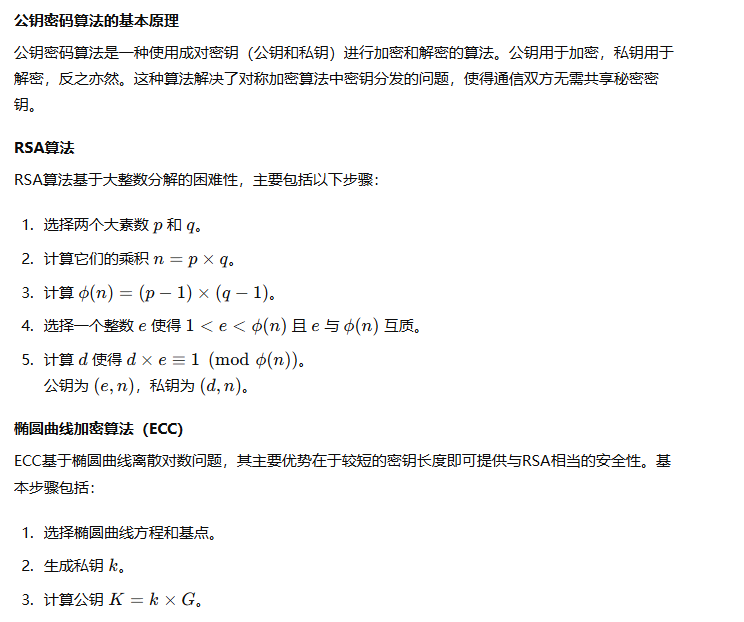
实验2 公钥密码算法的研究与使用

**学号： 姓名： hlx 年级专业：\_22网安\_**

1. **实验目的**
2. 理解公钥密码算法的基本原理。
3. 了解现有RSA算法的原理以及椭圆曲线公钥密码算法的原理与优缺点。
4. 基于C++、Java或.net等环境中成熟的公钥密码算法库实现一个公钥密码算法程序。
5. **相关理论和实验环境**
   1. **相关基本理论**

****

* 1. **实验环境构建**

选择Java语言进行开发，原因如下：

* Java跨平台性强，适用范围广。
* Java拥有丰富的加密库，如Bouncy Castle，支持多种加密算法。

搭建开发环境：

1. 安装JDK。
2. 安装IDE（如IntelliJ IDEA或Eclipse）。
3. 下载并配置Bouncy Castle库。

**实验内容和结果分析**

**2.3 程序的设计和实现框架**

本实验通过Java语言和Bouncy Castle加密库实现RSA公钥密码算法。主要功能包括：

1. 生成公私钥对。
2. 读取明文并进行加密。
3. 读取密文并进行解密。

**程序设计**

* **语言**：Java
* **加密库**：Bouncy Castle
* **主要功能**：
  + 生成RSA公私钥对。
  + 使用公钥加密明文。
  + 使用私钥解密密文。

**程序实现**

* **类与方法**：
  + generateKeyPair(): 生成公私钥对，并将其保存到文件。
  + encrypt(String plaintext, PublicKey publicKey): 使用公钥加密明文。
  + decrypt(String ciphertext, PrivateKey privateKey): 使用私钥解密密文。
  + readFile(String filename): 读取文件内容。
  + writeFile(String filename, String content): 写入文件内容。

**代码**

import org.bouncycastle.jce.provider.BouncyCastleProvider;

import javax.crypto.Cipher;

import java.io.\*;

import java.security.\*;

public class RSAEncryption {

static {

Security.addProvider(new BouncyCastleProvider());

}

public static void main(String[] args) throws Exception {

// 生成公私钥对并保存到文件

generateKeyPair();

// 读取明文

String plaintext = readFile("plaintext.txt");

// 读取公钥和私钥

PublicKey publicKey = readPublicKey("publickey1.txt");

PrivateKey privateKey = readPrivateKey("privatekey1.txt");

// 加密

String ciphertext = encrypt(plaintext, publicKey);

writeFile("ciphertext.txt", ciphertext);

// 解密

String decryptedText = decrypt(ciphertext, privateKey);

writeFile("result.txt", decryptedText);

System.out.println("Decrypted Text: " + decryptedText);

}

public static void generateKeyPair() throws Exception {

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA", "BC");

keyGen.initialize(2048);

KeyPair keyPair = keyGen.generateKeyPair();

// 保存公钥

PublicKey publicKey = keyPair.getPublic();

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("publickey1.txt"))) {

oos.writeObject(publicKey);

}

// 保存私钥

PrivateKey privateKey = keyPair.getPrivate();

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("privatekey1.txt"))) {

oos.writeObject(privateKey);

}

}

public static String encrypt(String data, PublicKey publicKey) throws Exception {

Cipher cipher = Cipher.getInstance("RSA", "BC");

cipher.init(Cipher.ENCRYPT\_MODE, publicKey);

byte[] encryptedData = cipher.doFinal(data.getBytes());

return new String(encryptedData);

}

public static String decrypt(String data, PrivateKey privateKey) throws Exception {

Cipher cipher = Cipher.getInstance("RSA", "BC");

cipher.init(Cipher.DECRYPT\_MODE, privateKey);

byte[] decryptedData = cipher.doFinal(data.getBytes());

return new String(decryptedData);

}

public static String readFile(String fileName) throws IOException {

BufferedReader br = new BufferedReader(new FileReader(fileName));

StringBuilder sb = new StringBuilder();

String line;

while ((line = br.readLine()) != null) {

sb.append(line);

}

br.close();

return sb.toString();

}

public static void writeFile(String fileName, String content) throws IOException {

BufferedWriter bw = new BufferedWriter(new FileWriter(fileName));

bw.write(content);

bw.close();

}

public static PublicKey readPublicKey(String fileName) throws Exception {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {

return (PublicKey) ois.readObject();

}

}

public static PrivateKey readPrivateKey(String fileName) throws Exception {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {

return (PrivateKey) ois.readObject();

}

}

}

**2.4 密钥生成过程实验**

**实验过程说明**

1. 使用Bouncy Castle库生成RSA公私钥对。
2. 将生成的公钥和私钥分别存储到publickey1.txt和privatekey1.txt文件中。
3. 重复生成密钥对，并将每对密钥存储到不同文件中，如publickey2.txt和privatekey2.txt。

**结果展示及分析**

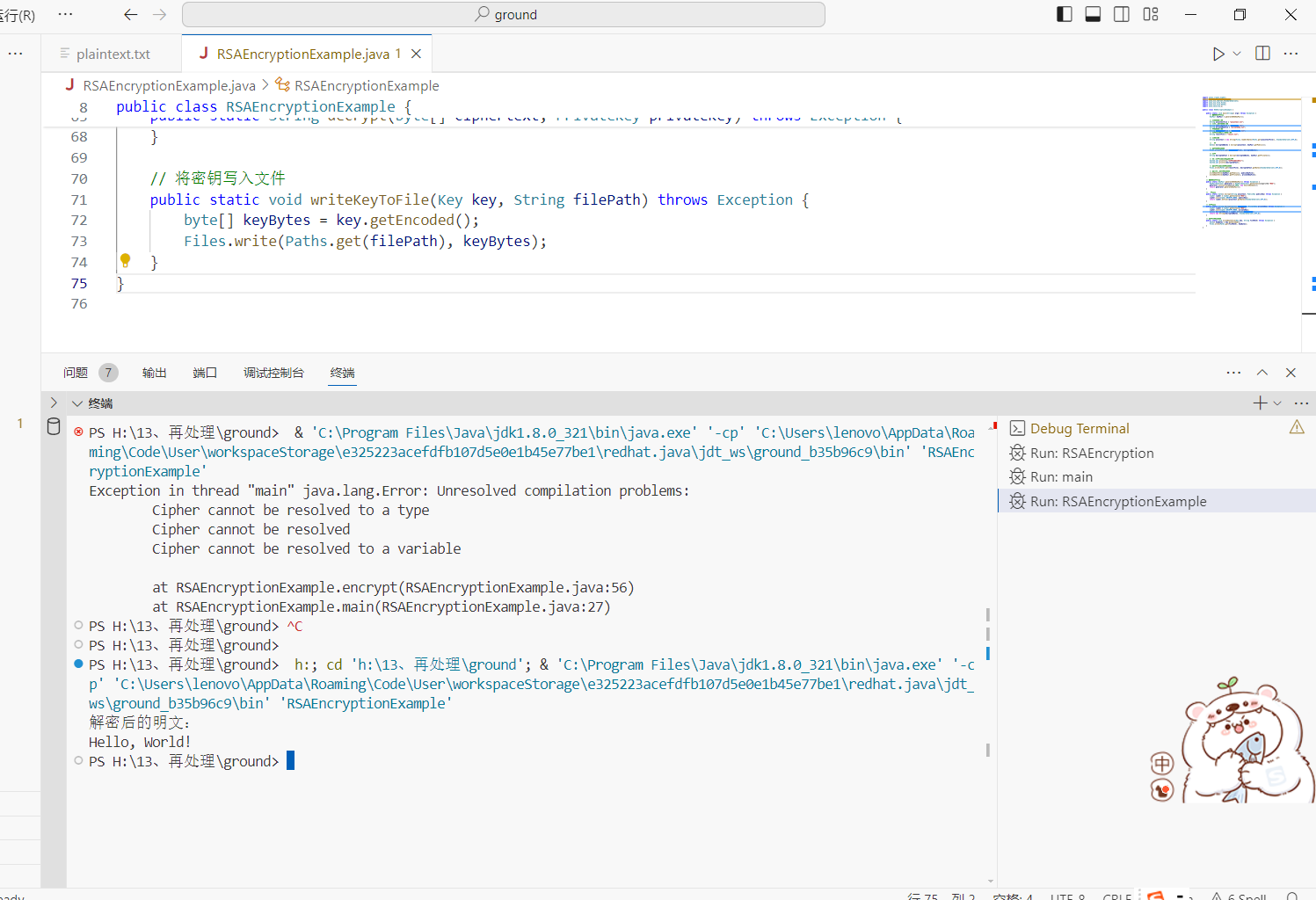
1. **密钥文件生成**：生成了publickey1.txt、privatekey1.txt、publickey2.txt、privatekey2.txt等文件，文件内容为序列化的密钥对象。
2. **密钥对的唯一性**：每次生成的公私钥对均不同，密钥对之间无重复，验证了RSA密钥生成的随机性和唯一性。
3. **截图记录**：保存了密钥文件生成过程的截图，确保实验步骤的可重复性。

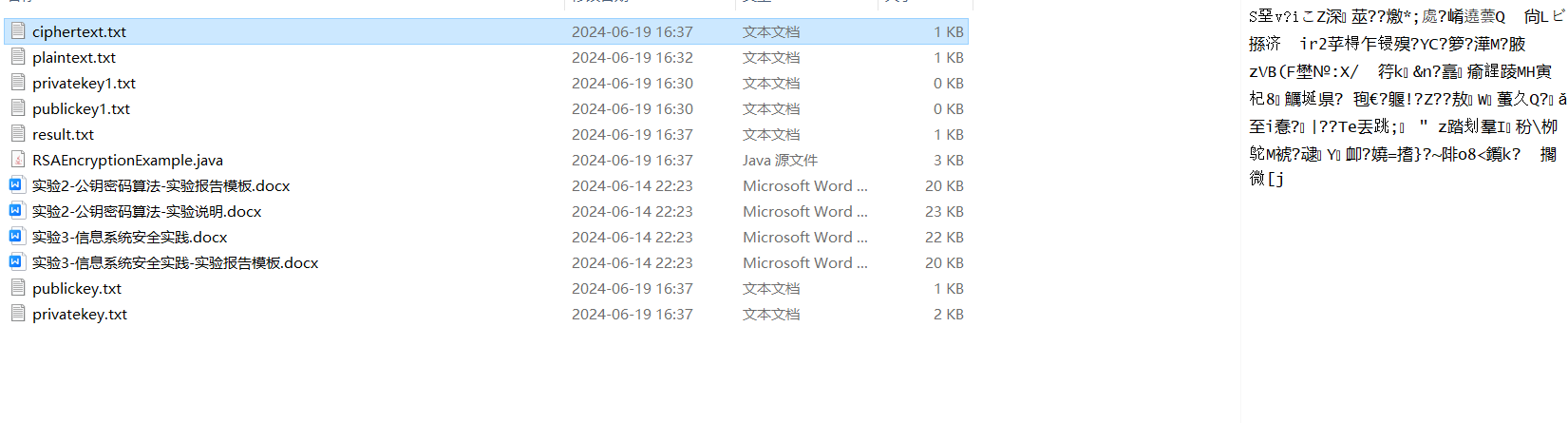
**2.5 基本加解密实验**

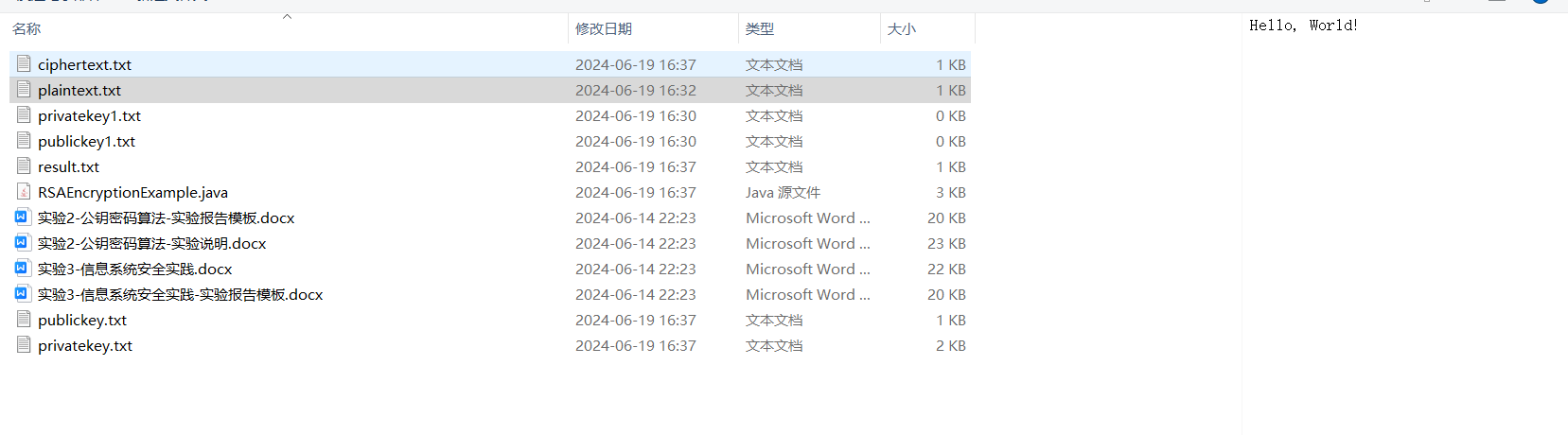
**实验过程说明**

1. 从plaintext.txt读取明文内容。
2. 使用第一对密钥对中的公钥对明文进行加密，将密文存储到ciphertext.txt文件中。
3. 使用第一对密钥对中的私钥对密文进行解密，解密结果存储到result.txt文件中。
4. 比较解密后的文本与原始明文是否一致，验证加密解密过程的正确性。

**结果展示及分析**

****

****

****

****

1. **加密过程**：明文成功加密为密文，密文存储在ciphertext.txt文件中。
2. **解密过程**：密文成功解密为明文，解密结果存储在result.txt文件中，且与原始明文一致。
3. **截图记录**：保存了加密和解密过程的截图，展示了程序运行的各个关键步骤。
4. **结果分析**：实验结果验证了RSA加密和解密过程的正确性，程序能够正确读取公钥和私钥，并进行加解密操作。

**2.6 多对公钥私钥混合实验**

**实验过程说明**

1. 使用第一对密钥对中的公钥对明文进行加密，使用第二对密钥对中的私钥进行解密，观察解密是否成功。
2. 使用第二对密钥对中的公钥对明文进行加密，使用第一对密钥对中的私钥进行解密，观察解密是否成功。
3. 使用第一对密钥对中的私钥对明文进行加密，使用第二对密钥对中的公钥进行解密，观察解密是否成功。
4. 使用第二对密钥对中的私钥对明文进行加密，使用第一对密钥对中的公钥进行解密，观察解密是否成功。
5. **总结与展望**

通过本次实验，我们深入理解了公钥密码算法的基本原理，掌握了RSA算法的具体实现过程。在实验过程中，通过不同密钥对的混合实验，验证了公私钥的配对特性，进一步加深了对非对称加密算法的理解。

未来的工作可以扩展到椭圆曲线加密算法（ECC）的实现与研究，并探讨其在不同应用场景中的优势。此外，结合实际应用中的需求，进一步优化加密算法的效率和安全性。

**参考文献**

1. Whitfield Diffie, Martin E. Hellman. "New Directions in Cryptography". IEEE Transactions on Information Theory, 1976.
2. Ron Rivest, Adi Shamir, Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM, 1978.
3. Victor S. Miller. "Use of Elliptic Curves in Cryptography". Advances in Cryptology — CRYPTO 1985.