

Github 账号：Nora-Qiu

### 实验摘要：

RSA 密码算法是使用最为广泛的公钥密码体制。选择 5 个参数：两个素数 $p$ 和 $q$ 、模数 $N = pq$ 、加密指数 $e$ 和解密指数 $d$ 。设 $m$ 为待加密消息，RSA 体制破译相当于已知 $m^e \bmod N$ ，能否还原 $m$ 的数论问题。

有人制作了一个 RSA 加解密软件。Alice 使用该软件发送了一个通关密语，且所有加密数据已经被截获。实验目的为仅从加密数据恢复通关密语及 RSA 体制参数。并给出原文和参数。如果不能则给出已恢复部分并说明剩余部分不能恢复的理由。

将实验步骤分为以下三部分：

- 一、从帧中得到公钥和密文
- 二、确定密文的解密方法并解密
- 三、对解密明文进行整理

### 实验题目

有人制作了一个 RSA 加解密软件（采用的 RSA 体制的参数特点描述见密码背景部分）。在附件 3-1 中，已知该软件发送某个明文的所有参数和加密过程的全部数据。Alice 使用该软件发送了一个通关密语，且所有加密数据已经被截获。实验目的为仅从加密数据恢复通关密语及 RSA 体制参数。并给出原文和参数。如果不能则给出已恢复部分并说明剩余部分不能恢复的理由。

### 实验内容

RSA 密码算法描述如下：

1) RSA 体制参数选取

Step1. 每个使用者，任意选择两个大素数 $p$ 和 $q$ ，并求出其乘积

$N = pq$ 。

Step2. 令 $\varphi(N) = (p - 1)(q - 1)$ ，选择整数 $e$ ，使得 $\text{GCD}(e, \varphi(N)) = 1$ ，

并求出 $e$ 模 $\varphi(N)$ 的逆元 $d$ ，即 $ed \equiv 1 \bmod \varphi(N)$ 。

Step3. 将数对 $(e, N)$ 公布为公钥， $d$ 保存为私钥。

2) 加解密过程

Bob 欲传递明文 $m$ 给 Alice，则 Bob 首先由公开途径找出 Alice 的公

钥 $(e, N)$ ，Bob 计算加密的信息 $c$ 为： $c \equiv m^e \bmod N$ 。

Bob 将密文 $c$ 传送给 Alice。随后 Alice 利用自己的私钥 $d$ 解密：

$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \bmod N$ 。

2. Alice 使用的 RSA 密码体制：

1) 模数 $N = pq$ 规模为 1024 比特，其中 $p, q$ 为素数；

2) 素数 $p$ 由某一随机数发生器生成；

3) 素数 $q$ 可以随机选择，也可以由 2) 中的随机数发生器产生；

4) 可以对文本加密，每次加密最多 8 个明文字符；

5) 明文超过 8 个字符时，对明文分片，每个分片不超过 8 个字符；

6) 分片明文填充为 512 比特消息后再进行加密，填充规则为高位添加 64 比特标志位，随后加上 32 比特通信序号，再添加若干个 0，最后 64 比特为明文分片字符对应的 ASCII 码（注：填充方式参见加密案例，但注意每次通信的标志位可能变化）；

7) 分片加密后发送一个加密帧数据，帧数据文件名称为 FrameXX，

其中 XX 表示接收序号，该序号不一定等于通信序号；

8) 帧数据的数据格式如下，其中数据都是 16 进制表示，结构如下


1024bit 模数  $N$  | 1024bit 加密指数  $e$  | 1024bit 密文  $m^e \bmod N$ 。

9) 由于 Alice 初次使用该软件，可能会重复发送某一明文分片。

将实验过程分为以下几个步骤:

一、从帧中得到公钥和密文, 分别存储在链表种

```
#求出每个帧中的N, e和c
m, N, e, c = [], [], [], []
filename = ['Frame' + str(i) for i in range(21)]
for i in range(21):
    fd = open(filename[i], 'r')
    m.append(fd.read())
    fd.close()
for frame in m:
    N.append(frame[0:256])#N
    e.append(frame[256:512])#e
    c.append(frame[512:768])#C
```



二、确定密文的解密方法并解密

①低加密指数: 当  $e=3$  时, 如果明文过小, 导致明文的三次方仍然小于  $n$ , 那么通过直接对密文三次开方, 即可得到明文。

原理: 加密为  $c=m^e \bmod n$

解密为  $m=c^d \bmod n$

分析: 如果  $e$  过小就可以直接对  $c$  开方。  $m^e=k*n+c$

$e=3$  的分组有: 7, 11, 15

但对  $e=3$  的密文进行低加密指数攻击, 很长时间都跑不出结果

②低加密指数广播攻击: 假设加密的明文相同

$c1=m^e \bmod n1$

$c2=m^e \bmod n2$

$c3=m^e \bmod n3$

在  $e=3$  时可以得到  $c_x=m^3 \bmod (n_1*n_2*n_3)$

对  $c_x$  开 3 次方可以得出明文。但是对分组 7, 11, 15 进行广播攻击只能得到乱码

对  $e=5$  的分组 3, 8, 12, 16, 20 进行广播攻击得到结果:

```
Looking for low encryption index 5
e of Frame 3 is 5
e of Frame 8 is 5
e of Frame 12 is 5
e of Frame 16 is 5
e of Frame 20 is 5
end of the function
list of e=5: [3, 8, 12, 16, 20]
m of Frame3 : 79859945008197619126095179109374591466848832845633905102428354228364957393952749844260287135965730215484818378063570600824813771406228340542500306493542
b't is a f'
```

③共模攻击: 如果在 RSA 的使用中使用了相同的模  $n$  对相同的明文  $m$  进行了加密, 那么就可以在不分解  $n$  的情况下还原出明文  $m$  的值。 ( $e1$  和  $e2$  互素)

$c1=m^{(e1)} \bmod n$

$c2=m^{(e2)} \bmod n$

需要: 密文  $c_1, c_2$  并且  $n$  相同.

用扩展欧几里得算法求出  $re_1 + se_2 = 1 \bmod n$  的两个整数  $r$  和  $s$ , 可得:

$$c_1^r c_2^s = m^{(re_1 + se_2)} \bmod n = m \bmod n$$

识别: 不只有一个  $c$ , 有相同的  $n$

主要代码为:

```
for i in range(len(listn)-1):
    for j in range(i+1, len(listn)):
        if(listn[i]==listn[j]):
            print('They are the same', i, ' ', j, ': ', listn[i])
            assert (libnum.gcd(liste[i], liste[j]))
            _, s1, s2 = gmpy2.gcdext(liste[i], liste[j]) # 扩展欧几里得算法
            # 若 s1<0, 则 c1^s1 == (c1^-1)^(-s1), 其中 c1^-1 为 c1 模 n 的逆元。
            if s1 < 0:
                s1 = -s1
                listc[i] = gmpy2.invert(listc[i], listn[i])
            if s2 < 0:
                s2 = -s2
                listn[j] = gmpy2.invert(listc[j], listn[i])
            print('m of Frame', i, ' and ', j, ' is: ', pow(listc[i], s1, listn[i]) * pow(listc[j], s2, listn[i]) % listn[i])
            return pow(listc[i], s1, listn[i]) * pow(listc[j], s2, listn[i]) % listn[i]
print('end of the function')
-----
They are the same 0, 4 : 90058705186558569935261948496132914380077312570281980020033760044382510933070450931241348678652103772768114420567119848142360867111065753301402088676701
m of Frame 0 and 4 is: 7985094500508197619216095178940144263456537191513943586246038968032293358555096757707390195545717847658989113846815132728257700850422867276861362837615205
-----
b'My secre'
```

④利用公约数: 如果有两个  $n$  有相同的素因子则可直接分解  $n$

通过分别计算两个  $n$  的最大公约数得到帧 1 和 18 有相同素因子, 对他们进行分解, 得到明文。

主要代码为:

```
for i in range(len(listn)):
    for j in range(len(listn)):
        if(i!=j):
            gcd = gmpy2.gcd(listn[i], listn[j])
            if(gcd!=1 and gcd!=listn[i]):
                print('Frame', i, ' and Frame', j, ' have a common factor')
                print('gcd: ', gcd)

                common.append(i)
                common.append(gcd)
                # common.append(j)

for i in range(0, len(common), 2):
    q = listn[common[i]] // common[i+1]
    phi_n = (common[i+1] - 1) * (q - 1)
    d = gmpy2.invert(e[common[i]], phi_n)
    m_ = gmpy2.powmod(c[common[i]], d, listn[common[i]])
    m.append(int(m_))
print('end of the function')
return m

utilize common factors-----
Frame 1 and Frame 18 have a common factor
gcd: 7273268163465293471933643674908027120929096536045429682300347130226398442391418956862476173798834057392247872274441320512158525416407044516675402521694747
Frame 18 and Frame 1 have a common factor
gcd: 7273268163465293471933643674908027120929096536045429682300347130226398442391418956862476173798834057392247872274441320512158525416407044516675402521694747
end of the function
m of Frame0 : 798509450050819761921609518080167787156987724616253709366632221618908094578251267121296120303844455578044186369054794932649594298123971808053505859471726
798509450050819761921609518080167787156987724616253709366632221618908094578251267121296120303844455578044186369054794932649594298123971808053505859471726
b''. Imagin'
m of Frame18 : 7985094500508197619216095180632447543559573604830846774809932829993009346943656679076091111448196673223946159159299511453928830060256372341748824036810818
7985094500508197619216095180632447543559573604830846774809932829993009346943656679076091111448196673223946159159299511453928830060256372341748824036810818
b'm A to B'
```

⑤Pollard p-1 分解法: 找到给数  $n$  的一个因子  $d$ 。大致步骤如下:给定: 整数  $n$  (已知是合数).目标: 找到一个因子  $d|n$ .

步骤:

0) 固定整数  $B$ 1) 选择一个整数  $k$ ,  $k$  是大部分 (或者全部)  $b$  的乘积满足  $b \leq B$ ; 例如  $k = B!$ 2) 选择一个随机整数  $a$  满足  $2 \leq a \leq n-2$ .3) 计算  $r = a^k \bmod n$ .4) 计算  $d = \gcd(r-1, n)$ .5) 如果  $d = 1$  或者  $d = n$ , 回到步骤1. 否则,  $d$  就是要找的因子。

主要代码如下:

```
def pp1(n):
    B=2**20
    a=2
    for i in range(2,B+1):
        a=pow(a,i,n)
        d=gmpy2.gcd(a-1,n)
        if 1< d_ <n: #如果d=1或者d=n则要重新寻找d
            q=n//d_
            n=q*d_
    return d_

utilize pollard p-1-----
p of 2 is : 1719620105458406433483340568317543019584575635895742560438771105058321655238562613083979651479555788009994557822024565226932906295208262756822275663694111
q of 2 is : 5248406512257276757293534477361686456679280880304125291106733197354892893647364164212186415880889674435558369420400890814461263958618375991691022752189839
m of Frame 1 is: 7985094500508197619216095179955526231518359039504085499384375285208722951588232710528610745087205142997963341034305759963660378376322965694370023122954611
utilize pollard p-1-----
p of 6 is : 920724637201
q of 6 is : 159482692259010816139523195494724350795654007589889398757383554027183924116413427533184220914037106543253510345232484145256542086894498546422964942024070855408815
m of Frame 1 is: 798509450050819761921609518012475655952866268083577581824076467140479455042708870266548083667745302555445904556554197836227491297306315958671527603954464
utilize pollard p-1-----
p of 19 is : 108566349659
q of 19 is : 8672576161185989538639614103149718994898444713854221542046255310108199100830450746116307835487797028264925105145753290295500985600940585391739663001701132050035708
m of Frame 1 is: 7985094500508197619216095179786295903508055398172395180527985899012651352749376718391740653496957260441467636503057322091093265455339620683498933545692718
```

```
b' That is'
b' "Logic '
b'instein.'
```

将明文转换为字符串得到

## ⑥费马分解法

当  $|p-q|$  较小时,  $\frac{(p-q)^2}{4}$  也比较小, 进而  $\frac{p+q}{2}$  与  $\sqrt{n}$  相近, 可以按照以下方法分解:

- 顺序检查  $\sqrt{n}$  的每一个整数  $x$ , 直到找到一个  $x$  使得  $x^2 - n$  是平方数
- 根据平方差公式分解  $N$

主要代码如下:

```
x = gmpy2.iroot(N, 2)[0]
for j in range(1000000):
    x += 1
    if gmpy2.iroot(x ** 2 - N, 2)[1] == 1:
        y = gmpy2.iroot(x ** 2 - N, 2)[0]
        p = x + y
        q = x - y
        break
phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
m=pow(c, d, N)

D:\python\python.exe D:/密码学相关/实验二/RSAdescription.py
*****WELCOME*****
utilize Fermat [p-q]-----
a of Frame 10 is: 798509450050819761921609518029398688753896632216746613709715405760086614926594694802350928267700908110954750096802635708794604218289672526108082271905140
utilize Fermat [p-q]-----
a of Frame 14 is: 53520979004367358747577459270000434733004015395705219250808161273191300592272023531900128560160712066120630113270487961863116469354302804093244359196330361136105581526
```

### 三、根据解密明文进行整理

通过以上常规的 RSA 破解方法，得到了一些明文片段：

Frame0:My secre  
Frame1:. Imagin  
Frame2: That is  
Frame3:t is a f  
Frame4:My secre  
Frame6: " Logic  
Frame8:t is a f  
Frame10:will get  
Frame12:t is a f  
Frame16:t is a f  
Frame18:m A to B  
Frame19:instein.  
Frame20:t is a f

### 实验总结

这道题用不同的攻击方法对密文分组进行破解，虽然最后我得到了一些明文片段，但是无法将完整的明文信息回复出来。我从简单的低加密指数攻击开始做起，但由于无法运行得到结果一度不敢再做下去，但是经过上网查找资料没有发现有效的破解方法。最开始对一些破解方法都不熟悉，在编写代码的时候总是出错，导致最后的代码冗余且运行时间过长。希望有时间能再把代码优化一下！经过本次实验对 RSA 体制有了更深的了解。

### 参考文献

- [https://blog.csdn.net/qg\\_38204481/article/details/83189041](https://blog.csdn.net/qg_38204481/article/details/83189041)
- <https://www.jianshu.com/p/dda528239554>