RSA算法总结（数学知识/CTF题型）

原创

暮w光 于 2021-12-18 00:42:30 发布 595 收藏 3

分类专栏： # 密码学 文章标签： 算法 安全 web安全

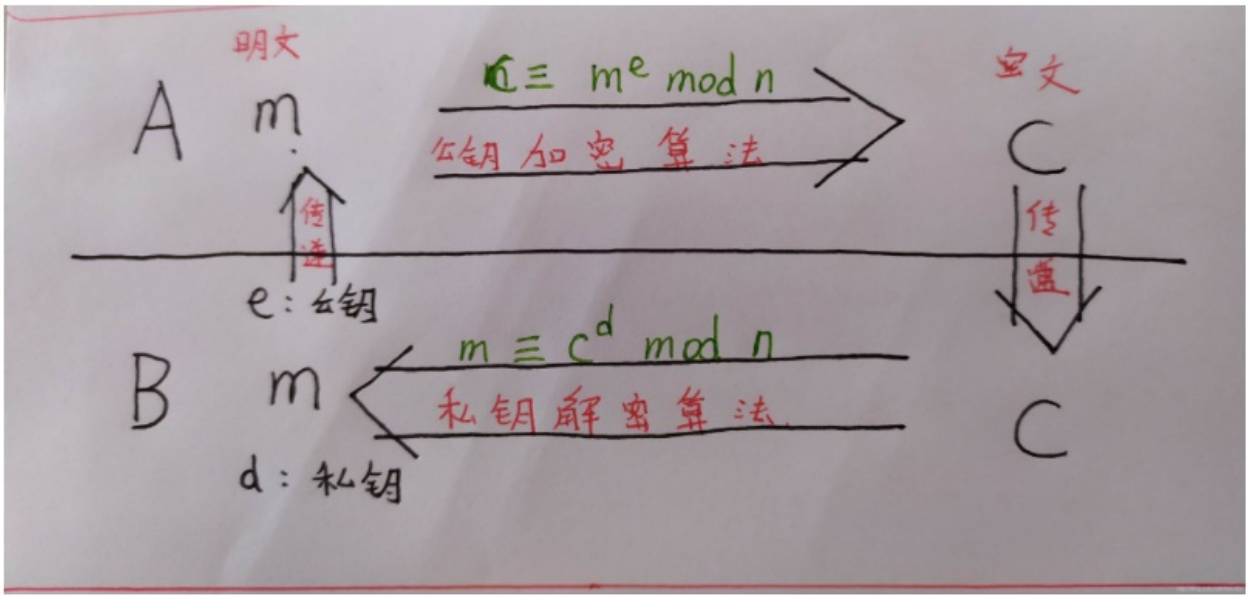
本文链接：https://blog.csdn.net/qi_SJQ_/article/details/121299078

一.RSA简介

简要概括就两句话：公钥（e）加密，私钥（d）解密。

一，原理：

信息传递的过程：



- A 要 向 B 传递信息 m
- 首先 B 要把 公钥 （n, e）传递给 A
- 然后 A 拿着公钥 进行 公钥加密算法 将 明文 m 变成 密文 c
- 接着 A 把 生成的 密文 c 传递 给 B
- 最后 B 再利用 私钥 （n，d）进行 私钥解密算法 还原出来 明文 m

二.RSA过程

rsa加密的过程：

- **随便找出两个 整数 q 和 p （q，p互素，即：公因数只有1）**
- **求出n = q * p**
- **φ(n)= （p-1）*（q-1） 欧拉公式**
- **公钥 e： 随机取，要求： e 和 φ(n) 互素（公因数只有 1）； 1< e < φ(n)）；**
- **私钥 d： ed ≡ 1 （mod φ(n)） （ed 除以 φ(n) 的 余数 为 1 ）**

加密算法：

$$c \equiv m^e (\mathrm{mod}\ n)$$

解密算法：

$$m \equiv c^d (\mathrm{mod}\ n)$$

三.RSA各参数

- p 和 q：两个大的质数，是另一个参数N的的两个因子。
- N/n：大整数，可以称之为模数。
- e 和 d：公钥和私钥，互为无反数的两个指数。
- (N, e)：公钥对，有N
- (N, d)：私钥对，也有N
- c 和 m：密文和明文。
- pow(x, y, z)：效果等效pow(x, y)1 % z， 先计算x的y次方，如果存在另一个参数z，需要再对结果进行取模。
- 密钥长度：n以二进制表示的的位数，例如密钥长度为512代表n用二进制表示的长度为512bit。

> 对于RSA加密算法，公钥(N, e)为公钥，可以任意公开，破解RSA最直接（亦或是暴力）的方法就是**分解整数N**，然后**计算欧拉函数φ(n)= (p-1) * (q-1)**,再通过**d * e ≡ 1 mod φ(N)**，即可计算出 d，然后就可以使用私钥(N, d)通过**m = pow(c,d,N)解密**明文。

四.数学知识

1.质数与互质数：

> 一个大于1的自然数，除了1和它本身外，不能被其他自然数整除（除0以外）的数称之为质数（素数）；否则称为合数。
> 例如，15＝3×5，所以15不是素数，13除了等于13×1以外，不能表示为其它任何两个整数的乘积，所以13是一个素数。
> 1不是质数，也不是合数
> **公约数只有1的两个数，叫做互质数。**

2.同余定理：

“≡”是数论中表示同余的符号。
同余的定义如下：
给定一个正整数m，如果两个整数a和b满足a - b能被m整除，即(a - b) mod m = 0，那么就称**整数a与b对模m同余**，记作a ≡ b (mod m)，同时可成立a mod m = b。
注意，同余与模运算是不同的。
显然,有如下事实：
（1）若a≡0(mod m)，则m|a；
（2）a≡b(mod m)等价于a与b分别用m去除，余数相同。

3.**欧拉函数：**

任意给定正整数n，计算在小于等于n的正整数之中，有多少个**与n构成互质**关系？计算这个值的方法就叫做欧拉函数，以φ(n)表示.
例如：在1到8之中，与8形成互质关系的是1、3、5、7，所以φ(n)=φ(8)=4。
φ(n)=(p-1)(q-1)由来：

在RSA算法中，欧拉函数对以下定理成立
1.如果n可以分解成两个互质的整数之积，即n=p×q，则有：φ(n)=φ(pq)=φ(p)φ(q);
2.当p为质数，φ(p)=p-1

所以有φ(n)=(p-1)(q-1)

例如：和15互质的为1,2,4,7,8,11,13,14,将15分解成3,5且两者互质，故φ(n)=φ(15)=φ(3) * φ(5)=(3-1)* (5-1)=8

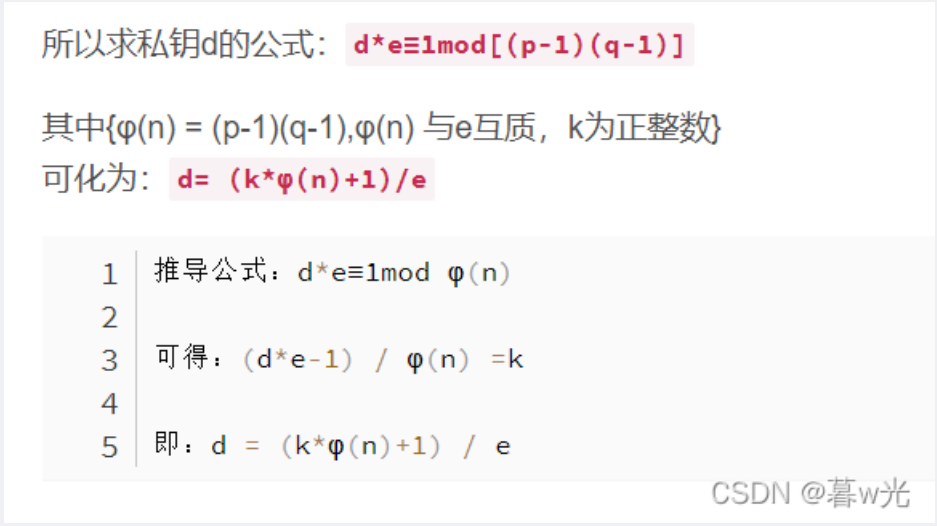
4.**欧拉定理：**

如果两个**正整数a和n两者互质**，则n的欧拉函数φ(n)也可以下面的等式也成立：a^φ(n)≡1(mod n)。
也就是说，a的φ(n)次方被n除的余数为1。
这就是φ(n)与公钥e互质的原因。

5.**模反元素：**
根据欧拉定理，有：
a^φ(n) = a × a^(φ(n)−1)≡1(mod n)

令b=a^(φ(n)−1)，得：a×b≡1(mod n)

b就是a的**模反元素** 所以，如果两个正整数a和n互质，那么一定**可以找到整数b使得ab-1被n整除**，或者说**ab被n除的余数是1**。
私钥d与公钥e的乘积d*e 减去1可以被φ(n)整除，或者说ed乘积除以φ(n)的余数为1。或者说de乘积与1对模φ(n)同余，**这就是私钥的d加密**



原理。

PS：利用两者乘积求n或由n反推p与q，通常n往往是一个 1024bit 的超大数，很难分解为两个质数。

脚本（n+e+c+p+q=m脚本）：

```
import libnum
from Crypto.Util.number import long_to_bytes

c = 0x6cd55a2bbb49dfd2831e34b76cb5bdfad34418a4be96180b618581e9b6319f86  #c：密文
n = 108539847268573990275234024354672437246525085076605516960320005722741589898641
#n = int("",16)
e = 65537
#e = int("",16)
q = 333360321402603178263879595968004169219
p = 325593180411801742356727264127253758939

d = libnum.invmod(e, (p - 1) * (q - 1))
m = pow(c, d, n)   # m 的十进制形式
string = long_to_bytes(m)  # m：明文
print(string)  # 结果为 b‘ m ’ 的形式
```

五.题型/常识等

1.最简单情况是知道p、q和公钥e，求出私钥d。

脚本如下：

```
import gmpy2
p = gmpy2.mpz(18443)    #初始化大整数
q = gmpy2.mpz(49891)
e = gmpy2.mpz(19)
phi_n = (p-1)*(q-1)
d = gmpy2.invert(e,phi_n)   #invert（x，m）返回y使得x * y == 1 modulo m，如果不存在y，则返回0
print("p=%s,q=%s,e=%s"%(p,q,e))
print("d is:\n%s"%d)
```

2.n较小时（256bit）可以rsatool或kali上yafu解密。

在线分解大素数：http://www.factordb.com/index.php

3.n较大时（大于768bit）只能爆破或利用在线网站， 这一类在线网站的原理是储存了部分n分解成功的的值。

4.由p,q,dp,dq,c求明文m的算法,其中涉及的**数学知识**：参考另一篇博客。

5.已知密文文件（flag.enc/cipher.bin/flag.b64）和公钥文件（pubkey.pem/key.pem/key.pub）求解明文 m：
1）利用rsatools工具：
语法：

```
python RsaCtfTool.py --publickey key.pem --uncipherfile cipher.bin
```

2）

首先，提取 pubkey.pem 中的 参数： （openssl linux自带，win下也可以安装，具体openssl的用法自行百度）

一般能提取出来 n的值

```
openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem
```

其次，用 yafu 分解 n 得到 q，p （win环境下）

```
yafu.exe factor(461616564564564654151561313213214659789765613131)
```

然后 制作 私钥： 生成私钥文件 private.pem

```
python rsatool.py -o private.pem -e 65537 -p 27512786035134892817328517438158115229 -q 319576316814478949870590
```

最后 用private.pem文件 解密 flag.enc文件

```
openssl rsautl -decrypt -in flag.enc -inkey private.pem
```


6.已知c，e，n（非常大），和 dp，dq，求解明文m（领航杯2019的一道题：EasyRSA或2019山东省赛“深思杯”中 被加密的消息 一题，两者类似。）：

```
import gmpy2
import libnum
e=65537
n=169697521655091326276302669687488543303407016921254276195598364883502982347355714803530786149755803784673559523337553139355165137735521633929526563214902684525566048589668999562421070084105586579243442956519392973280079322457416609105100329695275982662705110048576745348022033873996782318808942523284311332246535449486612837776459850282076095266548166451555589151977450625691245874123787160498140406706650794800556448734707566029933872619395669588062965997829434601415820451509710312112186170912832841185737140292663312273273987242651703526467940687027896459808100055493763995351108200524724198468018091101865571621270
dp=178162577529102887026968525752110809032954301272870546778254691395153764262362176924644112218994867137499094640516445986741064682559131062261837911628429379409097029216526333474939300999933541308990379662432616803961828707819264649048853406280396041879087489043552939304738922871883524437064521518735808180 5
c=0x6c78dcee37830f3ec4ab4989d40fbb595060b3fbc395d52ad26defc13372c1a3948c5388f4e450e46e016c7803133d6881e5efc3b90a4789448097c94124590b1e7949f2524d7edccd61a27691c18d090ac1f54643b563141306045417581e3b263f4ad2816136a48b106f3058b08e2a810f4ae8ef25916cc110b41ac8158ce69ecbe20fc60c1ddb20154c6646bc5142aefe47abf053a8ac949d5bc057bb18b191ad08070fe9ec5d76b1fceae685514532448c1b388b2d38e7241ac19c296e95e4e021a3a4015d909a1d53a2eb7fa86f6329f4e6c937f958be576c58fab4d9c9126999c99bb28718efc41a6f5db52b47942a2ddf21639f020b5489699cf22b46

for i in range(1,65538):
    if (dp*e-1)%i == 0:
        if n%(((dp*e-1)//i)+1)==0:
            p=((dp*e-1)//i)+1
            q=n//(((dp*e-1)//i)+1)
            phi = (p-1)*(q-1)
            d = gmpy2.invert(e,phi)%phi
            print(libnum.n2s(pow(c,d,n)))
```

7.公共模数攻击：【BUUCTF :RSA3】
适用于：使用相同的模数 N 、不同的私钥，加密同一明文消息。

基本原理

假如采用两个或者两个以上的公钥(N,e)来加密同一条信息，可以得到下面的结论：

```
c1 = pow(m, e1, N)
c2 = pow(m, e2, N)
```

分别拿对应的私钥来加密，可以得到相同的明文`m`

```
m = pow(c1, d1, N)
m = pow(c2, d2, N)
```

CSDN @暮w光

假设攻击者已知n,e1,e2,c1,c2，即可可以得到明文m，因为e1和e2互质，所以使用`欧几里得算法`（用于计算两个整数a,b的最大公约数）可以找到能够满足以下条件的`x`，`y`：

```
pow(x,e1)+pow(y,e2)=1
```

假设x为负数，需再使用欧几里得算法来计算

```
pow(c1,-1)
```

则可以得到

```
pow(pow(c1,-1),-x) * pow(c2,y) = p mod(n)
```

如果p<n，则p可以被计算出来。
地址：https://www.ichunqiu.com/battalion?q=2451

CSDN @暮w光

exp：

```python
from gmpy2 import invert
# 欧几里得算法
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)

def main():
  n = 22708078815885011462462049064339185898712439277226831073457888403129378547350292420267016551819052430779004755846649044001024141485283286483130702616057274698473611149508798869706347501931583117632710700787228016480127677393649929530416598686027354216422565934459015161927613607902831542857977859612596282353679327773303727004407262197231586324599181983572622404590354084541788062262164510140605868122410388090174420147752408554129789760902300898046273909007852818474030770699647647363015102118956737673941354217692696044969695308506436573142565573487583507037356944848039864382339216266670673567488871508925311154801
  c1 = 22322035275663237041646893770451933509324701913484303338076210603542612758956262869640822486470121149424485571361007421293675516338822195280313794991136048140918842471219840263536338886250492682739436410013436651161720725855484866690084788721349555662019879081501113222996123305533009325964377798892703161521852805956811219563883312896330156298621674684353919547558127920925706842808914762199011054955816534977675267395009575347820387073483928425066536361482774892370969520740304287456555508933372782327506569010772537497541764311429052216291198932092617792645253901478910801592878203564861118912045464959832566051361
  c2 = 18702010045187015556548691642394982835669262147230212731309938675226458555210425972429418449273410535387985931036711854265623905066805665751803269106880746769003478900791099590239513925449748814075904017471585572848473556490565450062664706449128415834787961947266259789785962922238701134079720414228414066193071495304612341052987455615930023536823801499269773357186087452747500840640419365011554421183037505653461286732740983702740822671148045619497667184586123657285604061875653909567822328914065337797733444640351518775487649819978262363617265797982843179630888729407238496650987720428708217115257989007867331698397
  e1 = 11187289
  e2 = 9647291
  s = egcd(e1, e2)
  s1 = s[1]
  s2 = s[2]
  # 求模反元素
  if s1<0:
    s1 = - s1
    c1 = invert(c1, n)
  elif s2<0:
    s2 = - s2
    c2 = invert(c2, n)

  m = pow(c1,s1,n)*pow(c2,s2,n) % n
  print(m)

if __name__ == '__main__':
  main()
```

8.已知n（非常大）,e,d求p,q（无法直接 从n分解）：

```
import random
from md5 import md5

def gcd(a, b):
   if a < b:
     a, b = b, a
   while b != 0:
     temp = a % b
     a = b
     b = temp
   return a

def getpq(n,e,d):
 p = 1
 q = 1
 while p==1 and q==1:
  k = d * e - 1
  g = random.randint ( 0 , n )
  while p==1 and q==1 and k % 2 == 0:
   k /= 2
   y = pow(g,k,n)
   if y!=1 and gcd(y-1,n)>1:
    p = gcd(y-1,n)
    q = n/p
 return p,q

def main():
 n = 16352578963372306131642407541567045533766691177138375676491913897592458965544068296813122740126583082006556
2176162960095164132028336982688456344974789881288503732218535169732590868457258134248505486825038271911215486932
8876324361903322432269807598766753186321346822365418165801275489758814702743722926909824696981122612988332759802
1859724836993626315476699384610680857047403431430525708390695622848315322636785398223207468754197643541958599210
1272613457709145146701990474350857144036414690162129583619939693045452140615601602677607864821633737844376418082
9265448934348761344616554298838268772959338488751627269065430 9
 e = 65537
 d = 9459928379973667430138068528059438139092368625339079253289560577985304435062213121398231875832264894458314 6
2957545555348575268564374326665463082995744200877525977631158565401485816534175754728411206188515800688147574055
3532826576260839430343960738520822367975528644329172668877696208741007648370045520535298040161675407779239300466
6816154938926922655422902554086735338530116621349538694326325540082353408648033776103524381462645247707103452734
3972410708019018291828554742616656180371664408941407838947507210331543263819757818610657662672886902036621407745
5194554930725576023274922741115941214789600089166754476449453
 p,q = getpq(n,e,d)
        print p
        print q
        print "Flag: flag{%s}" %md5(str(p + q)).hexdigest()
if __name__ == '__main__':
 main()
```

9.利用公约数：

识别

识别此类题目，通常会发现题目给了若干个n，均不相同，并且都是2048bit，4096bit级别，无法正面硬杠，并且明文都没什么联系，e也一般取65537。

例题

```
n1=90510139654040844828700878648214555351590086960429530219656310890957953488309543831273238532725289677293110451796054076935926656833116605812048865711463277202884558749272811281211173235796912047923999131066275432744570361724558148057156682937056036753868782209477221869141129904527221743637136302976851596693289515208919384034527976506858495236581919474114290688297340537451804607586042830513443396414298193731123652117392161604204941670719964385068505261683893868504997961020036254042456457962716903107488043274n2=13225948396179603816062046418717214792668512413625091569997524364243995991961018894150059207824093837420451375240550310050209398964506318518991620142575926623780411532257230701985821629425722030608722035570690474171259238153947095310303522831971664666067542649034461621725656234869005501293423975184701929729170077280251436216167293058560030089006140224375425679571181787206982712477261432579537981278055755345457376707695179331206248027500456465759026371981603356413949710994207370175501187315320536623858566574 3
```

直接分解成功。而`欧几里得算法`的时间复杂度为：O(log n)。这个时间复杂度即便是4096 bit也是秒破级别

根据`欧几里德算法`算出的`p`之后，再用`n`除以`p`即可求出`q`，由此可以得到的参数有`p`、`q`、`n`、`e`，再使用常规方法计算出`d`，即可破解密文。

思路

如果两次加密的`n1`和`n2`具有相同的素因子，可以利用`欧几里德算法`直接分解`n1`和`n2`.

通过`欧几里德算法`计算出两个`n`的最大公约数`p`：

```python
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
def gcd_digui(a, b):
    if b != 0:
        return a
    return gcd(b,a%b)

p = gcd(n1,n2)
```


10.Roll**按行加密**（加密是按行进行的）：

题目是这样的：

```
{920139713,19}
704796792
752211152
274704164
18414022
368270835
483295235
263072905
459788476
483295235
459788476
663551792
475206804
459788476
428313374
475206804
459788476
425392137
704796792
458265677
341524652
483295235
534149509
425392137
428313374
425392137
341524652
458265677
263072905
483295235
828509797
341524652
425392137
475206804
428313374
483295235
475206804
459788476
```

完全考猜测猜到n=920139713，e=19，以下每一列组成密文c。

思路是：分解n，得p,q 根据p,q 获得欧拉数，根据欧拉数 获得 d，根据d 获得m

参考write-up。

六、攻击方式

1.其实破解rsa最直接方法就是分解模数N得q、p，基本只适用于N较小的时候，不过为保证安全性大部分时候都很难解出来，参考：

RSA安全性分析

对于RSA加密算法，公钥`(N, e)`为公钥，可以任意公开，破解RSA最直接（亦或是暴力）的方法就是分解整数`N`，然后计算`欧拉函数φ(n)=(p-1) * (q-1)`,再通过`d * e ≡ 1 mod φ(N)`，即可计算出`d`，然后就可以使用私钥`(N, d)`通过`m = pow(c,d,N)`解密明文。

保障RSA的安全性

> 1.定期更换密钥
> 2.不同的用户不可以使用相同的模数N
> 3.p与q的差值要大，最好是差几个比特
> 4.p-1与q-1都应该有大的素因子，一般建议选择的两个大素数p、q使得p=2p+1和q=2q+1也是素数
> 5.e的选择不要太小
> 6.d的选择也是不可以太小，最好满足`d>=n的4分之1`

CSDN @暮w光

2.

RSA低加密指数e攻击（e很小或很大）：最好参考这篇博客。

在 RSA 中 e 也称为加密指数。由于 e 是可以随意选取的，选取小一点的 e 可以缩短加密时间（比如 e=2,e=3），但是选取不当的话，就会造成安全问题。

RSA小指数e攻击

如果RSA系统的公钥e选取较小的值，可以使得加密和验证签名的速度有所提高，但是如果e的选取太小，就容易受到攻击。

有三个分别使用不同的模数n1,n2,n3，但是都选取e=3，加密同一个明文可以得到：

```
c1 = pow(m,3,n1)
c2 = pow(m,3,n2)
c3 = pow(m,3,n3)
```

一般情况下，n1,n2,n3互素，否则会比较容易求出公因子，从而安全性大幅度的减低。

CSDN @暮w光

1）e=2：把密文c开平方求解
因为c≡m^e(mod n)，所以相当于将明文m平方后得到的密文c，开根号即可。

题目: 01-西湖论剑rsa

```
已知：

e=2

c=92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321676580866698866l

求明文m?
```

```python
import gmpy2
import libnum
c = 92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321676580866698866
m = gmpy2.isqrt(c)
m = int(m)
m_text = libnum.n2s(m)  #将 十六进制转为 字符
print(m_text)


# flag1{Th1s_i5_wHat_You_ne3d_FirsT}
```

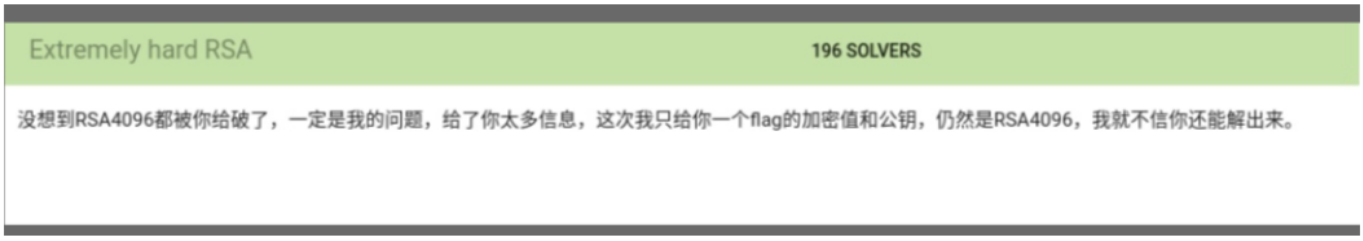

2）小明文攻击，例题：

适用情况：e较小，一般为3。

公钥e很小，明文m也不大的话，于是m^e=k*n+c 中的的k值很小甚至为0，爆破k或直接开三次方即可。

题目: 02-Jarvis OJ -Crypto-Extremely RSA，

给了 flag.enc 和 pubkey.pem 文件



因为e=3，很可能存在小名文攻击，

可以假设，k为0，将 c直接开三次方就可以得到明文 m

3）**低指数攻击**大多数出现在**e很大的时候：**

可以直接github用工具：https://github.com/pablocelayes/rsa-wiener-attack

3.

yafu工具使用：
不能直接分解n，不能使用公约数分解n，可以尝试yafu。

> 在p，q的取值差异过大，或者p，q的取值过于相近的时候，Format方法与Pollard rho方法都可以很快将n分解成功。
> 此类分解方法有一个开源项目yafu将其自动化实现了，不论n的大小，只要p和q存在相差过大或者过近时，都可以通过yafu很快地分解成功。

2.yafu使用方法

把yafu的环境变量配好，在cmd下输入

```
yafu-x64
```

```
factor(n)
```

3.使用yafu的时候遇到mismatched parens

这是因为在命令行里不支持过长的位数，所以我们只要把n的值从文件中去读取即可。

新建一个文件n.txt,文件最后一行换行，避免报错

```
yafu-x64 "factor(@)" -batchfile n.txt
```

4.在线分解网站

factor

sage

Mesieve

4.模不互素（存在两个或更多模数 n 且N1和N2不互质）：

适用情况：存在两个或更多模数，且gcd(N1,N2)!=1 也就是N1和N2不**互质**。
适用于，n超级大，用 **yafu 的factor分解不了** 的情况。

5.低加密指数广播攻击（模数n、密文c不同，明文m、加密指数e相同）

> 如果选取的加密指数较低，并且使用了**相同的加密指数**给一个接受者的群发送**相同**的信息，那么可以进行广播攻击得到明文。

适用范围:模数n、密文c不同，明文m、加密指数e相同。

一般的话e=k，k是题目给出的n和c的组数。

选取相同的加密指数`e`（例如e=3），对相同的明文m进行了加密并进行了消息的发送。有以下的等式成立。

```
c1 = pow(m, e, n1)
c2 = pow(m, e, n2)
c3 = pow(m, e, n3)
...
```

对上述等式运用`中国剩余定理`，在`e=3`时，可以得到

```
c_x = pow(m, 3, n1 * n2 * n3)
```

通过对 c_x进行三次开方可以得到明文。

识别

一般来说都是给了三组加密的参数和明密文，其中题目很明确地能告诉你这三组的明文都是一样的，并且e都取了一个较小的数字。

例题：Jarvis OJ -2018强网杯nextrsa-Level9

exp：

```python
#!/usr/bin/python
#coding:utf-8

import gmpy2
import time
def CRT(items):
    N = reduce(lambda x, y: x * y, (i[1] for i in items))
    result = 0
    for a, n in items:
        m = N / n
        d, r, s = gmpy2.gcdext(n, m)
        if d != 1: raise Exception("Input not pairwise co-prime")
        result += a * s * m
    return result % N, N
# 读入 e, n, c
e = 3
n = [856452939859749605250987551348123451190557129360825359177435234523787673329310883120372300895836722448948996961465670345596254926131544232744308965207457170865150544737930916610033106544017278196887549738641066771502618005791336320845011109556621923830338788802516140704347729137893141226904984974445754793226413737741112719294033280545231754721924805580219705045672662451686002410864257170381271937038729216667030019724157546141764859230986966981337401076576654460769101195796865258150488633125293614690145691058910248480703929456670391703309302814045284974791053786595809872069356982125618959389412111135773191919 89L,122221662972773428052606680420667337492588436220574975745514926808205739706180633567108108912216703663961488620705300684317726302713000645171413313809594138114828900801035117563639202993876201815251722473840854499446506786163988909470627038793077215062286728392704934535016486447870193601319910561583752964848707237174961843320785212219152349596276839522518652278492497524152421242357764289440528735010451274420314235381832828458009713595907351848506489862637218238048594108669271174132894610147544568025669329657105290635154052964760076428498007729341709619939250171460171977628051485334350406759623324696439151924 23L,505722403449977679353265451617891495453754741066440940368043210856907985674176436202318560459582926391892712146557869120190422791289702524477155386010271442934916328351069539119377485332395165312310950921536193785072493418382650807040423979171022921406338208139156495493554439251476916683081547545921879263938379671182475229115889529210335427463247055917926655068109545223966616521398699349610974705831468530348522230214483549005640293913322500323311213127563541932198289996544091252522575936868471715707716177178903529280898069381899400305195745292409238361901176051346615633641550303346790420492393767770845418243L]
c = [200109715577899319481307989830302019500384502691441045328210306679244007888699202385797295146726302218040960631491067424128699668147012254666063921710304113391195592807900403220811043633934535034174657683861740020158707945671486947222158730942988591324392534125314451879908454762752513488001667314811761555307555811537100859669767655055918095964178497835970556504405980351592880914954532056980446878699327560534470129944095981555522638075717139827581320663196127773064667082221355109181740556477357275040295075034302886094107451590376849483430552755732690671654607115848454801887065314503671471056294937361007260929 45L,192000529198181965585675287012240821551058528461097820216818481072264952936870214168711174449879238378102388728368181904576675094097140216691608158094136538804917926403464742488595598677437157155523727389092550501966380064722793643166789412578948989530883668617865004720957528905935214283258381481848917782832298703166940597341090453974483473204876054129882290470151749988935897315031143372731214636019847923393379701935968136601786362227643321559999935069145186005653941967924571449621800407866073356870202784428991469541268535802442642735265092380604946249808079933229751353666531819771478665671464923561370194144255L,139472154012792262758499374959660321249191375586503999463104145888271695325176008063849757465288838641176795125846754200258241826031590919024113159147462776573417414698101534673255911504491870664161628847444729412933247508123226824120148845586570093361529101634655204899712741578307286038726506352787416001618618307838494031229252162807775046441301376876537150849330433171916619633088324289555690337870719964068649997036795755254304111019900942536961264449228876589176900457905080244699242681321593225034738685978381387554331419676416079269629174285035653249394565248264369623848738941240461653762001300914160185208 0L]


data = zip(c, n)
x, n = CRT(data)
m = gmpy2.iroot(gmpy2.mpz(x), e)[0].digits()
print m
```

七.题型总结（待积累补充）：

各题型解析，exp参考的：https://blog.csdn.net/huanghelouzi/article/details/82943615

- 已知p、q、e求解d
- 已知p、q、e、c求解明文
- 已知c、n、e求解明文
- 已知c、e，求解明文
- 求解d
-

参考：
CTF —crypto —RSA原理及各种题型总结
CTF密码学中RSA学习以及总结
RSA算法原理（数学）
RSA加密算法详细解说
CTF中RSA套路
玩转RSA加密算法（一）