南昌大学实验报告

学生姓名：丁俊　　学　　号：8003119100　专业班级：信息安全 193 班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新　　实验日期：2021.11.5　实验成绩：______

一、实验项目名称

程序实现 DES 加密算法

二、实验目的

1、理解 DES 算法的概念及原理

2、理解 DES 算法的加密流程

三、实验基本原理

RSA 算法（公开密钥算法）的原理：

(1). 选择两个大的素数 p 和 q（典型情况下为 1024 位）

(2). 计算 $n = p * q$ 和 $z =（p-1）*（q-1）$.

(3). 选择一个与 z 互素的数，将它称为 d

(4). 找到 e，使其满足 $e*d = 1 \bmod z$

提前计算出这些参数以后，我们就可以开始执行加密了。首先将明文分成块，使得每个明文消息 P 落在间隔 $0*P<n$ 中。为了做到这一点，只要将明文划分成 k 位的块即可，这里 k 是满足 $2^k<n$ 的最大整数。

为了加密一个消息 P，只要计算 $C=P^e(\bmod n)$ 即可。为了解密 C，只要计算 $P=C^d(\bmod n)$即可。可以证明，对于指定范围内的所有 P，加密盒解密互为反函数。为了执行加密，你需要 e 和 n；为了执行解密，你需要 d 和 n。因此，公钥是有（e，n）对组成，而私钥是有（d，n）对组成。

加密算法：$C = Ek(M) = M ^ e \bmod n$

解密算法：$M = Dk(C) = C ^ d \bmod n$

求逆元：利用扩展欧几里得算法，求 a mod n 的逆元，即求一个 b 能够使得 ab mod n = 1 的最小值 b。有一个定理，当 $ax + by = 1$ 且 a，b 互素时解 x 是 a 模 b 的乘法逆元。

求 $a^b \bmod n$ 的值：利用快速幂算法，减少计算的次数，提高程序计算速度，利用取模性质，

即$(a*b) \bmod n = [(a \bmod n) * (b \bmod n)] \bmod n$，减少一些超大数的计算量。

四、主要仪器设备及耗材

Windows 操作系统，DevCpp

五、实验步骤

如下是 c++代码，简单利用 RSA 加解密算法进行简单数据的加密解密。

```cpp
#include <iostream>
#include <bits/stdc++.h>
#define ll long long
using namespace std;

// 8003119100 明文
ll n;
ll p, q, e;
ll d;
// 最大公因数
ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
}

// 判断是否是素数
bool isPrime(ll n) {
    bool flag = false;
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0)
            flag = true;
        if (flag)
            return false; // 不是素数
    }
    if (!flag)
        return true; // 是素数
}

// 求 a^b mod n 的值,利用快速幂
ll ksc(ll a, ll b, ll n) {
    ll ans = 0;

    while (b) {
        if (b & 1)
```

```
            ans = (ans + a) % n;
        a = (a + a) % n;
        b = b >> 1;
    }
    return ans;
}

ll Modular(ll a, ll b, ll n) {
    ll ans = 1, base = a;
    while (b) {
        if (b & 1) {
            // 优化1
            ans = ksc(ans, base, n) % n;
        }
        base = ksc(base, base, n) % n;
        b >>= 1;
    }
    return ans;
}

//求乘法逆元
// ax+by=1 中 x 是 amodb 的乘法逆元
void extgcd(ll a, ll b, ll &d, ll &x, ll &y) {
    if (b == 0) {
        d = a, x = 1, y = 0;
        return;
    }
    extgcd(b, a % b, d, y, x);
    y -= a / b * x;
    return;
}

// 乘法逆元的结果
ll getMod(ll a, ll b) {
    ll d, x, y;
    extgcd(a, b, d, x, y);
    if (d == 1)
        return (x + b) % b;
    else
        return -1;
}

// 加密
ll Encryption(ll value) {
```

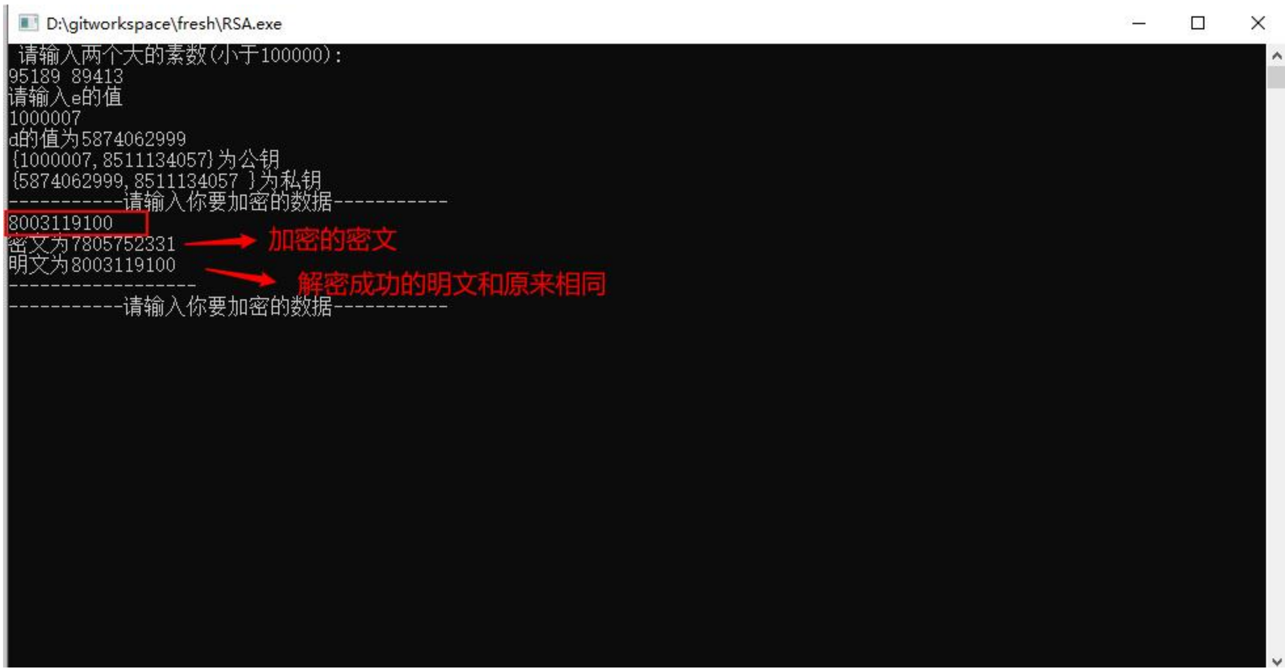
```
    return Modular(value, e, n);
}

// 解密
ll Decryption(ll value) {
    return Modular(value, d, n);
}

int main() {
    while (1) {
        cout << " 请输入两个大的素数(小于100000):" << endl;
        cin >> p >> q;
        if (isPrime(p) && isPrime(q))
            break;
    }
    while (1) {
        cout << "请输入e的值" << endl;
        cin >> e;
        if (gcd(e, (p - 1) * (q - 1)) == 1 && e + 2 < (p - 1) * (q - 1))
            break;
    } // 输入的e作为公钥密码
    n = p * q;
    d = getMod(e, (p - 1) * (q - 1)); // 获得e模(p-1)(q-1)的乘法逆元

    cout << "d的值为" << d << endl;
    cout << "{" << e << "," << n << "}" << "为公钥" << endl;
    cout << "{" << d << "," << n << " }" << "为私钥" << endl;
    // 进行加密操作
    while (1) {
        cout << "-----------请输入你要加密的数据-----------" << endl;
        ll k;
        cin >> k;
        ll m = Encryption(k);
        cout << "密文为" << m << endl;
        cout << "明文为" << Decryption(m) << endl;
        cout << "------------------" << endl;
    }
    return 0;
}
```

六、实验数据及处理结果



在选取 p、q 和加密密钥 e 时，必须保证(p-1)(q-1)=phi_n 是大于你要输入的明文数据的，这样取模的时候才不会导致解密的时候数据恢复错误，比如这里 8511134057>8003119100(我的学号)。

可以看到，由明文加密得到的密文再由解密密钥解密得到的结果和原明文一样，说明算法本身和密钥选取是正确的。

七、思考讨论题或体会或对改进实验的建议

本次实验程序不能达到 RSA 密码的现实要求，即 p、q 的选取要很大(几百位甚至上千位)，可以采用高精度算法来处理那些由计算机不能计算的大数字，不然计算机计算会发生溢出。

八、参考资料

现代密码学第 4 版