实验 2 RSA 密码算法

姓名：___________宗晴____________学号：_________200110513_________

一、 运行截图

```
lab2
D:\conda\envs\lab2\python.exe E:/密码学/实验二/200110513_宗晴_密码学基础_实验2/lab2.py
p =  249199
q =  569659
n =  141958453141
e =  39587042563
ϕ(n) =  141957634284
加密后的密文为： [25947051825, 79614854646, 132730115167, 51801782763, 122347298993, 95323288245, 111299652270, 61394989717, 53017490066, 135150763850, 67838675321, 10445453807,
 32451851111, 121435434401, 41472737813, 99510791716, 16604033993, 8937318400, 11916572842, 128830583718, 58483521079, 42272581161, 67838675321, 49247033536, 12842195820,
 76084130681, 67838675321, 80583910753, 109419982435, 48276973322, 102042173056, 8143792048, 119155909678, 118525282599, 18180729027, 118302091549, 48939050408, 53343812050,
 134436556424, 99510791716, 11402462673, 128254672178, 32451851111, 24820967943, 2289666601, 67838675321, 53325619782, 93118634454, 138975004938, 21549294530, 53464836029,
 84535118118, 55567532012, 60953646600, 140022479220, 60406377104, 124458401638, 19331552844, 107538086014, 55567532012, 110163718396, 13383853049, 49223103588, 7196313163,
 76994357681, 105169274710, 11916572842, 76084130681, 10445453807, 60406377104, 53091459651, 68859378544, 93118634454, 22993235652, 9819942895, 6021499637, 105641108798,
 8163275090, 129006674195, 33018597367, 134436556424, 99510791716, 20726771902, 109617994988, 48276973322, 76433123792, 81824877670, 33312109218, 4129343275, 93118634454,
 76297192074, 19934675813, 278652957, 93118634454, 13882113497, 16444311286, 140022479220, 116325559091, 76297192074, 8723964498, 67838675321, 67838675321, 12791084566,
 48276973322, 36918217478, 122852256825, 125116015707, 110958416337, 91316414319, 79856170521, 7725833964, 1497737118, 119155909678, 76084130681, 101469119284, 28825599013,
 19373722934, 76297192074, 123839973295, 109662284582, 2289666601, 41431375973, 99945753690, 68859378544, 8163275090, 128002984714, 20726771902, 59333338225, 49733007890,
 4129343275, 101689334650, 41352100738, 55567532012, 141201477218, 140022479220, 60406377104, 11916572842, 92750529638, 59412461012, 12054234888, 59412461012, 92467315208,
 134436556424, 46188935943, 9819942895, 104624629887, 20726771902, 76994357681, 109662284582, 48629602520, 70711492365, 60406377104, 55567532012, 129006674195, 53464836029,
 67838675321, 123206093202, 30504162200, 8163275090, 84033987178, 13052995926, 101689334650, 123206093202, 89538597584, 82160505866, 84033987178, 140715775835, 70711492365,
 114797646698, 70711492365, 94682748250, 119155909678, 32451851111, 36313089447, 98872247810, 99510791716, 6161670129, 22982991901, 11257417735, 96613442842, 134436556424,
 99510791716, 20726771902, 109617994988, 48276973322, 36675139107, 37669707750, 128149422472, 33816441055, 104477744544, 106092336035, 41431375973, 84535118118, 63694391231,
 14618048648, 119155909678, 76084130681, 111351397800, 109617994988, 55327794934]
解密后的明文为：
 2002AMTURINGAWARDRSAanacronymforRivestShamirandAdlemanusesalgorithmicnumbertheorytoprovideanefficientrealizationofapublickeycryptosystemaconceptfirstenvisionedtheoreticallyby
 WhitfieldDiffieMartinHellmanandRalphMerkleRSAisnowthemostwidelyusedencryptionmethodwithapplicationsthroughouttheInternettosecureonlinetransactionsIthasalsoinspiredbreakthroug
 hworkinboththeoreticalcomputerscienceandmathematics

Process finished with exit code 0
```

二、 实验过程中遇到的问题有哪些？你是怎么解决的。

```
if Y=0 then return X=gcd(f,d);
```

```
 if Y3=0 then return  X3=gcd(f, d); no inverse;
if  Y3=1 then return  Y3=gcd(f, d); Y2=d^-1 mod f;
```

首先，指导书中有部分伪代码如上图所示，起初我并未理解其含义，误以为是使用递归进行计算，后来我仔细比对了 ppt 上的讲解，才发现这几句伪代码的含义分别是 y 等于 0 时，x 是 f 和 d 的最大公因数；y3 等于 0 时，x3 是 f 和 d 的最大公因数并且此时无乘法逆；y3 等于 1 时，y3 是 f 和 d 的最大公因数并且此时 y2 是所求的乘法逆。

```
Q=X3/Y3 ;
```

然后，在扩展欧几里得算法中，有如上图的计算过程。起初阅读伪代码时，我很疑惑，因为通过这样的计算，y3 下一轮始终会变成 0，返回没有乘法逆的结果。后来比对 ppt 后我才发现此处的除法是指整除操作。同时，此次实验我是使用 python 进行完成的，在做整除运算时，我起初像 c 语言一样直接使用了符号“/”，但结果始终不对。经过单步调试后我才意识到 python 中的整除是“//”，最后结果终于正确了。

三、 请说明你的字符分组方式，以及关键的算法例如扩展欧几里德，素数检测，快速幂等。

在本次实验中，我首先去掉文本中的空格和其它标点符号，然后将每个字母或数字与一个两位的十进制数字对应，数字对应到 00-09，a-z 对应到 10-35，A-Z 对应到 36-61。最后将两个字符，即 4 位十进制数字组成一个明文分组。若最后一个分组不足四位，即只有一个字母，那么我就用数字 62 进行填充。

关键算法：

扩展欧几里得：不仅可以求出两个正整数的最大公因子，而且当两个正整数互素时，还可求出其中一个数关于另一个数的乘法逆元。由于本实验中，已经保证了公钥 e 与 n 的欧拉函数 fai_n 互质，因此一定可以通过扩展欧几里得法求出 e 关于 fai_n 的乘法逆。

算法如下：

```
# 产生私钥d
# d为b对a的乘法逆，即e对fai_n的乘法逆
def extend_euclid(a, b):
    x = [1, 0, a]
    y = [0, 1, b]
    t = [0, 0, 0]
    while y[2] != 1:
        q = x[2] // y[2]
        for i in range(3):
            t[i] = x[i] - q * y[i]
        x[:] = y[:]
        y[:] = t[:]
    d = (y[1] + a) % a
    return d
```

素数检测：通过 Miller-Rabin 算法进行素性检测，它可以判断一个数“为合数”或者“很有可能为素数”。具体而言，它通过测试该数是否满足如下两个素数的性质，来判断该数是否“很有可能为素数”。

- 素数性质1：若p是素数，a是小于p的正整数，则$a^2 \bmod p = 1$，当且仅当$a \bmod p = 1$或$a \bmod p = -1 \bmod p = p - 1$。
  - 运用模算术运算规则$(a \bmod p)(a \bmod p) = a^2 \bmod p$可证。
- 素数性质2：设p是大于2的素数，有$p - 1 = 2^k q$，其中$k > 0$，q为奇数。设a是整数且$1 < a < p - 1$，下面两个条件之一成立：
  - a^q模p和1同余，即$\boldsymbol{a^q \bmod p = 1}$，等价地，$a^q \equiv 1 (\bmod p)$。
  - 整数$a^q, a^{2q}, a^{4q}, \cdots, a^{2^{k-1}q}$中存在一个数，模$p$时和$-1$同余。即存在一个$j (1 \le j \le k)$满足$\boldsymbol{a^{2^{j-1}q} \bmod p = p - 1}$。
    - 利用费马定理和素数性质1可证。

具体算法如下图所示：

```python
# 对prime进行素性检测
def is_prime(prime):
    k = 0
    q = prime - 1
    while not q % 2:
        q >>= 1
        k += 1
    assert prime - 1 == (1 << k) * q

    a = random.randrange(2, prime)  # 2到prime-1之间的随机整数
    remain = quick_pow(a, q, prime)
    if remain == 1 or remain == prime - 1:
        return True  # 很有可能为素数

    for i in range(1, k):
        if quick_pow(a, ((1 << i) * q), prime) == prime - 1:
            return True  # 很有可能为素数

    return False  # 为合数
```

快速幂：因为在 RSA 的加密和解密过程中均使用到了求幂后取模的运算，因此我们可以根据取模运算的性质，使用快速幂取模的方式来进行快速且不溢出的运算。

具体算法如下图所示：

```python
# mod n下的快速幂算法
# m为底数，e为指数，计算m**e%n
def quick_pow(m, e, n):
    ans = 1
    while e:
        if e & 1:
            ans = (ans * m) % n
        m = m * m % n
        e >>= 1
    return ans
```