，该算法每次处理固定长度的数据段，称之为分组。DES分组的大小是64位，如果加密的数据长度不是64位的倍数，可以按照某种具体的规则来填充位。

“混乱和扩散”的原则：混乱的目的是为隐藏任何明文同密文、或者密钥之间的关系，而扩散的目的是使明文中的有效位和密钥一起组成尽可能多的密文。两者结合到一起就使得安全性变得相对较高。

DES算法具体通过对明文进行一系列的排列和替换操作来将其加密：过程的关键就是从给定的初始密钥中得到16个子密钥的函数。要加密一组明文，每个子密钥按照顺序（1-16）以一系列的位操作施加于数据上，每个子密钥一次，一共重复16次。每一次迭代称之为一轮。要对密文进行解密可以采用同样的步骤，只是子密钥是按照逆向的顺序（16-1）对密文进行处理。

## 1.2 AES加解密

AES全称为Advanced Encryption Standard，是美国联邦政府采用的一种区块加密标准，用来替代原先的DES。

AES加密过程涉及到4种操作：字节替代（Sub Bytes）、行移位（Shift Rows）、列混淆（Mix Columns）和轮密钥加（Add Round Key）。解密过程分别为对应的逆操作。由于每一步操作都是可逆的，按照相反的顺序进行解密即可恢复明文。加解密中每轮的密钥分别由初始密钥扩展得到。算法中16字节的明文、密文和轮密钥都以一个4⨉4的矩阵表示。AES的具体加密解密流程，如图1.1所示。

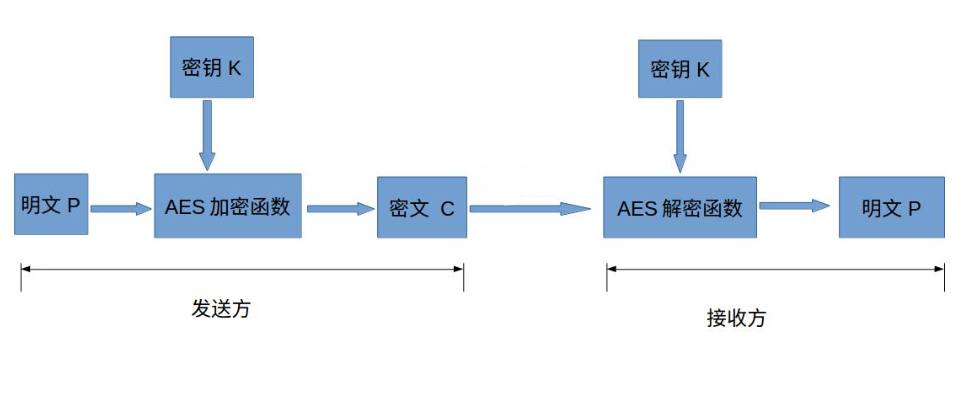


图 1.1 AES加密解密流程

下面简单介绍下各个部分的作用与意义：

* **明文P**

没有经过加密的数据。

* **密钥K**

用来加密明文的密码，在对称加密算法中，加密与解密的密钥是相同的。密钥为接收方与发送方协商产生，但不可以直接在网络上传输，否则会导致密钥泄漏，通常是通过非对称加密算法加密密钥，然后再通过网络传输给对方，或者直接面对面商量密钥。密钥是绝对不可以泄漏的，否则会被攻击者还原密文，窃取机密数据。

* **AES加密函数**

设AES加密函数为E，则 C = E(K, P),其中P为明文，K为密钥，C为密文。也就是说，把明文P和密钥K作为加密函数的参数输入，则加密函数E会输出密文C。

* **密文C**

经加密函数处理后的数据

* **AES解密函数**

设AES解密函数为D，则 P = D(K, C),其中C为密文，K为密钥，P为明文。也就是说，把密文C和密钥K作为解密函数的参数输入，则解密函数会输出明文P。

## 1.3 RSA加解密

RSA加密算法，是世界上第一个非对称加密算法，也是数论的第一个实际应用。它的算法如下：

1. 找两个非常大的质数p和q（通常p和q都有155十进制位或都有512十进制位）并计算n=pq，k=(p-1)(q-1)。
2. 将明文编码成整数M，保证M不小于0但是小于n。
3. 任取一个整数e，保证e和k互质，而且e不小于0但是小于k。加密钥匙（称作公钥）是(e, n)。
4. 找到一个整数d，使得ed除以k的余数是1（只要e和n满足上面条件，d肯定存在）。解密钥匙（称作密钥）是(d, n)。

加密过程：加密后的编码C等于M的e次方除以n所得的余数。

解密过程：解密后的编码N等于C的d次方除以n所得的余数。

只要e、d和n满足上面给定的条件。M等于N。

# 二、实验环境

## 2.1 硬件

暗夜精灵5笔记本电脑。

## 2.2 软件

运行系统：Windows 10

开发工具：IntelliJ IDEA 2019、Java 1.8

编程语言：Java

# 三、实验内容

根据实验指导书的要求，编程实现第一节“实验原理”中提到的相关算法，并设计相关数据进行测试，记录实验结果；

# 四、实验结果

## 4.1 DES加解密

本次实验，实现了DES加解密的两种不同形式：字符串加解密与文件加解密。

### 4.1.1 DES加解密字符串

本次实验，使用以下代码对字符串进行DES加解密测试。

|  |
| --- |
| public class DES\_Test {  public static void main(String[] args) {  String es = DES.*encrypt*("aaaaaaaa","asdisadjcnmvsdfdsf.ad");  System.*out*.println(es);  String ds = DES.*encrypt*("aaaaaaaa",es);  System.*out*.println(ds);  } } |

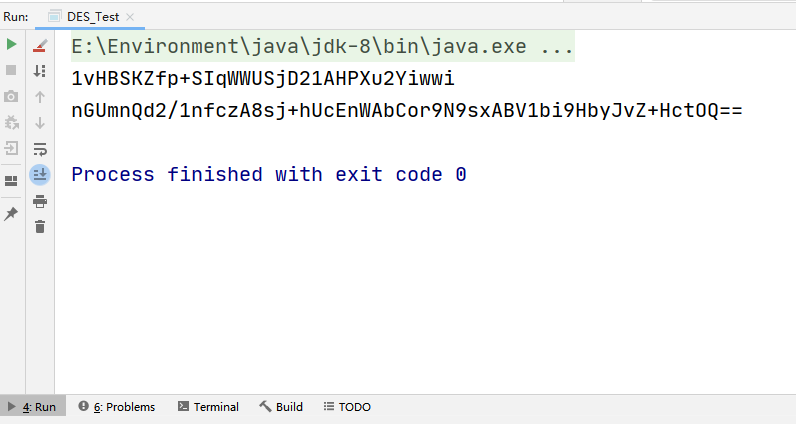


图4.1字符串加密结果

### 4.1.2 DES加解密**图片**

本次实验，使用以下代码对图片进行DES加解密测试。

|  |
| --- |
| public class DES\_Test {  public static void main(String[] args) {  DES.*encryptFile*("aaaaaaaa","src/test.png","src/test.png.e"); DES.*decryptFile*("aaaaaaaa","src/test.png.e","src/test\_after.png");  } } |

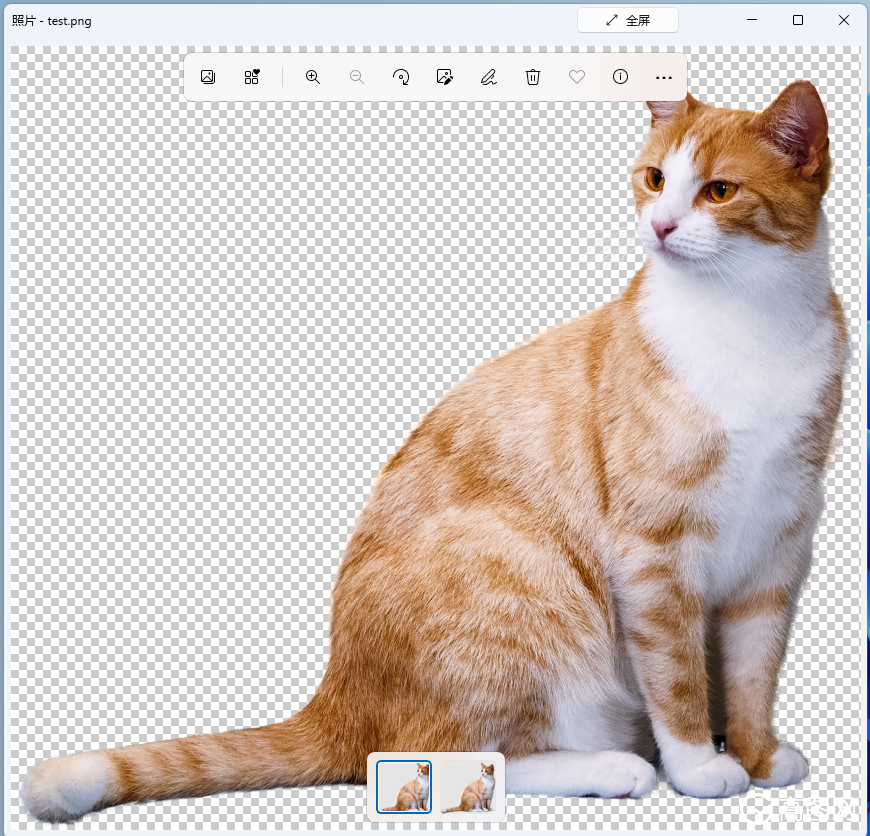


图 4.2 打开测试原图片

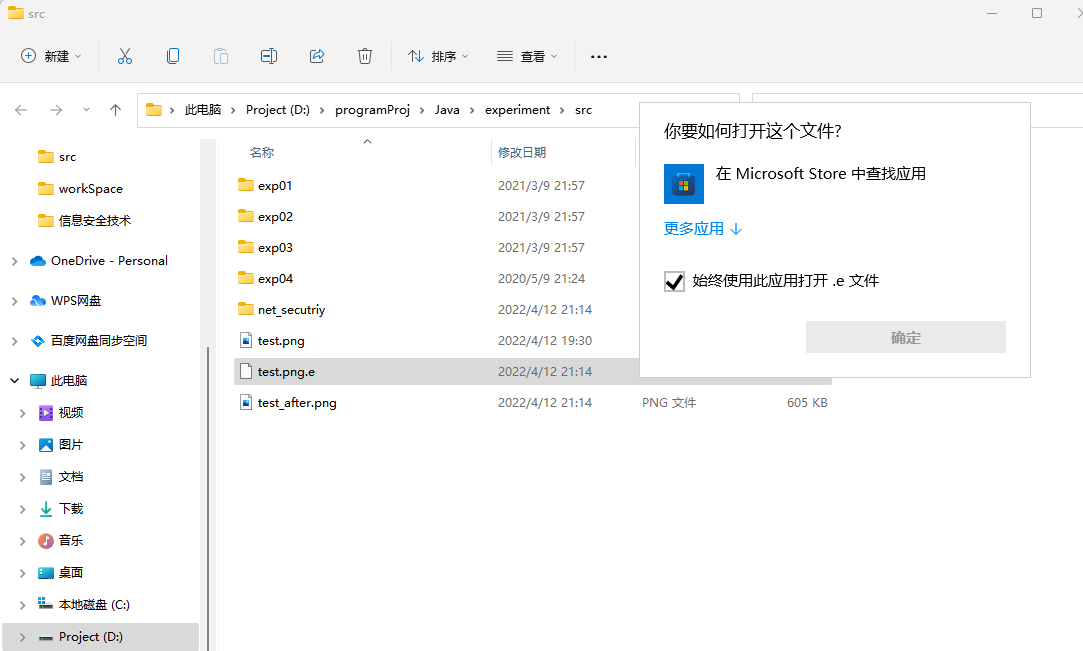


图 4.3 打开加密后的测试图片

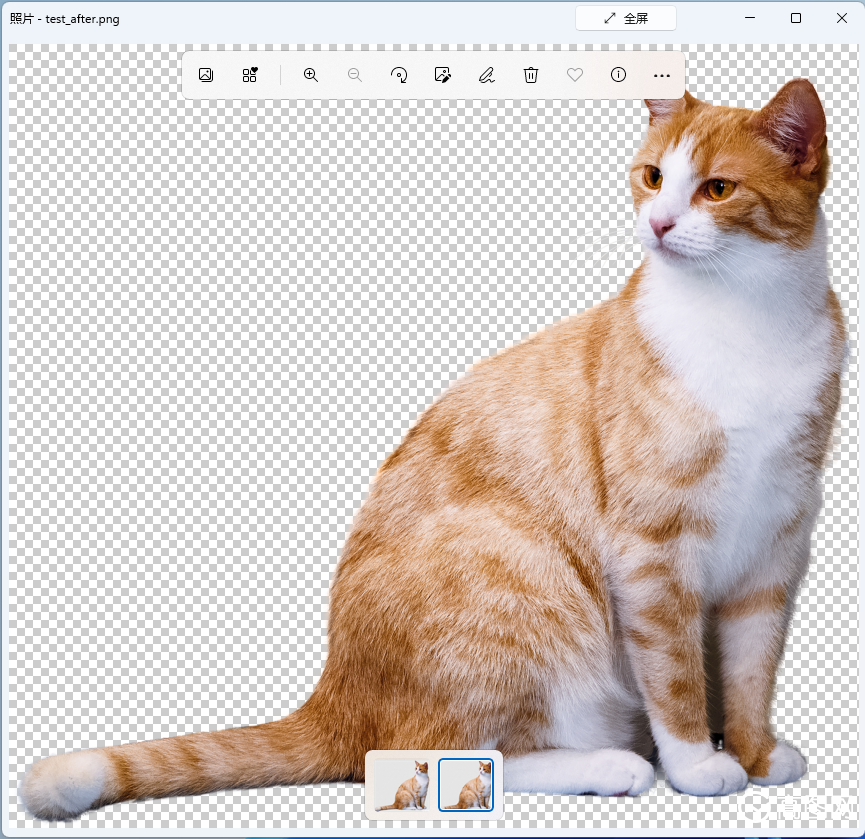


图 4.4 打开解密后的测试图片

本次实验，使用DES算法程序加密解密图片的过程，如图4.3所示，其中a)为加密，b)为解密，加解密的key值为123qwe。

使用Windows自带的照片查看器打开测试原图，如图4.2所示。

使用Windows自带的照片查看器打开被加密后的原测试图，如图4.3所示，显然，原测试图经过加密后，无法正常打开。

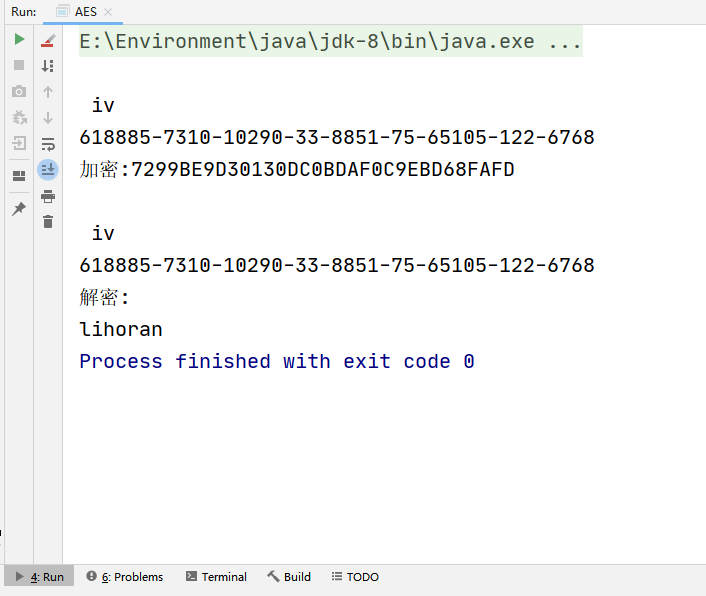
使用Windows自带的照片查看器打开被解密后的原被加密测试图，如图4.4所示，显然，原被加密图经过解密后，可以正常打开。

## 4.2 AES加解密

### 4.2.1 AES加解密字符串

本次实验，使用以下代码对字符串进行AES加解密测试。

|  |
| --- |
| public class AES {  public static void main(String[] args) throws UnsupportedEncodingException {  String string = "lihoran";  String key = "1234123412341234";  byte[] result = *encrypt*(string.getBytes("UTF-8"), key.getBytes("UTF-8"));  System.*out*.println("加密:"+*bytes\_String16*(result));  byte[] dresult = *decrypt*(result, key.getBytes("UTF-8"));  System.*out*.println("解密:");  for (int i = 0; i < dresult.length; i++) {  System.*out*.print((char) dresult[i]);  }  }    public static void print\_bytes(byte[] b){  System.*out*.print("\n iv \n");  for (int i=0;i<16;i++){  System.*out*.print(b[i]);  }  System.*out*.print("\n");  }   public static String bytes\_String16(byte[] b) {  char[] \_16 = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };  StringBuilder sb = new StringBuilder();  for (int i = 0; i < b.length; i++) {  sb.append(\_16[b[i] >> 4 & 0xf]).append(\_16[b[i] & 0xf]);  }  return sb.toString();  }   public static byte[] encrypt(byte[] data, byte[] key) {  if (key.length != 16) {  throw new RuntimeException("Invalid AES key length (must be 16 bytes)");  }  try {  SecretKeySpec secretKey = new SecretKeySpec(key, "AES");  byte[] enCodeFormat = secretKey.getEncoded();  SecretKeySpec seckey = new SecretKeySpec(enCodeFormat, "AES");  *// 创建密码器* Cipher cipher = Cipher.*getInstance*("AES/CBC/PKCS5Padding");  byte[] iv = new byte[cipher.getBlockSize()];    SecureRandom secureRandom = new SecureRandom();  secureRandom.setSeed(key);  secureRandom.nextBytes(iv);  IvParameterSpec niv = new IvParameterSpec(iv);  cipher.init(Cipher.*ENCRYPT\_MODE*, seckey, niv);*// 初始化  print\_bytes*(iv);  byte[] result = cipher.doFinal(data);  return result; *// 加密* } catch (Exception e) {  e.printStackTrace();  throw new RuntimeException("encrypt fail!", e);  }  }   public static byte[] decrypt(byte[] data, byte[] key) {  if (key.length != 16) {  throw new RuntimeException("Invalid AES key length (must be 16 bytes)");  }  try {  SecretKeySpec secretKey = new SecretKeySpec(key, "AES");  byte[] enCodeFormat = secretKey.getEncoded();  SecretKeySpec seckey = new SecretKeySpec(enCodeFormat, "AES");  *// 创建密码器* Cipher cipher = Cipher.*getInstance*("AES/CBC/PKCS5Padding");  byte[] iv = new byte[cipher.getBlockSize()];    SecureRandom secureRandom = new SecureRandom();  secureRandom.setSeed(key);  secureRandom.nextBytes(iv);  IvParameterSpec niv = new IvParameterSpec(iv);  cipher.init(Cipher.*DECRYPT\_MODE*, seckey, niv);*// 初始化  print\_bytes*(iv);  byte[] result = cipher.doFinal(data);  return result; *// 解密* } catch (Exception e) {  e.printStackTrace();  throw new RuntimeException("decrypt fail!", e);  }  }   public static byte[] genarateRandomKey() {  KeyGenerator keygen = null;  try {  keygen = KeyGenerator.*getInstance*("AES/CBC/PKCS5Padding");  } catch (NoSuchAlgorithmException e) {  throw new RuntimeException(" genarateRandomKey fail!", e);  }  SecureRandom random = new SecureRandom();  keygen.init(random);  Key key = keygen.generateKey();  return key.getEncoded();  } } |



### 4.2.2 AES加解密文件

本次实验，使用以下代码对文件（以图片为例）进行AES加密解密测试。

|  |
| --- |
| */\*\*  \* AES加密  \*  \* @param plaintext 明文  \* @param Key 密钥  \* @param EncryptMode AES加密模式，CBC或ECB  \* @return 该字符串的AES密文值  \*/* public static String AES\_Encrypt(String plaintext, String Key,String EncryptMode) {  String PlainText=null;  try {  PlainText=plaintext;  if (Key == null) {  return null;  }  Key = *getMD5*(Key);  byte[] raw = Key.getBytes("utf-8");  SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");  Cipher cipher = Cipher.*getInstance*("AES/"+EncryptMode+"/PKCS5Padding");  if(EncryptMode=="ECB") {  cipher.init(Cipher.*ENCRYPT\_MODE*, skeySpec);  }else {  IvParameterSpec iv = new IvParameterSpec(Key.getBytes("utf-8"));*//使用CBC模式，需要一个向量iv，可增加加密算法的强度* cipher.init(Cipher.*ENCRYPT\_MODE*, skeySpec, iv);  }  byte[] encrypted = cipher.doFinal(PlainText.getBytes("utf-8"));  String encryptedStr=new String(new BASE64Encoder().encode(encrypted));  return encryptedStr;  *//return new String(encrypted);//此处使用BASE64做转码功能，同时能起到2次加密的作用。* } catch (Exception ex) {  System.*out*.println(ex.toString());  return null;  } }  */\*\*  \* AES解密  \*  \* @param cipertext 密文  \* @param Key 密钥  \* @param EncryptMode AES加密模式，CBC或ECB  \* @return 该密文的明文  \*/* public static String AES\_Decrypt(String cipertext, String Key,String EncryptMode) {  String CipherText=null;  try {  CipherText=cipertext;  *// 判断Key是否正确* if (Key == null) {  *//System.out.print("Key为空null");* return null;  }  Key=*getMD5*(Key);  byte[] raw = Key.getBytes("utf-8");  SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");  Cipher cipher=Cipher.*getInstance*("AES/"+EncryptMode+"/PKCS5Padding");  if(EncryptMode=="ECB") {  cipher.init(Cipher.*DECRYPT\_MODE*, skeySpec);  }  else  {  IvParameterSpec iv = new IvParameterSpec(Key.getBytes("utf-8"));*//使用CBC模式，需要一个向量iv，可增加加密算法的强度* cipher.init(Cipher.*DECRYPT\_MODE*, skeySpec, iv);  }  byte[] encrypted1 = new BASE64Decoder().decodeBuffer(CipherText);*//先用base64解密  //byte[] encrypted1 = CipherText.getBytes();* try {  byte[] original = cipher.doFinal(encrypted1);  String originalString = new String(original,"utf-8");  return originalString;  } catch (Exception e) {  System.*out*.println(e.toString());  return null;  }  } catch (Exception ex) {  System.*out*.println(ex.toString());  return null;  } } |

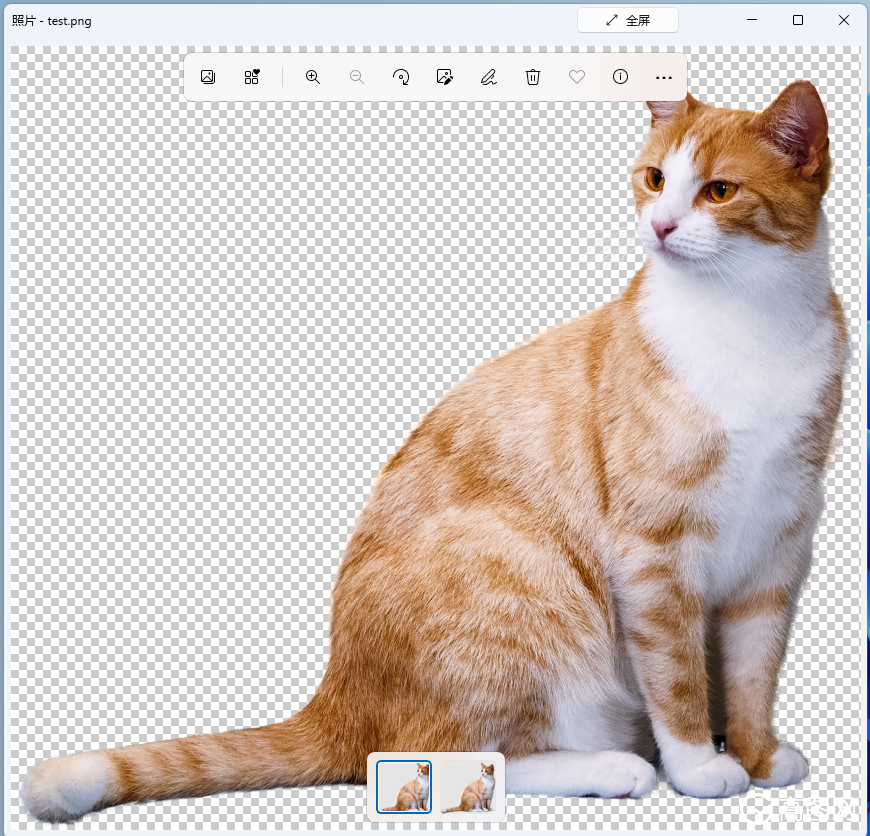


图 4.38 打开测试原图片

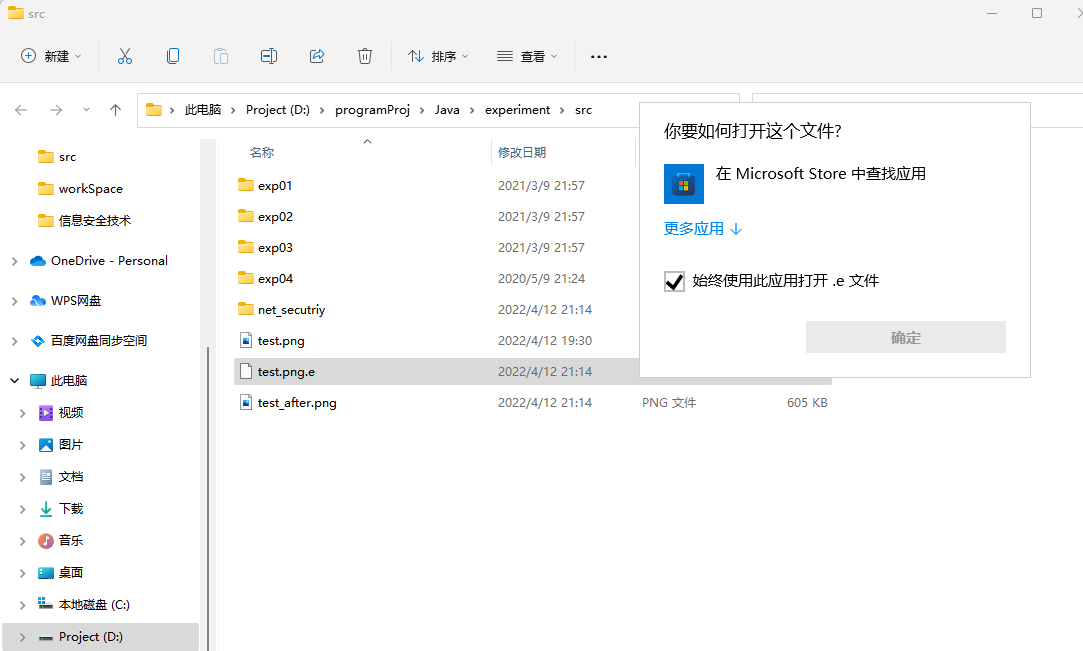


图 4.39 打开加密后的测试图片

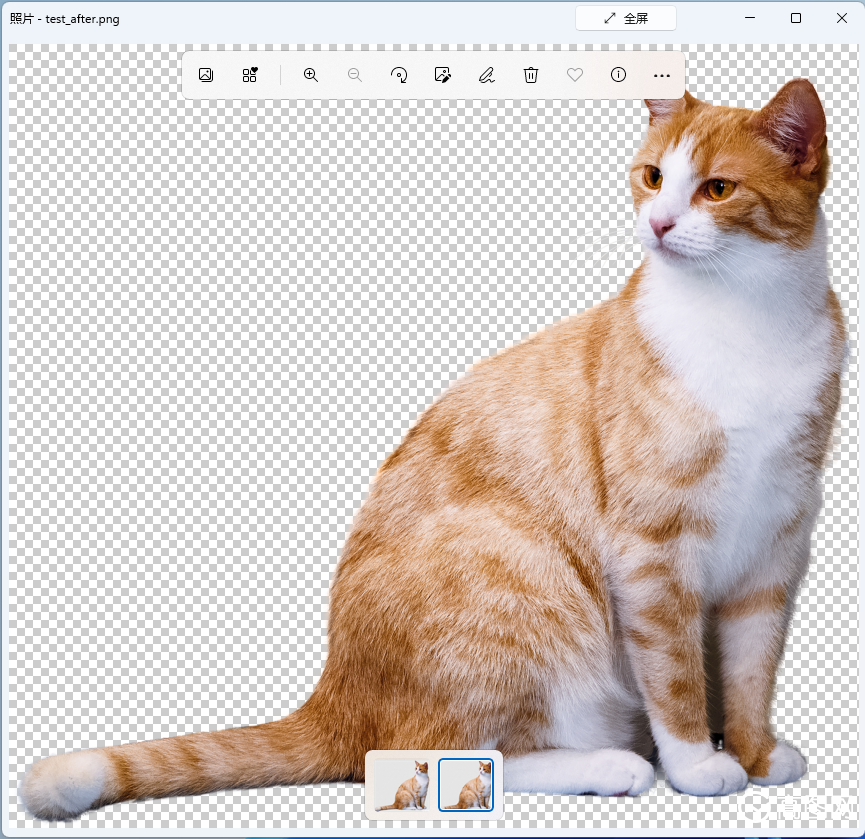


图 4.40 打开解密后的测试图片

本次实验，使用AES算法程序加密解密图片的过程，分别如图4.36与图4.37所示，加密的key值均为123qwe，IV值均为None。

使用Windows自带的照片查看器打开测试原图片，如图4.38所示。

使用Windows自带的照片查看器打开被加密后的原测试图片，如图4.39所示，显然，原测试图片经过加密后，无法正常打开。

使用Windows自带的照片查看器打开被解密后的原被加密测试图片，如图4.40所示，显然，原被加密图片经过解密后，可以正常打开。

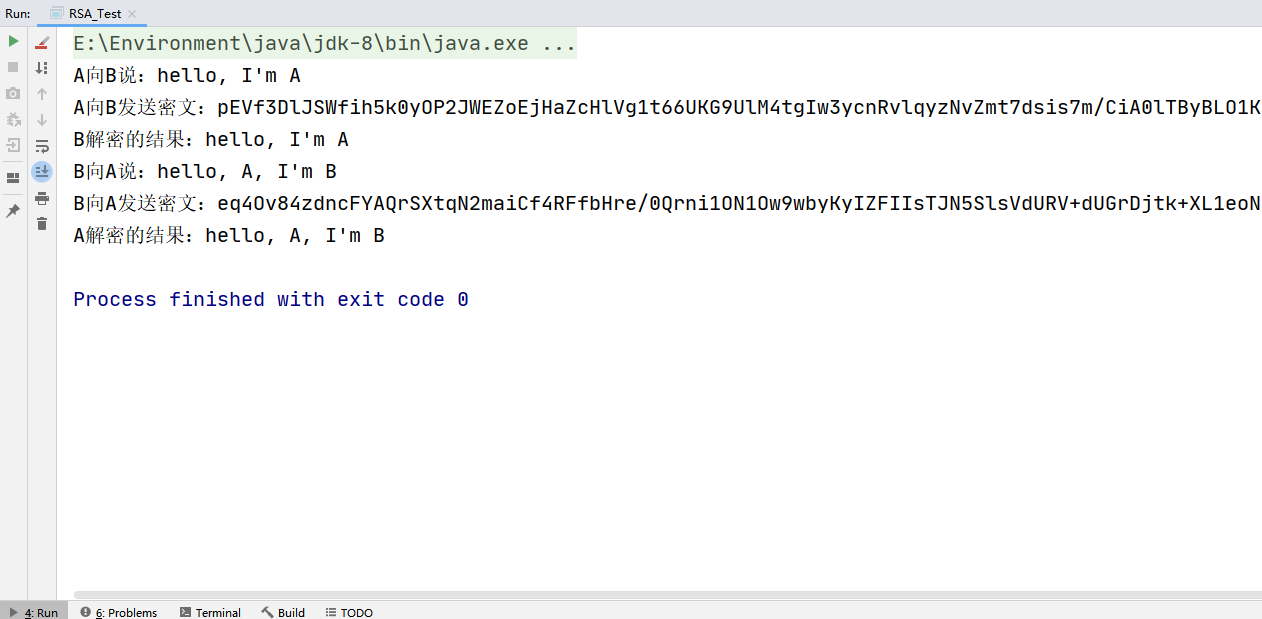
本实验所实现的AES算法程序，在对文件（图片）进行加解密的过程中，会保留原文件（图片），方便对比。另外，为使实验结果不冗余，本次实验实现AES算法对文件进行加解密，仅以图片为例，其他格式的文件可以参照加解密图片的过程操作。

## 4.3 RSA加解密

本次实验，使用以下代码对字符串进行RSA加解密测试。

|  |
| --- |
| public class RSA\_Test {  public static void main(String[] args) {  try {  RSA\_OBJ obj\_A=new RSA\_OBJ();    RSA\_OBJ obj\_B=new RSA\_OBJ();  String data\_A2B="hello, I'm A";  String en\_result=obj\_A.*AEncrypt*(data\_A2B,obj\_B.*APublicKey*() );  System.*out*.println("A向B说："+data\_A2B);  System.*out*.println("A向B发送密文："+en\_result);  String de\_result=obj\_B.*ADecrypt*(en\_result);  System.*out*.println("B解密的结果："+de\_result);    String data\_B2A="hello, A, I'm B";  en\_result=obj\_B.*AEncrypt*(data\_B2A,obj\_A.*APublicKey*() );  System.*out*.println("B向A说："+data\_B2A);  System.*out*.println("B向A发送密文："+en\_result);  de\_result=obj\_A.*ADecrypt*(en\_result);  System.*out*.println("A解密的结果："+de\_result);    }   catch (Exception e) {  e.printStackTrace();   }  } } |

实验结果如图所示。



非对称加密算法（如RSA）同前文所述DES、AES等对称加密算法不同的是，对称加密算法的加密与解密密钥相同，而非对称加密算法的加密密钥与解密密钥不同。众所周知，RSA加密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。

# 五、实验心得

通过本次实验，我对DES、AES、RSA算法有了基本的了解，结合课上学习的理论知识，结合应用实践，让我对原理具有更深的理解，增强了自己的代码能力。另外，根据实验指导书的使用Java语言的教程，我通过Java语言进一步的联系和实现了实验所需要的效果，不仅十分具有成就感，也对语言和加密算法有了更加深刻的认识。