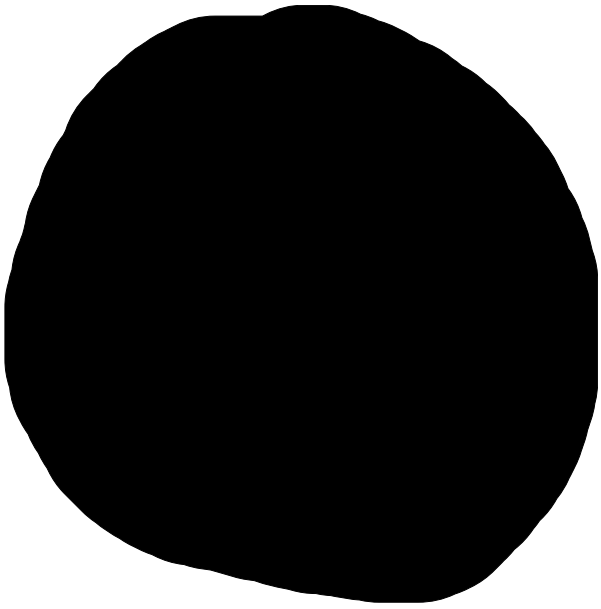
[redacted]

密码学课程设计说明书



学　　号 [redacted]

姓　　名 [redacted]

专　　业 [redacted]

一、题目要求

题目要求包括三个部分：生成密钥、加密文件、解密文件。

2.编程任务

Alice生成RSA密钥：

1. Alice生成一对RSA公钥(n, b)和私钥(p, q, a)，其中随机素数p, q都为512比特长；或都为1024比特长度；两种长度都需支持。

   要求 基于开源代码NTL实现，NTL为密码界流行的高精度代数数论库。不支持其他方式。NTL网址：https://www.shoup.net/ntl/。网站上或下载下来的文档中有详细的使用说明书。

   注意 为了保证通过素性检测的数是素数的概率足够大，需测试足够的次数，如产生512比特的素数，通过测试的数不是素数的概率（即错误概率）接近$1/2^{512}$。

2. Alice把公钥传给Bob。不需要实现网络传输，下同。

Bob加密文件m并发给Alice：

1. 输入文件m，m为任意长。如何把文件转成所需要的格式，请自行决定。CBC模式中，如果m的长度不是分组的倍数时，需要填充，如何填充请自行决定，但解密时需要把填充去掉。
2. 生成128比特的随机数k作为临时的会话密钥。
3. 用Alice的公钥加密k得到$c_1 \leftarrow k^b \bmod n$。
4. 用会话密钥k加密文件m得到c_2，其中加密算法为AES，使用CBC模式。

   要求 AES的列混合运算及其逆运算使用我的课件中提供的快速算法。S-Box使用查表方式实现。AES的测试数据可参考AES的flash动画。

5. 把(c_1, c_2)发给Alice。

Alice解密恢复文件m：

1. 用Alice的RSA私钥解密c_1得到k。
2. 用k解密c_2得到m。
3. 输出m。

在生成密钥时需产生随机数，随机数的生成需要使用一下伪随机数比特生成器：

> 上面所涉及到的随机数（包括随机素数生成，CBC中的IV）请使用ANSI X9.17算法生成，不可使用系统提供的伪随机数函数生成。用于密码用途的随机数需使用专用于密码用途的伪随机数算法生成。
>
> **算法ANSI X9.17 伪随机数比特生成器**
>
> 输入：随机（秘密的）64比特种子s，整数m，两个DES密钥k_1, k_2。
>
> 输出：m个64位伪随机比特串$x_1, x_2, \ldots, x_m$。
>
> 1. 计算中间值$I = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(D)))$，其中$D$表示当前日期/时间的精确表示，DES表示DES加密，DES^{-1}表示DES解密。
> 2. For $i = 1$ to m:
>   1. $x_i \leftarrow DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(I \oplus s)))$
>   2. $s \leftarrow DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(x_i \oplus I)))$
> 3. 返回$x_1, x_2, \ldots, x_m$。

二、设计说明

本程序参考以上题目要求的思路，完成了随机密钥的生成、加密过程和解密过程。

1、 随机密钥的生成

随机密钥的生成需编写一个 ANSI X9.17 伪随机数比特生成器。其中需要 DES 加密和解密。DES 加密的步骤包括 IP 置换、F 函数、S 盒代换等，体现 DES 加密整体思路的代码如下：

```
ZZ DES_encrypt(ZZ input_64, ZZ input_key_64)
{
    ZZ ip_temp;
    ip_temp = IP_permutation(input_64);
    ZZ temp_right, temp_left;
    temp_right = ip_temp & 0xffffffff;//取初始置换IP后的左半部分
    temp_left = (ip_temp - temp_right) >> 32;//取初始置换IP后的右半部分
    for (int i = 0; i < 16; i++)
    {
        temp_left ^= F_function(temp_right, DES_key_schedule(input_key_64, i));
        ZZ temp;
        temp = temp_left;
        temp_left = temp_right;
        temp_right = temp;
    }
    ZZ rounded;//经过16轮运算以后拼接起来的结果
    temp_left <<= 32;//将左半部分进行左移
    rounded = temp_right + temp_left;//相加进行拼接
    rounded = IP_permutation_inverse(rounded);
    return rounded;
}
```

同时，伪随机数生成器还需要 DES 的解密函数，该过程和 DES 加密的过程相同。体现 DES 整体思路的代码如下：

```
ZZ DES_decrypt(ZZ input_64, ZZ input_key_64)
{
    ZZ ip_temp;
    ip_temp = IP_permutation(input_64);
    ZZ temp_right, temp_left;
    temp_right = ip_temp & 0xffffffff;//取初始置换IP后的左半部分
    temp_left = (ip_temp - temp_right) >> 32;//取初始置换IP后的右半部分
    for (int i = 15; i >= 0; i--)
    {
        temp_left ^= F_function(temp_right, DES_key_schedule(input_key_64, i));
        ZZ temp;
        temp = temp_left;
        temp_left = temp_right;
        temp_right = temp;
    }
    ZZ rounded;//经过16轮运算以后拼接起来的结果
    temp_left <<= 32;//将左半部分进行左移
    rounded = temp_right + temp_left;//相加进行拼接
    rounded = IP_permutation_inverse(rounded);
    return rounded;
}
```

最后完成生成伪随机数的随机数生成器：

```
ZZ ANSI(int bits)
{
    srand(unsigned(time(NULL)));
    ZZ key1(rand());
    ZZ key2(rand());
    ZZ input_D;
    ZZ res(0);
    ZZ ans(1);
    int m = 8;

    for (int i = 0; i < bits; i++)
    {
        ans *= 2;
    }
    //cout << ans << endl;
    res = 0;
    for (int i = 0; i < (bits / 64); i++)
    {
        input_D = ZZ(rand());
        res += random_generate_mediate(input_D, key1, key2);//max=9223
        if (i != (bits / 64 - 1))
        {
            for (int j = 0; j < 64; j++)
                res *= 2;
        }
    }
    return res;
}
```

2、 加密过程

加密有两个步骤，先用密钥对文件使用 AES 加密，再对密钥进行 RSA 加密。
RSA 加解密中，在进行寻找乘法逆元和平方计算的时候，需要使用扩展的欧几里得算法和平方乘算法。
扩展的欧几里得算法代码如下：

```
ZZ exgcd(ZZ a, ZZ b, ZZ &x, ZZ &y)
{
    ZZ num, term;
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    num = exgcd(b, a % b, x, y);
    term = x;
    x = y;
    y = term - a / b * (y);
    return num;
}
```

```
ZZ find_a(ZZ a, ZZ p, ZZ q)
{
    ZZ phi_n(0);
    phi_n = (p - 1)*(q - 1);
    ZZ b;
    b = phi_n;
    ZZ x, y;
    exgcd(a, b, x, y);
    x = x % b;
    if (x <= 0)
        x = x + b;
    return x;
}
```

平方乘算法如下：

```
ZZ pow_mod(ZZ a, ZZ b, ZZ mod)
{
    ZZ result(1);
    while (b != 0)
    {
        if ((b & 1) != 0)
        {
            result = (result*a) % mod;
        }
        a = (a*a) % mod;
        b >>= 1;
    }
    return result;
}
```

AES 加密需要进行 AddRoundKey、SubBytes、MixColumns 和 ShiftRows 操作，体现 AES 整

体思路的代码如下（其中涉及 ZZ 大整数和矩阵之间的转化）：

```
ZZ AES_encrypt(ZZ key, ZZ state)
{
    AES_key_schedule(key);
    state = AddRoundKey(state, key);
    for (int i = 1; i <= 10; i++)
    {
        int mat[4][4] = { 0 };
        state = SubBytes(state);
        ZZ_to_matrix(state, mat);
        ShiftRows(mat);
        if (i != 10)
            MixColumns(mat);
        state = matrix_to_ZZ(mat);
        state = AddRoundKey(state, keys[i]);
    }
    return state;
}
```

3、 解密过程

AES 解密和加密用到的基本步骤相同，只是需要进行逆过程。体现整体思路的代码如下：

```
ZZ AES_decrypt(ZZ state)
{
    state = AddRoundKey(state, keys[10]);//无问题
    int mat[4][4] = { 0 };
    for (int i = 9; i >= 1; i--)
    {
        ZZ_to_matrix(state, mat);//将state转化为矩阵
        ShiftRows_inv(mat);//无问题
        state = matrix_to_ZZ(mat);//将矩阵转化为ZZ
        state = SubBytes_inv(state);
        state = AddRoundKey(state, keys[i]);//无问题
        ZZ_to_matrix(state, mat);//将state转化为矩阵
        MixColumns_inv(mat);
        state = matrix_to_ZZ(mat);
    }
    ZZ_to_matrix(state, mat);
    ShiftRows_inv(mat);

    state = matrix_to_ZZ(mat);
    state = SubBytes_inv(state);
    state = AddRoundKey(state, keys[0]);

    return state;
}
```

三、使用说明

1、 输入

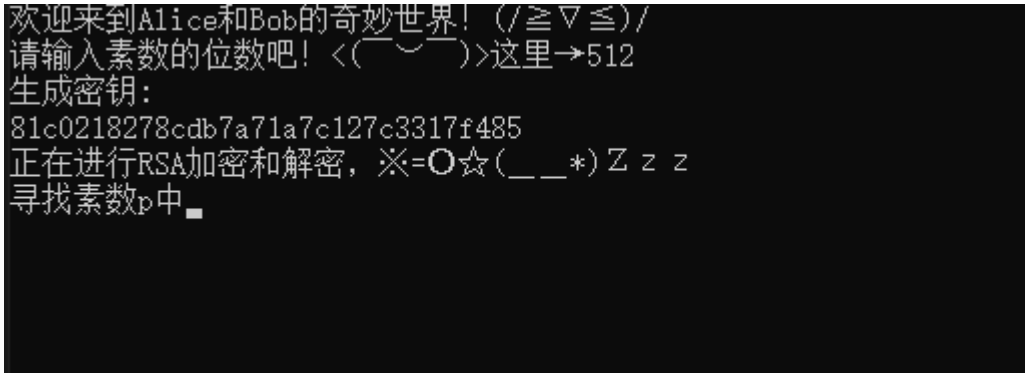
通过文件输入，文件是同目录下的文本文件 plaintext.txt。若需修改加密内容，则只需修改该文件中的内容。（需要保证可执行文件同目录下有文本文件 plaintext.txt，否则会无法打开文件）。
控制台还会请求用户输入生成素数的位数。

2、 输出

生成的密钥和素数会输出到控制台上，生成的密文和破解后的明文分别输出在同目录下文本文件 ciphertext.txt 和 cracked.txt 中，在程序执行完毕以后查看。

3、 具体说明

该程序的随机素数生成适用于所有 64 倍位数的素数。因此，在程序开始时，会要求用户输入随机素数的位数。用户需输入 64 的倍数，若没有输入 64 位倍数，则程序会默认将素数的位数更换为 64 的倍数。
输入后会自动生成素数 p，当控制台显示寻找素数中并光标闪烁，则说明正在生成随机素数 p（需要一定时间）。



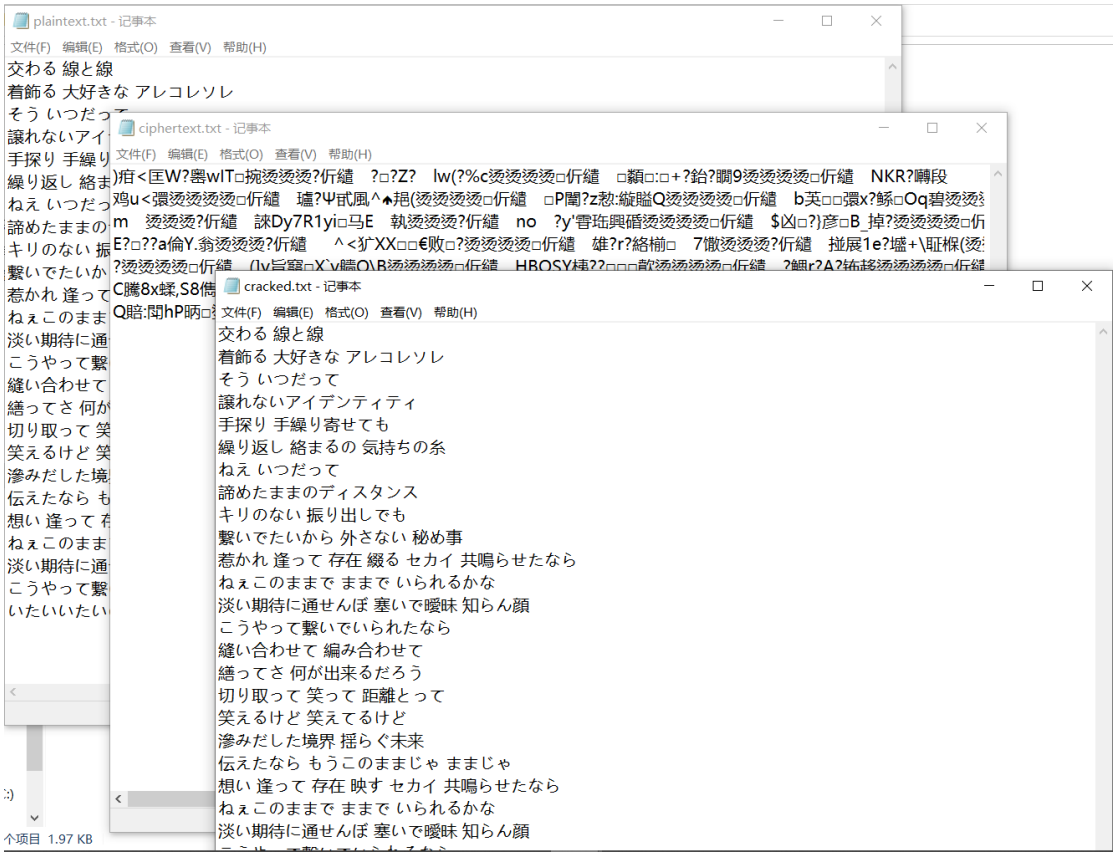
在生成随机素数 p 过后会输出 p，并同理产生随机素数 q（同样需要一定时间）。

生成随机素数 q 过后会继续生成 RSA 密钥 a 和 b，并输出。

```
欢迎来到Alice和Bob的奇妙世界！(/≧▽≦)/
请输入素数的位数吧！<(￣ ￣)>这里→512
生成密钥:
e546586033a43d6ac4d042739819e126
正在进行RSA加密和解密，※=O☆(__*)Z z z
寻找素数p中

素数p:
dae5689b4bcb1ea80a28c9fa4c55a9b920bdbc002caca1d3d622d8595474512fb0f93285ba1a156ccfe55956b4a3f03eb086ddd50b2707da2e0446ea
86b3737f
寻找素数q中



素数q:
ca6311c0c94d2342f0d33d86c1e44817d2e47e92aa1c0f3e231132927a742e545c305f26d76735ed5da41d8c133ffc752c5f6ee5d9d0860b4a428f6e
624dc1a3
生成b中
b:0bba4b35475d79123
生成a中
a:24a5be018091d2b7b
加密后的密钥:
859521fdad0377fc8846665a4fd4a57fb5e7cf7e8a080d3c94330ab65a2b8299812f17d8e0947bc5918be83233af2c625b7f6776297acbe17b46ea23
6c4adcc3
解密后的密钥:
e546586033a43d6ac4d042739819e126
正在用该密钥进行AES加密和解密，请稍后ZZzz…(。-ω-)

AES加解密成功！o(〃'▽'〃)o
可前往ciphertext.txt和cracked.txt查看密文和解密后的明文哟(/ω＼*)…… (/ω・＼*)
```

接着进行加密和解密，生成的密文和破解之后的明文可在同目录下的 ciphertext.txt 和 cracked.txt 中查看。

输出结果如下：



可以通过修改 plaintext.txt 中的内容来对不同的内容进行加密和解密。