RSA 实现

一、算法描述

①随机选择两个大素数 p 和 q，计算 n=p·q，以及 φ(n)=(p-1)·(q-1)；

②选择 e=65537，如果不满足(e, φ(n))=1，则选择一个随机整数(e, φ(n))=1。

③求出私钥 d，使得 e·d=1(mod φ(n))。

④加密时，明文为 M，密文 $C=M^e \bmod n$；

⑤解密时，密文为 C，解密后明文 M' = $C^d \bmod n$

二、分步实现

1.产生指定位数的大随机数，如 p 的 bits=524 位，q 的 bits=500 位，这样后面可以满足 n 大致为 1024 位（可能为 1023 位）。

①产生位数为 bits 的大数 bignum 的函数为 int genBN(BN result, int bits)，利用 sha-1 结果产生 bits 个二进制位的大数 result;

②产生方法为，对于从低到高每一位（一位包含了 32 个二进制位），调用 void writerand(char *addr)函数，往 addr 中写一个随机数，然后调用之前写的 SHA-1 函数，计算 addr 出随机数文件的哈希值，然后取 32 个二进制位(8 个十六进制位，也就是 8 个哈希结果的字符)。

③最高位(最高的可能有 1-32 个二进制位)需要特殊处理，因为最高如果是要 32 位，产生的哈希值第一位如果是 A 以下，如 0-9，那最高就没有 32 位，得到的大数会小于 bits。这种情况下，特殊化处理，多次产生直到产生的是所要的为止。其它时候可以左右移位来满足最高位的需求。

2.产生指定位数的素数

①由 int findprime(BN a, int bits)函数实现，该函数调用了 genBN(BN result, int bits)函数产生随机数，调用了 int fermat_b(BN a)费马检测函数对大数进行素性检测，同时它利用了 exclu()函数预先产生的前几十个素数的乘积结果，利用了 gcd_b(BN a, BN b, BN & result)函数求公因子。同时需要用到 SETONEBIT_B(BN num, uint32_t u)函数设置大数 num 为一个 uint32_t 的数 u。用到了加法函数 add_b(BN a, BN b, BN sum)。

②过程是：

(1)首先读取前 20 个素数的乘积 fac，第 20 个素数存在 temp3 中；

(2)随机产生满足位数要求的大数 bignum;

(3)求公因子 gcd_b(fac,bignum,gcd)，如果 gcd=1，就拿这个大数去做费马检测步骤(7)。如果 gcd 比第 20 个素数还要大，或者很不幸这个大数是偶数 gcd=2 或者 gcd 比 temp3 大，就回到(2)；

(4)对大数进行微调，初始化微调变量 linshi=2，循环变量 i=0;

(5)大数 bignum 加上 linshi，得到调整后的 adjnum，对 adjnum 和 fac 求最大公因子 gcd。

(6)如果 gcd 为 1，就进行费马检测，步骤(7)，linshi 设为 2。如果 gcd 不是 1，且 gcd<temp3，则 linshi=linshi*gcd，循环变量 i++，回到(5)；否则重新生成一个，回到(2)。

(7)如果费马检测结果为 1，则判定产生的是素数，否则回到(2)。

3.公钥 e 的选取

默认选择 e=65537,但是计算(e, φ(n))可能不为 1，这种情况下，就随机选择一个 17 位的 e，使用 findprime()，选择好以后存在 e 的路径中。

4.d 的产生，求逆

①inv_b(BN a, BN n, BN & x)函数实现，该函数调用了 gcd_b(BN a, BN b, BN & result)函数，先求公因子判断是否存在逆元，然后才求逆。

②求逆时，对于 inv_b(a,n,x)，初始化 u=1,g=a,v1=0,v3=n.

③用 div_l(g,v3,q,t3)计算带余除法,g=q·v3+t3，令 t1=u-q·v1 (mod n)，u=v1，g=v3，v1=t1，v3=t3。

④如果 v3=0，则把 u 赋值给 x，作为结果，否则回到步骤②。

5.加密过程为模幂运算 $C=M^e \bmod n$

①使用模平方算法，配合调用 modmul()模乘函数；

②对指数 e 进行二进制展开，初始化 a=1，b=M，m=n。

③对于 e 的二进制展开，从低到高一位位来判断，b=b*b (mod m)，如果这一位为 1，则 a=a*b(mod n)，否则 a=a；

④最后 e 的二进制位都算完了，a 就是结果。

6.解密时因为已知 p 和 q，可以使用中国剩余定理加速解密，加速以后的时间是加速以前的一半

①中国剩余定理在 void crt_b(BN a, BN b, BN p, BN q, BN & result)中实现。

②先计算 b1 和 b2：$a^b = b1 \bmod p$　　$a^b = b2 \bmod q$

③得到方程组　　x=b1 mod p

　　x=b2 mod q

④求逆然后得到结果：

m1=p;m2=q　m=m1*m2;

M1=m2=q　M2=m1=p

M1*M1'=1 mod p　　M2*M2'=1　mod q

result= b1*M1'*M1 + b2* M2'*M2　mod(m)

三、实验结果

1.加密解密过程，用之前产生好的 p 和 q，设置的 e=65537，加密时间明显比解密时间短，其中明文是某个文件的哈希值。

```
是否要产生私钥p q? 输入1产生，输入0不产生
0
是否要改变默认文件路径? 输入1改变，输入0不改变
p 存放在myp4.txt
q 存放在myq4.txt
公钥e存放在e.txt
明文存放在plain.txt
密文存放在cipher.txt
解密文件存放在decry.txt
0
p is  DB252597E8E42B738A0A868CD1220A09BB06E431F424731F9929FC631B159CA4459515B9D2E28A25C475423B187031F8DC3E3F05EFFA2F470DFFE0D1F098E40C2F9
q is  8A3AF3C87CA64615B4252FD5700EE976D2E6FB8883A092EF93C1BC89FD5B7882B209A6DA5FE7A42A653D5F315C86F8E541AFAD74B78A338331FDD6677D07D
初始e is  10001
n is  76547D5E4A5EA82759F0AD1AA3678EE21910DFFCD470D8D25241634D192039738264F5F11F4EC737C576C10D7814ABAB2DCA54C4D843F5118942F03C6E53A444E7AF
370F12BD0751566C7C2FC986173DF35030FF17D2089F756AF9FB1323A44D2FD3CE16333212AB97327C75E76D4700FE1F48023F18BFD2E6DE50FDEBC88395
p-1 is  DB252597E8E42B738A0A868CD1220A09BB06E431F424731F9929FC631B159CA4459515B9D2E28A25C475423B187031F8DC3E3F05EFFA2F470DFFE0D1F098E40C2F
8
q-1 is  8A3AF3C87CA64615B4252FD5700EE976D2E6FB8883A092EF93C1BC89FD5B7882B209A6DA5FE7A42A653D5F315C86F8E541AFAD74B78A338331FDD6677D07C
Φ(n) is  76547D5E4A5EA82759F0AD1AA3678EE21910DFFCD470D8D25241634D192039738264F5F11F4EC737C576C10D7814ABAB2DCA54C4D843F5118942F03C6E539692
954D14D1937E044C4CA86CCAAB8E7A9EED9FE34D1817D503A6AB8C2DF0B98A3C564D104D9CE9B7EA00687718F137F0CD8BDACE07C4D8D64FB09E1216F70FF020
d is  2FB963D6E3A7C9AA155C3C74C1F2C26DEE2E20289885EAECE8BFF68CE1D55D567900BF30C0AFC529793AAC22028C53B0768764290010686EA15554F0A302C4A726C
E438A283F0B7D53318ADA50391F10765B50722F079B93F3F710A164CBEB4171A7DB8CFD0739CAD0D74FB5A4E759071178E8113A360CC432F7D9C4588E81
明文是 is  4EA72B9F99A196581A8ED410DC3016FC74130B26
加密耗时27 ms
密文是 is  2F4E66961E1584A9D0DB8719E43E84B9AF924E0691F6B6890502B1B09149B4C5DD8FBFA564D555CC75A8BA194641C53901FE6BFD48D96BB6D215189CE3FD881
7C5C24ADE926B1286CCDB00C02D3D5334B59DDCE1BDBAA8EBA52B705CC0F19586CC10B95444E3048A311715291CD092A45FFCBE9A20B87A8308B353B2CF44DF93

解密密耗时527 ms
解密为 is  4EA72B9F99A196581A8ED410DC3016FC74130B26

明文为4EA72B9F99A196581A8ED410DC3016FC74130B26
解密为4EA72B9F99A196581A8ED410DC3016FC74130B26
```

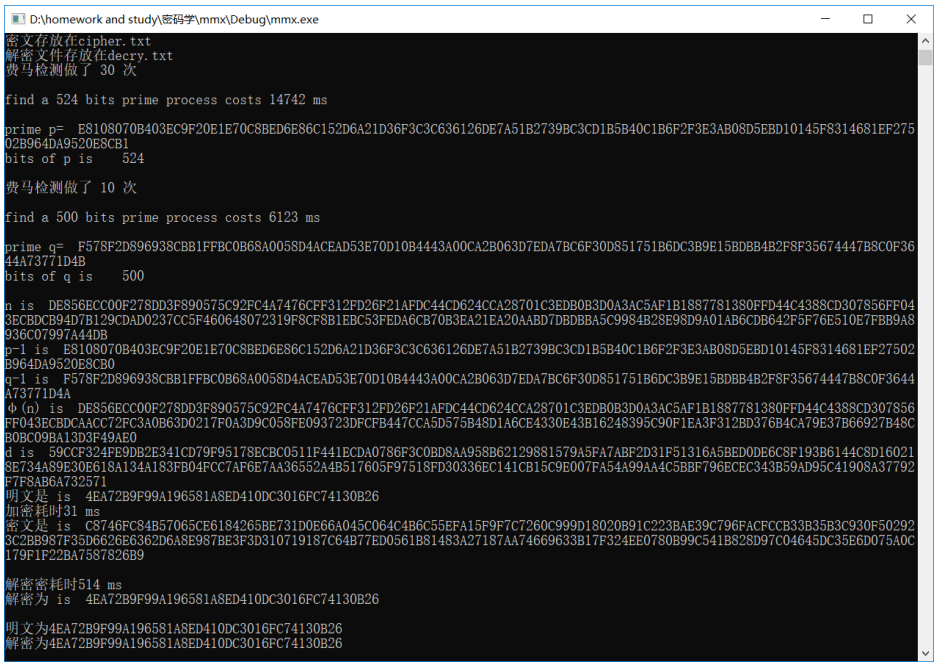
2. 当默认的 e=65537 与 φ(n)不互素时，会随机产一个 17 位的素数，这里产生的为 1B011。

```
是否要产生私钥p q? 输入1产生，输入0不产生
0
是否要改变默认文件路径? 输入1改变，输入0不改变
p 存放在myp4.txt
q 存放在myq4.txt
公钥e存放在e.txt
明文存放在plain.txt
密文存放在cipher.txt
解密文件存放在decry.txt
0
p is  DB252597E8E42B738A0A868CD1220A09BB06E431F424731F9929FC631B159CA4459515B9D2E28A25C475423B187031F8DC3E3F05EFFA2F470DFFE0D1F098E40C2F9
q is  8A3AF3C87CA64615B4252FD5700EE976D2E6FB8883A092EF93C1BC89FD5B7882B209A6DA5FE7A42A653D5F315C86F8E541AFAD74B78A338331FDD6677D07D
初始e is  2711
n is  76547D5E4A5EA82759F0AD1AA3678EE21910DFFCD470D8D25241634D192039738264F5F11F4EC737C576C10D7814ABAB2DCA54C4D843F5118942F03C6E53A444E7AF370F12BD0751
566C7C2FC986173DF35030FF17D2089F756AF9FB1323A44D2FD3CE16333212AB97327C75E76D4700FE1F48023F18BFD2E6DE50FDEBC88395
p-1 is  DB252597E8E42B738A0A868CD1220A09BB06E431F424731F9929FC631B159CA4459515B9D2E28A25C475423B187031F8DC3E3F05EFFA2F470DFFE0D1F098E40C2F8
q-1 is  8A3AF3C87CA64615B4252FD5700EE976D2E6FB8883A092EF93C1BC89FD5B7882B209A6DA5FE7A42A653D5F315C86F8E541AFAD74B78A338331FDD6677D07C
φ(n) is  76547D5E4A5EA82759F0AD1AA3678EE21910DFFCD470D8D25241634D192039738264F5F11F4EC737C576C10D7814ABAB2DCA54C4D843F5118942F03C6E539692954D14D1937E
044C4CA86CCAAB8E7A9EED9FE34D1817D503A6AB8C2DF0B98A3C564D104D9CE9B7EA00687718F137F0CD8BDACE07C4D8D64FB09E1216F70FF020
费马检测做了 1 次
之前公钥e与φ(n)不互素，修改以后的e is  1B011
d is  57CB6AE57D54FC2ADED3C21EE4620EE7C673AE2DC19A4990666083838751CF0290274FACED251FDEDE833577C850B8C36431DF7EB38DFB57943DDE52D3B7A5827D9D2FBD5FF378511
C56D1EB1FB628424A1EB09E621116232172AA3193596E8OD8A4DD8641D7C41255F19C6F4ACBDA4ED390FB0C2A6DA35F6AB2A0B3DCB274F31
明文是 is  4EA72B9F99A196581A8ED410DC3016FC74130B26
加密耗时30 ms
密文是 is  345AEA8A81B8963EA77D1416B8CBD4161C9D42A30AD22B54F2F11F5F16351DD1A5C3F1D4A19D938858AF357394342CF7872EF06B4934382749C0705002EFBF2202EB382D549
D2C1CCD4D5E75D1DC1CEA4FCC4BC2E326A317B6ADCA1871882A79B5C9C4AF9D633117FD38570454117A8DF52D27C1AD66495D8CAD8192AF826FDC

解密密耗时518 ms
解密为 is  4EA72B9F99A196581A8ED410DC3016FC74130B26

明文为4EA72B9F99A196581A8ED410DC3016FC74130B26
解密为4EA72B9F99A196581A8ED410DC3016FC74130B26
```

3. 完整的从产生密钥、加密、解密的过程如下，时间比较长，尤其产生密钥，一般 500 位的要平均 10 秒左右



4. 因为比较难比较到底加解密是否正确，可以调用 SHA-1 函数来验证，这时 SHA-1 每次调用必须对初始化向量赋予初值，和之前的 mysha1()用来随机寻找素数时的每调用一次都有一个状态时完全不一样，之前那个状态要是每次都是定值，寻找素数很慢，而这里要验证，所以必须初值每次都固定。

```
int checkresult(char * plainpath, char * decrypath)
{
    char plainhash[81] = { 0 };
    char decryhash[81] = { 0 };

    SHA1(plainpath, plainhash);
    SHA1(decrypath, decryhash);

    if (strcmp(plainhash, decryhash) != 0)
        return 0;
    else
        return 1;
}
```

四、调用说明

①大数类型为BN，使用时先声明变量

```
BN p, q, n, eula, e, d;
BN p_1, q_1;
BN plain;//明文
BN cry;//加密以后
BN dec;//解密以后
```

②为了防止错误，需要对使用 memset 它们清零，虽然之后每个函数入口都会做清零，但是谨慎起见还是清零为好。

```
memset(p, 0, sizeof(p));
```

③如果要产生P和Q，调用genpq()产生大数并写入文件。writebn()是把大数写到文件中，readbn是从文件中读出大数。

```
genpq(p_path, q_path);
SETONEBIT_B(e, 10001U);
writebn(e_path, e);
readbn(p, p_path);
readbn(q, q_path);
readbn(e, e_path);
```

④mul(p, q, n)计算 n，用 subuint_b(p, 1U, p_1)计算 p-1，用 subuint_b(q, 1U, q_1)计算 q-1，用 mul(p_1, q_1, eula)计算 φ(n)，然后判断 e 是否与 φ(n)互素。

```
gcd_b(e, eula, temp3);
if (cmp_b(temp3, ONE_BN) != 0) {
    while (cmp_b(temp3, ONE_BN) != 0)//如果不互素！！！悲剧了
    {
        findprime(e, 17);
        gcd_b(e, eula, temp3);
    }
    cout << "之前公钥e与φ(n)不互素，修改以后的e is  " << bn2str(e) << endl;
    writebn("e.txt", e);
}
```

⑤inv_b(e,eula, d)获取私钥 d，然后用 readbn(plain, plain_path)获取明文，用 modexp_b(plain, e,n, cry)对明文加密，使用中国剩余定理 crt_b(cry, d, p, q, dec)进行解密。

⑥调用 checkresult(plain_path, decry_path)对明文和解密后文件进行对比校验。

五、总结

完整调用一次下来，其实实际主要还是花了在产生素数上，时间和做费马检测的次数成正比，如果费马检测次数上升很快，时间就会花的很多，如果有其它办法有效避免素性检测次数，时间也可以提升。

时间花的最多的应该是除法，因为其它太多函数调用除法了，而且是多次调用。而除法是用简单的类似二进制展开然后一位位相减的办法，所以除法不快，总的速度就上不去。如果使用 SRT 方式做除法，预测然后纠正的办法，除法循环次数就不会是二进制的次数，而是基数的次数，时间花的会比较少。SRT 的除法不太会实现。。。。。

文件夹里面没有附带.exe 文件，不知道为什么，现场编译 VS 的时候，就能找到素数，如果把.exe 拷贝到其它地方，文件名设置也是对的，就是跑不出来。。。编译时 bignum.h 和 rsa.cpp 要在同一个文件夹下。如果 VS 编译开优化，耗费时间是这个的 1/5 左右。

```
plain.txt
请输入密文存放路径
cipher.txt
请输入解密后存放路径
decry.txt
设置以后:

p 存放在p.txt
q 存放在q.txt
e 存放在e.txt
明文存放在plain.txt
密文存放在cipher.txt
解密文件存放在decry.txt
费马检测做了 6 次

find a 524 bits prime process costs 3133 ms

prime p=  8134921228047DA6F7316FACA051CBF9D81AA4D7002BA33549E4AB898435B45567DC33EA814DC1D60BD728F6DF1DB404D3A7569CF78AFABE9D85CDBCDD3CF6D354D
bits of p is    524

费马检测做了 49 次

find a 500 bits prime process costs 17255 ms

prime q=  904DD8D5801F6FAC9D53CC80D51DD426DF85B522C3F6FA5A2B53D63FB11D9A1CD1737DAD86D7A5B4AA400262C818CC7EDCCB57005149E56BA5A8602870C71
bits of q is    500

n is  48D4DC6A429F91AA7A4A38F004734C792E9C4E2FE87C417E1A47FDB6F8CB284B75932B115450CFCAD3FF8F847E1C137CB7CE463BABB4DEB7F881F8540C5C473AD27168B10923799F5DED262FFC
5CE56421247CC25625DB91C37D891DB7ADC276902AAA3CB369B1B4D7130160109261F32BA3738F78BA8EF20CD5FD87634F22FD
p-1 is  8134921228047DA6F7316FACA051CBF9D81AA4D7002BA33549E4AB898435B45567DC33EA814DC1D60BD728F6DF1DB404D3A7569CF78AFABE9D85CDBCDD3CF6D354C
q-1 is  904DD8D5801F6FAC9D53CC80D51DD426DF85B522C3F6FA5A2B53D63FB11D9A1CD1737DAD86D7A5B4AA400262C818CC7EDCCB57005149E56BA5A8602870C70
φ(n) is  48D4DC6A429F91AA7A4A38F004734C792E9C4E2FE87C417E1A47FDB6F8CB284B75932B115450CFCAD3FF8F847E1C137CB7CE463BABB4DEB7F881F8540C5C3F278947415333F108354C2886
EE178FF6053469146449C64783D3223B9CF86D5A1F2B1EEAEFFC73E37D0938EF6E0F25932BEE4BD7615909907B593FD52D915AE140
d is  2DFB302842886CB47AAACD59C3E424A8C287CC338B125EBE8059C90EFEE7EE2E0D9D576BC69975D9C59B05B38287EF396A990FBF96F923BCDEAD700F62230C265292C2EDDAC20CC24C5753C748
C06847C8E13FE3C77857D8B76A9203FB2AF4453C0390F69172C47359CE4AB540BE57A2A03B34345168C7757A6EE34A01CF671
明文是 is  4EA72B9F99A196581A8ED410DC3016FC74130B26
加密耗时31 ms
密文是 is  2FBDCFEA49FD0B02B7EAD3CF6BF8AEB015D3EFBD6993D0041B8BEE9F5E75C2C4E2D4E571CD6099CE91C31D455488667B653968F34E1A0053A6CD2ED603F053651C6F4E4A020865DC60188
4DF213B30283973F92AD611C3BCC5B289B8CEDF7D91BFAB1B6B43FB7C994E1AA1C0ADDCCB74A872351D3D43B6FB06145ECC7986F27F

解密密耗时521 ms
解密为 is  4EA72B9F99A196581A8ED410DC3016FC74130B26

明文为4EA72B9F99A196581A8ED410DC3016FC74130B26
解密为4EA72B9F99A196581A8ED410DC3016FC74130B26

加解密正确!
```

六、参考文献

实现时，格式一开始参考了《密码学 C/C++语言实现》中的格式，经典的算法参照《信息安全数学基础》及《密码学 C/C++语言实现》进行了实现。

[1]陈恭亮.信息安全数学基础(第2版)[M].北京：清华大学出版社.2014：7-8，；20-37,76-78,80-82,97-99,198-201.

[2]迈克尔·威尔森巴赫.密码学C/C++语言实现(第2版)[M].北京：机械工业出版社.2016：7-9,12-17,21-23,43-46,78-79,112-113.