

# 《应用密码学》实验报告

课程: 应用密码学      实验名称: AES 基本变换

姓名: 杨佳妮      实验日期: 2024.4.15

学号: 2022132006 实验报告日期: 2024.4.15

班级: 信安实验 221

教师评语:	成绩:
签名:	
日期:	

## 一、实验名称

## RSA 算法的实现

## 二、实验环境（详细说明运行的系统、平台及代码等）

1. VC

### 三、实验目的

- (1) 加深对 RSA 算法的理解;
- (2) 加深对平方乘算法和模重复平方方法的理解;
- (3) 加深对模块化设计的理解, 提高编程实践能力

#### 四、实验内容、步骤及结果

## 1. 实验内容

- (1) 完成 RSA 算法加密和解密功能;
- (2) 按照欧几里得扩展算法求, 计算 RSA 私钥;
- (3) 按照平方乘算法和模重复平方法, 分别计算  $a^m \bmod n$ , 完成 RSA 的加密和解密。

## 2. 实验步骤

## 主要函数实现方法介绍

### 1、生成 RSA 私钥的原理如下：

选择两个大素数  $p$  和  $q$ ，并计算它们的乘积  $n=p*q$ 。计算  $n$  的欧拉函数  $\phi(n)=(p-1)*(q-1)$ 。选择一个整数  $e$ ，使得  $1<e<\phi(n)$ ，且  $e$  与  $\phi(n)$ 互素。计算  $e$  关于  $\phi(n)$ 的模反元素  $d$ ，即满足  $ed\equiv 1(\bmod \phi(n))$ 的整数  $d$ 。公钥为 $(n, e)$ ，私钥为 $(n, d)$ ，具体算法：

```
int calculate_private_key_06(int e, int phi_n) {  
    int d = 2;  
    while ((d * e) % phi_n != 1) {  
        d++;  
    }  
    return d;  
}
```

### 2、模重复平方原理

#### 算法原理

模重复平方算法是用来快速计算 $b^n \bmod m$ 的一个算法。

考虑直接计算 $b^n \bmod m$ ，需要 $n-1$ 次乘法，也就是递归计算

$$b^n \equiv (b^{n-1} \bmod m) \cdot b \bmod m.$$

不过，当 $n$ 很大的时候，计算会非常耗时。

现在，考虑将 $n$ 的**二进制**表示 $n = n_0 + n_1 2 + n_2 2^2 + \cdots + n_{k-1} 2^{k-1}$ ，其中 $n_0, n_1, \cdots, n_{k-1}$ 为0或者1。

则：

$$\begin{aligned} b^n &\equiv b^{n_0 + n_1 2 + n_2 2^2 + \cdots + n_{k-1} 2^{k-1}} \bmod m \\ &\equiv b^{n_0} \cdot (b^2)^{n_1} \cdot ((b^2)^2)^{n_2} \cdots (b^{2^{k-1}})^{n_{k-1}} \bmod m \end{aligned}$$

通过观察，我们可以知道，我们需要 $b$ 的偶次幂。设 $b_0 = b$ 的话，下一个偶次幂 $b_1 = b_0^2 \bmod m$ ，以此类推， $b_2 = b_1^2 \bmod m, \cdots, b_{k-1} = b_{k-2}^2 \bmod m$ ，然后代入上面的公式，有：

$$b^n \equiv b_0^{n_0} \cdot b_1^{n_1} \cdot b_2^{n_2} \cdots b_{k-1}^{n_{k-1}} \bmod m.$$

又由于 $n_0, n_1, \cdots, n_{k-1}$ 为0或者1,所以，我们只需要通过不断平方计算出 $b_i$ ，令初始结果为1，然后，如果 $n_i$ 为1则乘上 $b_i \bmod m$ ，否则，不管，继续计算 $b_{i+1}$ 。

可以看出，通过上面的计算，我们只需要最多 $2\lceil \log_2 n \rceil$ 次乘法，和一次 $n$ 的分解。

代码：

```
int square_and_multiply_06(int base, int exponent, int modulus) {  
    long long result = 1;  
    int i = 0;  
    printf("模重复平方\n");  
    while (exponent > 0) {  
        if (exponent % 2 == 1) {
```

```

        result = (result * base) % modulus;
    }
    base = (base * base) % modulus;
    exponent /= 2;
    printf("i=%d,\t%lld\n", i, result);
    i++;
}
return (int)result;
}
}

```

### 3.实验结果

The screenshot shows the VS Code editor with a C program for RSA. The code includes a GCD function, a modular exponentiation function, and a main function that prompts the user for public key components (e, p, q), calculates the modulus n, Euler's totient function φ(n), and the private key d. It then prompts for a message m, performs encryption, and finally decryption to retrieve the original message.

```

C:\> VSCODE-C > C RSA.c > main()
1  #include <stdio.h>
2
3  // 求最大公约数
4  int gcd_06(int a, int b) {
5      if (b == 0)
6          return a;
7      else
8          return gcd_06(b, a % b);
9  }
10
11 // 计算模重复平方
12 int square_and_multiply_06(int base, int exponent) {
13     long long result = 1;
14     int i = 0;
15     printf("模重复平方\n");
16     while (exponent > 0) {
17         // 当 exponent 为奇数时, 更新 result
18         if (exponent % 2 == 1) {
19             result = (result * base) % modulus;
20         }
21         // 每次循环更新 base 和 exponent
22         base = (base * base) % modulus;
23         exponent /= 2;
24     }
25     return result;
26 }
27
28 int main() {
29     int e, p, q;
30     printf("请输入公钥, 两素数: e p q: ");
31     scanf("%d %d %d", &e, &p, &q);
32     long long n = p * q;
33     printf("输出模数 n: %lld\n", n);
34     long long phi_n = (p - 1) * (q - 1);
35     printf("输出模数 n 的欧拉函数: %lld\n", phi_n);
36     int d;
37     printf("输出模数私钥 d: ");
38     while (1) {
39         scanf("%d", &d);
40         if (gcd_06(d, phi_n) == 1) {
41             printf("请输入要加密的明文 m: ");
42             scanf("%d", &m);
43             printf("RSA加密模幂运算底数指数模数: %d %d %d\n", d, e, n);
44             long long ciphertext = square_and_multiply_06(d, e);
45             printf("加密后密文为: %lld\n", ciphertext);
46             printf("RSA解密模幂运算底数 指数 模数: %d %d %d\n", d, e, n);
47             long long plaintext = square_and_multiply_06(ciphertext, d);
48             printf("解密后的明文为: %lld\n", plaintext);
49             printf("请按任意键继续... |");
50             getchar();
51             break;
52         }
53     }
54     return 0;
55 }

```

Terminal Output:

```

请顺序输入公钥, 两素数: e p q: 7 13 17
输出模数 n: 221
输出模数 n 的欧拉函数: 192
输出模数私钥 d: 55
请输入要加密的明文 m: 22
RSA加密模幂运算底数指数模数: 22 7 221
模重复平方
i=0, 22
i=1, 40
i=2, 61
加密后密文为: 61
RSA解密模幂运算底数 指数 模数: 61 55 221
模重复平方
i=0, 61
i=1, 14
i=2, 22
i=3, 22
i=4, 107
i=5, 22
解密后的明文为: 22
请按任意键继续... |

```

### 五、实验中的问题及心得

对于 RSA 算法，在编写程序过程中，核心部分的程序是课程所给示例代码，课后自己也重新扩展完善敲了一遍代码，对 RSA 算法有了更深的了解和掌握，也掌握了 RSA 算法的规则。

## 附件：程序代码

```
#include <stdio.h>

// 求最大公约数
int gcd_06(int a, int b) {
    if (b == 0)
        return a;
    else
        return gcd_06(b, a % b);
}
```

```
// 计算模重复平方
int square_and_multiply_06(int base, int exponent, int modulus) {
    long long result = 1;
    int i = 0;
    printf("模重复平方\n");
    while (exponent > 0) {
        // 当 exponent 为奇数时, 更新 result
        if (exponent % 2 == 1) {
            result = (result * base) % modulus;
        }
        // 每次循环更新 base 和 exponent
        base = (base * base) % modulus;
        exponent /= 2;
    }
```

```
    // 输出当前循环的信息
    printf("i=%d,\t%lld\n", i, result);
    i++;
}
return (int)result;
}
```

```
// 计算私钥 d 值
int calculate_private_key_06(int e, int phi_n) {
    int d = 2;
    while ((d * e) % phi_n != 1) {
        d++;
    }
    return d;
}
```

```
int main() {
    int e, p, q, n, phi_n, d, m, c;
```

```
// 读入数据公钥 e 和两素数 p、q 以及待加密消息 m
printf("请顺序输入公钥，两素数：e p q: ");
scanf("%d %d %d", &e, &p, &q);
```

```
// 计算 n 值
n = p * q;
```

```
// 计算 n 的欧拉函数
phi_n = (p - 1) * (q - 1);
```

```
// 计算私钥 d 值
d = calculate_private_key_06(e, phi_n);
```

```
// 输出模数 n 和 n 的欧拉函数
printf("输出模数 n: %d\n", n);
printf("输出模数 n 的欧拉函数: %d\n", phi_n);
```

```
// 输出模数私钥 d
printf("输出模数私钥 d: %d\n", d);
```

```
// 输入要加密的明文 m
printf("请输入要加密的明文 m: ");
scanf("%d", &m);
```

```
// 使用公钥加密得到密文 c
printf("RSA 加密模幂运算底数指数模数: %d %d %d\n", m, e, n);
c = square_and_multiply_06(m, e, n);
printf("加密后密文为: %d\n", c);
```

```
// 使用私钥解密得到明文 m
printf("RSA 解密模幂运算底数 指数 模数: %d %d %d\n", c, d, n);
m = square_and_multiply_06(c, d, n);
printf("解密后的明文为: %d\n", m);
```

```
return 0;
}
```