南开大学
Nankai University

南　　开　　大　　学

计 算 机 学 院

网络安全技术作业报告

基于 RSA 算法自动分配密钥的加密聊天程序

朱浩泽 1911530

年级：2019 级

专业：计算机科学与技术

班级：计算机科学与技术 2 班

2022 年 4 月 29 日

目录

一、 实验目的

1. 加深对 RSA 算法基本工作原理的理解。
2. 掌握基于 RSA 算法的保密通信系统的基本设计方法。
3. 掌握在 Linux 操作系统实现 RSA 算法的基本编程方法。
4. 了解 Linux 操作系统异步 IO 接口的基本工作原理。

二、 实验内容

1. 要求在 Linux 操作系统中完成基于 RSA 算法的自动分配密钥加密聊天程序的编写。
2. 应用程序保持第三章“基于 DES 加密的 TCP 通信”中示例程序的全部功能，并在此基础上进行扩展，实现密钥自动生成，并基于 RSA 算法进行密钥共享。
3. 要求程序实现全双工通信，并且加密过程对用户完全透明。

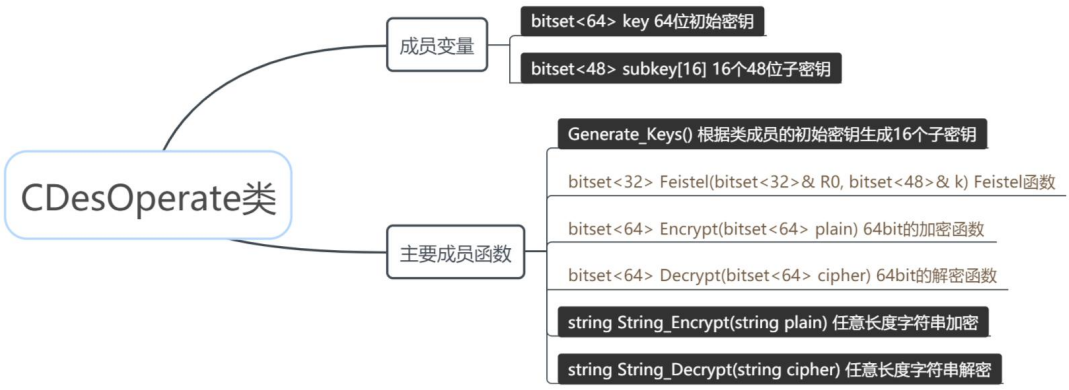
三、 实验步骤及实验结果

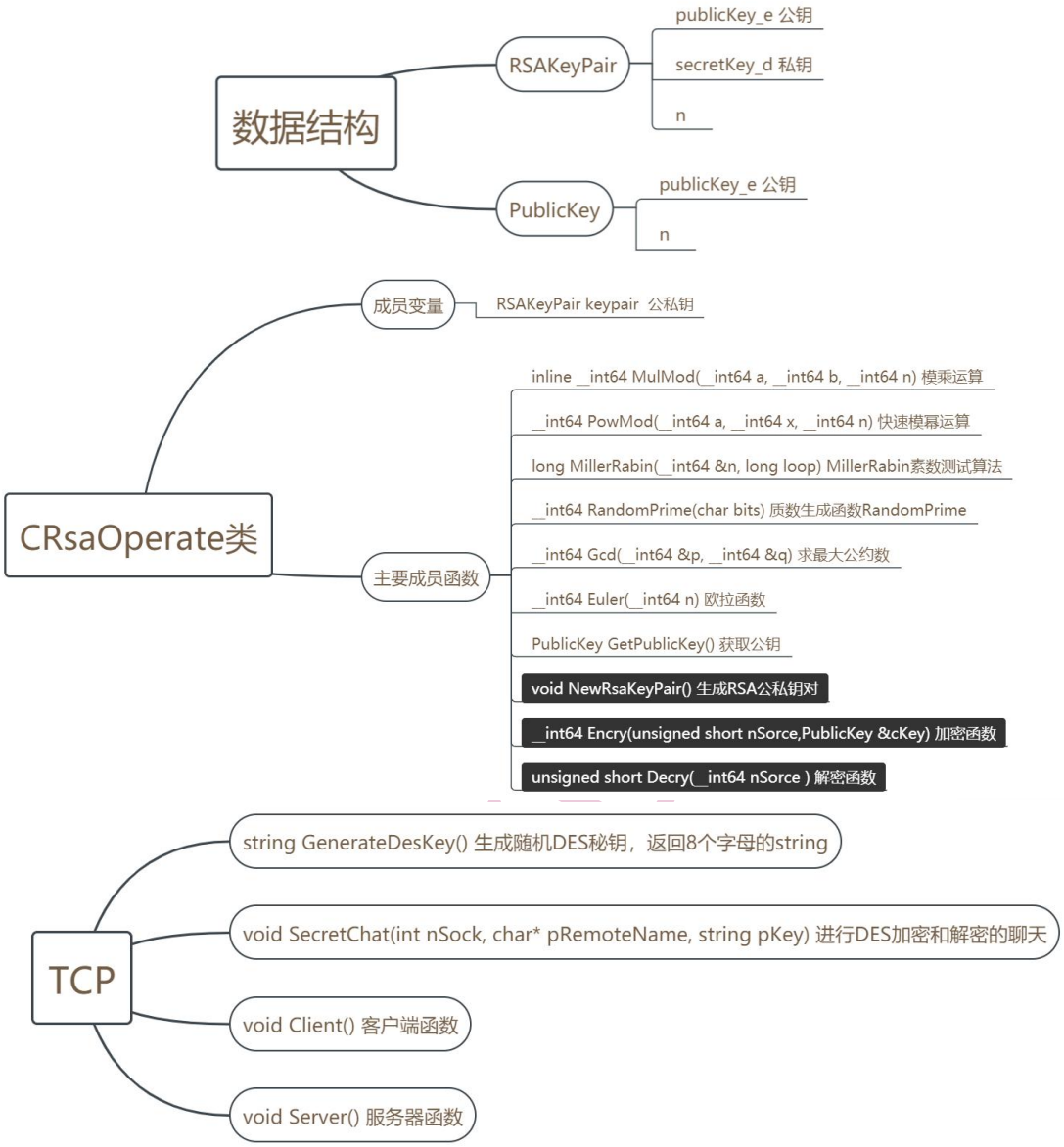
(一) 实验环境

macOS 12.3(基于 unix), C++11, Cmake

(二) 代码结构

CDesOperate.h CDesOperate.cpp 封装了 DES 加密解密的 CDesOperate 类。
CRsaOperate.h CRsaOperate.cpp 封装了 RSA 加密解密的 CRsaOperate 类。
TCP.h TCP.cpp 服务器和客户端的处理函数。
main.cpp 主程序。





（三） 代码实现

1. **封装 DES 加密解密算法的类**

CDesOperate 类较实验一没有发生变化，后文对此类不加赘述。

2. **封装 RSA 加密解密算法的类，并检验其正确性。**

建立两个数据结构，存储 RSA 公私钥对的全部数据

```
struct PublicKey {
   __int64 nE;
   __int64 nN;
};

struct RSAKeyPair {
   __int64 publicKey_e;
   __int64 secretKey_d;
```

```
    __int64 n;
};
```

m_cParament，该变量是存储 RSA 的基本参数，定义如下：

```
struct RsaParam{
    __int64 e;
    __int64 n;
    __int64 d;
    __int64 f;
    __int64 p;
    __int64 q;
    __int64 s;
};
```

编写 RSA.h 和 RSA.cpp 两个文件，在 RSA.h 文件中进行函数与类的声明，在 RSA.cpp 文件中进行定义。
CRsaOperate 类中的内容

- MulMod()
  模乘运算即计算两个数的乘积然后取模
- PowMod()
  快速模幂运算
- MillerRabin()
  MillerRabin 素数测试算法
- RandomPrime()
  质数生成函数 RandomPrime
- Gcd()
  求最大公约数
- Euler()
  欧拉函数：返回小于 n 且与 n 互质的正整数个数
- GetPublicKey()
  获取公钥
- NewRsaKeyPair()
  生成 RSA 公私钥对
- Encry()
  加密函数
- Decry()
  解密函数

```
class CRsaOperate {
public:
    RsaParam m_cParament;
    CRsaOperate();
    static inline __int64 MulMod(__int64 a, unsigned long b, unsigned long n);
```

```
    static __int64 PowMod(__int64 base, __int64 pow, __int64 n);
    static long RabinMillerKnl(__int64 &n);
    static long RabinMiller(__int64 &n, long loop);
    static __int64 RandPrime(char bit);
    static __int64 Gcd(__int64 &p, __int64 &q);
    static __int64 Euclid(__int64 e, __int64 t_n);
    static __int64 Encry(unsigned short nScore, PublicKey &cKey);
    unsigned short Decry(__int64 nScore);
    PublicKey GetPublicKey();
};
```

- 模乘运算
  模乘运算即计算两个数的乘积然后取模

```
inline __int64 CRsaOperate::MulMod(__int64 a, unsigned long b, unsigned long n) {
    return (a % n) * (b % n) % n;
}
```

- 快速模幂运算
  利用快速幂算法进行快速模幂计算，可以在保证数据准确的情况下，快速得到高次幂模运算后的结果。

```
__int64 CRsaOperate::PowMod(__int64 base, __int64 pow, __int64 n) {
    __int64 a = base, b = pow, c = 1;
    while(b){
        while(!(b & 1)){
            b >>= 1;
            a = MulMod(a, a, n);
        }
        b--;
        c = MulMod(a, c, n);
    }
    return c;
}
```

- MillerRabin 素数测试算法
  概率算法：MillerRabin 素数测试算法
  缺省参数 loop 为默认重复测试参数为 100
  Rabin-Miller 算法具有较低的时间复杂度，但是无法保证所得结果一定是正确的，而是以一定的概率得到一个素数，所以需要设定一个安全参数来保证 Rabin-Miller 算法以高概率得到一个素数。

```
long CRsaOperate::RabinMillerKnl(__int64 &n) {
    __int64 a, q, k, v;
    q = n - 1;
    k = 0;
    while(!(q & 1)) {
        ++k;
        q >>= 1;
    }
    a = 2 + rand() % (n - 3);
    v = PowMod(a, q, n);
    if(v == 1) {
```

```
        return 1;
    }
    for(int j = 0; j < k; j++) {
        unsigned int z = 1;
        for(int w = 0; w < j; w++) {
            z *= 2;
        }
        if(PowMod(a, z*q, n) == n - 1)
            return 1;
    }
    return 0;
}


long CRsaOperate::RabinMiller(__int64 &n, long loop=100) {
    for(long i = 0; i < loop ; i++){
        if(!RabinMillerKnl(n)){
            return 0;
        }
    }
    return 1;
}
```

- 质数生成函数 RandomPrime
  保证最高位是 1，再加上一个随机数，保证最低位是 1，即保证是奇数，进行拉宾 - 米勒测试 30 次，全部通过认为是质数

```
__int64 CRsaOperate::RandPrime(char bit) {
    __int64 base;
    do{
        base = (unsigned long)1 << (bit - 1);
        base += rand() % (base);
        base |= 1;
    }
    while(!RabinMiller(base, 30));
    return base;
}
```

- 求最大公约数
  两数相等，最大公约数就是本身；辗转相除法,gcd(a,b)=gcd(b,a-qb)

```
__int64 CRsaOperate::Gcd(__int64 &p, __int64 &q) {
    unsigned long long a = p > q ? p : q;
    unsigned long long b = p < q ? p : q;
    unsigned long long t;
    if( p == q ){
        return p;
    }else{
        while(b){
            a = a % b;
            t = a;
            a = b;
            b = t;
        }
        return a;
```

```
    }
}
```

- 欧拉函数: 返回小于 n 且与 n 互质的正整数个数

```
__int64 CRsaOperate::Euclid(__int64 e, __int64 t_n) {
    unsigned long long Max = 0xffffffffffffffff - t_n;
    unsigned long long i = 1;
    while(1){
        if(((i*t_n)+1)%e == 0){
            return ((i*t_n)+1)/e;
        }
        i++;
        unsigned long long Tmp = (i+1)*t_n;
        if(Tmp > Max){
            return 0;
        }
    }
    return 0;
}
```

- 公私钥对生成

  除了加解密函数外，另外一个重要的内容是 RSA 大素数生成问题。在 CRSASection 类中 RsaGetParam 函数是用来生成 RSA 所有必要的参数，其中主要内容都是生成两个大素数。

```
RsaParam RsaGetParam() {
    RsaParam Rsa = { 0 };
    unsigned long long t;
    Rsa.p = CRsaOperate::RandPrime(16);
    Rsa.q = CRsaOperate::RandPrime(16);
    Rsa.n = Rsa.p * Rsa.q;
    Rsa.f = (Rsa.p - 1) * (Rsa.q - 1);
    do {
        Rsa.e = rand() % Rsa.f;
        Rsa.e |= 1;
    }
    while(CRsaOperate::Gcd(Rsa.e, Rsa.f) != 1);
    Rsa.d = CRsaOperate::Euclid(Rsa.e, Rsa.f);
    Rsa.s = 0;
    t = Rsa.n >> 1;
    while(t) {
        Rsa.s++;
        t >>= 1;
    }
    return Rsa;
}
```

- 获取公钥对

```
PublicKey CRsaOperate::GetPublicKey() {
    PublicKey cTmp;
    cTmp.nE = this -> m_cParament.e;
    cTmp.nN = this -> m_cParament.n;
    return cTmp;
}
```

- RSA 解密函数

```
unsigned short CRsaOperate::Decry(__int64 nScore) {
   unsigned long long nRes = PowMod(nScore, m_cParament.d, m_cParament.n);
   unsigned short *pRes = (unsigned short *)&(nRes);
   if(pRes[1] != 0 || pRes[3] != 0 || pRes[2] != 0) {
      return 0;
   }
   else {
      return pRes[0];
   }
}
```

3. **通讯部分**

服务端将公钥发送给希望与其通话的客户端，然后客户端就可以通过公钥加密生成的 DES 密钥发送给服务端，服务端收到消息后利用自己的私钥进行解密便成功与客户端得到了相同的会话密钥。

四、 概述

(一) 第一节

如图1所示

图 1: Caption

表

N/n\Algo	naive-conv	naive-pool	omp-conv	omp-pool
64/2	0.0167	0.01255	0.04142	0.03799
64/4	0.03599	0.0394	0.0458	0.0421

表 1: 性能测试结果 (4 线程)(单位:ms)

带单元格表格

Cost		To				
		A	*B*	*C*	*D*	*E*
From	*B*	7	0	1	3	8
	C	8	1	0	2	7
	D	8	3	2	0	5

表 2: 结点 C 距离向量表 (无毒性逆转)

(二) 第二节

伪代码

Algorithm 1 初始化 obj 文件信息——对应 MeshSimplify 类中 readfile 函数,Face 类 calMatrix 函数

Input: obj 文件, 顶点、边、面列表

Output: 是否读取成功

function calMatrix($Face$)
    $normal \leftarrow e1 \times e2$
    $normal \leftarrow normal/normal.length$
    $temp[] \leftarrow normal.x, normal.y, normal.z, normal \cdot Face.v1$
    $Matrix[i][j] = temp[i] * temp[j]$
    return $Matrix$
end function
根据 obj 的 v 和 f 区分点面信息, 读取并加入列表
$scale \leftarrow$ 记录点坐标中距离原点最远的分量, 以便后续 OpenGL 进行显示
$ori \leftarrow$ 记录中心点, 便于 OpenGL 显示在中心位置, 避免有的 obj 偏移原点较多
根据三角面片信息, 计算一个面的三条边
计算每个面的矩阵 $\leftarrow calMatrix$
将每个面的矩阵加到各点, 由点维护
return True

代码

```
void ord()
{
   double head,tail,freq,head1,tail1,timess=0; // timers
   init(N);
   QueryPerformanceFrequency((LARGE_INTEGER *)&freq );
   QueryPerformanceCounter((LARGE_INTEGER *)&head);
   for (int i=0; i<NN; i++)
      for (int j=0; j<NN; j++)
         col_sum[i] += (b[j][i]*a[j]);
   QueryPerformanceCounter ((LARGE_INTEGER *)& tail) ;
   cout << "\nordCol" <<(tail-head)*1000.0 / freq<< "ms" << endl;
}
```

逐列访问平凡算法

(三) 第三节

参考文献 [?] [?]

多行公式

$$a + b = a + b \tag{1}$$

$$\frac{a+b}{a-b} \tag{2}$$

行内公式: $\sum_{i=1}^{N}$

超链接 YouTube

带标号枚举

1. 1
2. 2

不带标号枚举

- 1
- 2

切换字体大小

五、 总结