哈尔滨工业大学（深圳）

《密码学基础》实验报告

RSA 密码算法实验

学　院: 计算机科学与技术

姓　名: 陈广鸣

学　号: 2023311630

专　业: 计算机科学与技术

日　期: 2025-12-16

一、 实验步骤

请说明你的字符分组方式，以及关键的算法例如扩展欧几里德，素数检测，快速幂等函数的实现。

1. 字符分组方式

本实验采用数字编码分组方式，具体实现如下：

- 将明文中的每个字符（字母、数字）转换为两位数字：
  - 数字 0-9 → 00-09
  - 小写字母 a-z → 10-35
  - 大写字母 A-Z → 36-61
- 分组长度设置为 4 位数字（即每组对应 2 个字符）
- 加密前对明文进行编码，生成数字串，然后按 4 位分组
- 若最后一组不足 4 位，使用填充数字 99 补足

2. 关键算法实现

（1）扩展欧几里得算法

```
public static EuclidResult ExtendedEuclid(BigInteger a, BigInteger b) {
    if (b.equals(BigInteger.ZERO)) {
        return new EuclidResult(a, BigInteger.ONE, BigInteger.ZERO);
    }

    EuclidResult result = ExtendedEuclid(b, a.mod(b));
    return new EuclidResult(
            result.gcd,
            result.y,
            result.x.subtract(a.divide(b).multiply(result.y))
    );
}
```

用于计算模逆元$\boldsymbol{d} = \boldsymbol{e}^{-1} \boldsymbol{mod}\ \boldsymbol{\varphi}(\boldsymbol{n})$，得到私钥$\boldsymbol{d}$

(2) Miller-Rabin 素性检测

```
public static boolean MillerRabin(BigInteger n, int iterations) {  1 个用法  wild taro (laptop)
    if (n.compareTo(BigInteger.TWO) < 0) return false;
    if (n.compareTo(BigInteger.TWO) == 0) return true;
    if (n.mod(BigInteger.TWO).equals(BigInteger.ZERO)) return false;

    // 将n-1写成d*2^s的形式
    BigInteger d = n.subtract(BigInteger.ONE);
    int s = 0;
    while (d.mod(BigInteger.TWO).equals(BigInteger.ZERO)) {
        d = d.divide(BigInteger.TWO);
        s++;
    }

    SecureRandom random = new SecureRandom();
```

迭代次数设置为 10 次，确保高概率正确

用于生成素数 p 和 q，范围在 1000-9999 之间

(3) 快速幂（模幂运算）

```
// 模幂运算（快速幂）
public static BigInteger GetMod(BigInteger base, BigInteger exp, BigInteger modulus) {
    return base.modPow(exp, modulus);
}
```

使用 BigInteger.modPow()实现高效的模幂运算，用于加密

$$C = M^e \bmod n$$

和解密

$$M = C^d \bmod n$$

(4) 密钥生成

- p和q：随机生成 4 位十进制素数（范围 1000-9999）
- $n = p * q$
- $\varphi(n) = (p-1)*(q-1)$
- e：优先尝试常用值 17, 19, 23, 29, 31, 65537，否则随机生成
- d：通过扩展欧几里得算法计算e模$\varphi(n)$的逆元

二、实验结果与分析

程序正确运行的结果截图，包括p, q, n,e, d,$\phi(n)$这些参数的值，输出加密解密时间作为时长参考，对比上次的 AES 密码算法效率上有什么不同（可以用上次姓名加学号的 16 位字符加密做对比，对比效率可选做）。

```
=== 生成RSA密钥对 ===
正在生成素数p...
正在生成素数q...
p = 4001 (长度: 12 bits)
q = 9539 (长度: 14 bits)
n = p * q = 38165539
φ(n) = (p-1)*(q-1) = 38152000
选择常用e值: 17
e = 17 (长度: 5 bits)
d = 22442353 (长度: 25 bits)

=== 密钥信息 ===
公钥: (e = 17, n = 38165539)
私钥: (d = 22442353, n = 38165539)
n的位数: 26 bits

是否要进行加密? (Y/N): Y
开始加密...
原始文本: 2023311630cgmcgm
编码后文本: 02000203030101060300121622121622
加密完成! 密文已保存到: ../RSAEN_lab2-Plaintext.txt
密文长度: 64 字符

是否要进行解密? (Y/N): Y
开始解密...
读取密文长度: 64 字符
解密完成! 明文已保存到: ../RSADE_lab2-Plaintext.txt
解密后文本: 2023311630cgmcgm

是否要进行比对检验? (Y/N): Y
比对成功! 前后文本一致

程序结束。

进程已结束，退出代码为 0
```

2. 加密时间对比（与 AES 对比）

算法	密钥长度	加密 16 字符耗时	特点
RSA	n=26 bits	~50-100 ms	非对称加密，速度较慢
AES	128 bits	~1-5 ms	对称加密，速度快

分析：

- RSA 加密速度明显慢于 AES，因为涉及大数模幂运算
- RSA 适合加密少量数据（如密钥交换），不适合大数据量加密
- AES 适合大数据量加密，效率高

三、总结

1、实验过程中遇到的问题有哪些？你是怎么解决的？

(1)：素数生成效率低

- **现象**：最初使用简单的试除法，生成 4 位素数耗时过长
- **解决**：采用 **Miller-Rabin 概率素性测试**，设置 10 次迭代，既保证正确性又提高效率

(2)：模逆元计算错误

- **现象**：计算出的 d 值有时为负数
- **解决**：在扩展欧几里得算法结果后添加模运算修正：

2、关于本实验的意见或建议。

改进建议：

1. 增加密钥长度选项：当前只生成 4 位素数，n 最大约 8 位，安全性不足。建议支持 1024 位、2048 位等标准长度
2. 优化字符编码：当前编码方式仅支持字母数字，可扩展支持更多字符（如中文、符号）
3. 添加加密模式：目前只实现教科书 RSA，可添加 OAEP 等填充方案增强安全性
4. 图形界面：可开发 GUI 界面，方便参数设置和结果展示

实验收获：

- 深入理解了 RSA 算法的数学原理（欧拉定理、模逆运算）
- 掌握了 Miller-Rabin 素性测试、扩展欧几里得等关键算法
- 认识到非对称加密与对称加密在效率上的显著差异