CTF ——crypto ——RSA原理及各种题型总结

原创

Captain Hammer 于 2019-09-23 20:56:16 发布 35820 收藏 301
分类专栏： CTF 类型题总结

本文链接：https://blog.csdn.net/vhkjhwbs/article/details/101160822

RSA原理及各种题型总结

Table of Contents

6，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）

写在最前面

（文章中的所有python脚本，都是我在python3.7 / python2.7跑过，输出正常的脚本,如果你跑不出来,极可能是你环境的问题）

（文章中的每一个例题，虽是收集来的，但都动手操作过，实际求解出flag 后 再挂出来的）

要求的库包括但不限于：gmpy2、pycryptodome、libnum

gmpy2：https://pan.baidu.com/s/1De5h6XmkAuop69aYDiQ2gg 提取码: 6y2t

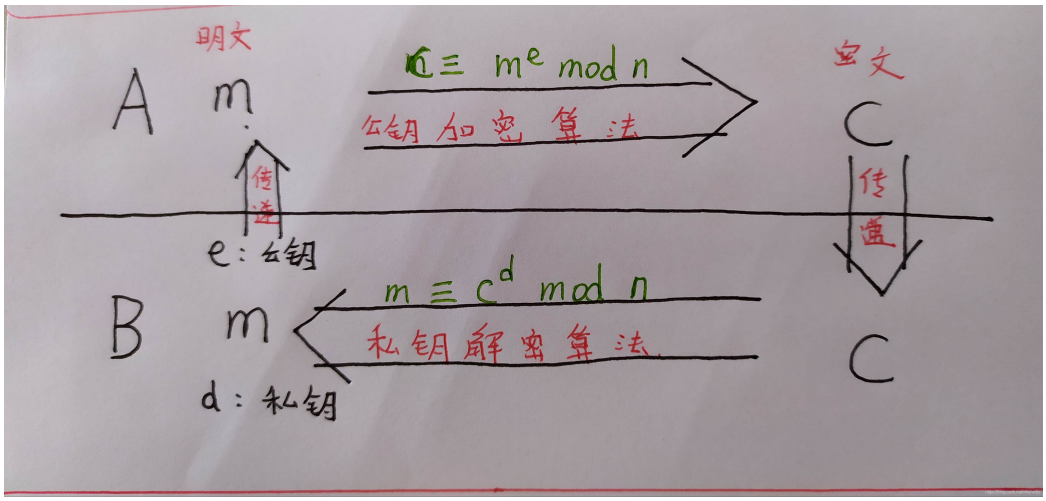
安装命令： pip3 install gmpy2-2.0.8-cp37-cp37m-win_amd64.whl

pycryptodome: pip3 install pycryptodome

libnum: pip3 install libnum

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

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：

$$c \equiv m^e \pmod n$$

解密算法：

$$m \equiv c^d \pmod n$$

还有一点必须要了解，就是 签名消息 其实就是一个校验码，确保密文在传播过程中没有被篡改过

签名消息（攻防世界中的一个题涉及到这个知识，所以在这里提一下）

RSA也可以用来为一个消息署名。假如Alice想给Bob传递一个署名的消息的话，那么她可以为她的消息计算一个散列值（Message digest），然后用她的私钥“加密”（如同前面“加密消息”的步骤）这个散列值并将这个“署名”加在消息的后面。这个消息只有用她的公钥才能被解密。Bob获得这个消息后可以用Alice的公钥“解密”（如同前面“解密消息”的步骤）这个散列值，然后将这个数据与他自己为这个消息计算的散列值相比较。假如两者相符的话，那么Bob就可以知道发信人持有Alice的私钥，以及这个消息在传播路径上没有被篡改过。

二，CTF 中的 常见的十种类型：

1，已知 p，q，e 求 d?

（ed 除以 (q-1)(p-1) 的 余数 为 1 ）

```
import gmpy2
p = 38456719616722997
q = 44106885765559411
e = 65537

s = (p-1)*(q-1)
d = gmpy2.invert(e,s)
print ("dec: " + str(d))
print ("hex: " +  hex(d))
```

2，已知 n（比较小），e 求 d?

（n = q * p , ed 除以 (q-1)(p-1) 的 余数 为 1 ）**(n往往是一个 1024bit 的超大数，很难分解为两个 质数)**

n的分解 用yafu 中的 factor（n ）命令 **（yafu kali自带，也有win版）**

3，已知 公钥（n, e） 和 密文 c 求 明文 m?

方法一：（n,e不太大的情况下）

首先将 n 用 yafu分解为 q 和 p

再利用脚本 ：

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

c = 0x6cd55a2bbb49dfd2831e34b76cb5bdfad34418a4be96180b618581e9b6319f86
n = 10853984726857399027523402435467243724652508507660551696032000572274158989864l
#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
```

方法二：（n , e 比较大的时候）

直接利用 RsaCtfTool进行爆破：

利用 n,e生成公钥文件 test.pem：

```
python RsaCtfTool.py --createpub --n 46065781....  --e 3546111...  > test.pub
```

```
[RsaCtfTool] ^_^ Orz python RsaCtfTool.py --createpub --n 460657813884289609896372056585544172485318117026246263899
744329237492701820627219556007788200590119136173895989001382151536006853823326382892363143604314518686388786002989 2
488008148612485950753262770996453386949770974591685308987760072936957281019760694239716965242377552271870614182028 4
991147912479399072259 7 --e 3546111024413075720565721818279258991983453502287537309310893932754639165444566268942454
150961078344657784095323731871253185546147225993017915289162128393681210660355410088082615345005860236527677122716 2
578520428096468800468032830012484968047710530251937737009257810782711682139182621097232037761496754782761 9 > test.p
em
```

再利用 下面的 使用 公钥爆破

4，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？

方法一：(key.pem 和 cipher.bin)

直接用 RsaCtfTool进行破解：

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

```
root@kali:~/document/RsaCtfTool# python RsaCtfTool.py --publickey key.pem --uncipherfile cipher.bin
[!] Warning: Wiener attack module missing (wiener_attack.py) or SymPy not installed?
[+] Clear text : Congratulations! Here is a treat for you: flag{how_d0_you_7urn_this_0n?}
root@kali:~/document/RsaCtfTool#
```

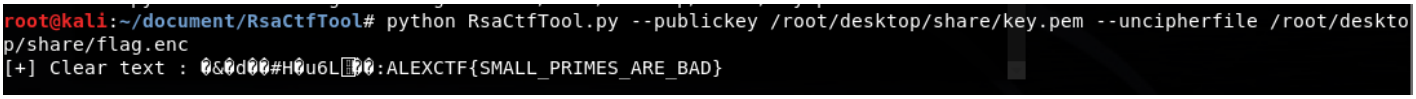
方法二：（flag.b64 和 key.pub）

先处理flag.b64（将flag.b64中的内容进行解 base64操作）

使用 notepad++ 打开 flag.b64文件使用 插件中的 MIME Tools 中的 base64 decode 将文件内容解密，然后另保存为 flag.enc 文件

然后使用 RsaCtfTool 工具进行破解：

```
python RsaCtfTool.py --publickey /root/desktop/share/key.pem --uncipherfile /root/desktop/share/flag.enc
```



方法三：

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 275127860351348928173285174381581152299 -q 3195763168144789498
```

最后 用private.pem文件 解密 flag.enc文件

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

5，有私钥 private.pem 和密文 flag.enc

方法一：利用 rsactftool。

```
python RsaCtfTool.py --private private.pem  --uncipherfile flag.enc
```

方法二：利用 openssl：

```
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：给了e，n，c,由于特别大，没法直接用质因数分解求得 q， p

还给了 phint = d % （p - 1） 其实 phint = dp

qhint = q % （p - 1） 其实 qhint = dq

(2019山东省赛“深思杯”中 被加密的消息 一题 跟本题 的思路一样)

```
import gmpy2
import libnum
e=65537
n=169697521655091326276302669687488543303407016921254276195598364883502982347355714803530786149755803784673
dp=178162577529102887026968525752110809032954301272870546778254691395153764262362176924644112218994867137 49
c=0x6c78dcee37830f3ec4ab4989d40fbb595060b3fbc395d52ad26defc13372c1a3948c5388f4e450e46e016c7803133d6881e5efc

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)))
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python37\python3.exe C
flag{0039f13c24789a62747f7b7e94d6ac79d163213b}
```

7，已知n（非常大）,e,d求p,q（无法直接 从n分解）

一看这个标题你就应该有个觉悟，n一定无法直接分解得到p和q。

题目: 10-存货5

题目给出了2个文件,1个是加密脚本chall.py，1个是加密后输出的内容output.txt

分析一下加密脚本：

```
from gmpy2 import invert
from md5 import md5
from secret import p, q

e = 65537
n = p*q
phi = (p-1)*(q-1)
d = invert(e, phi)

print n, e, d
print "Flag: flag{%s}" %md5(str(p + q)).hexdigest()
```


加密脚本真的是很简单啊，flag就是str(p+q)进行md5运算之后的得到的字符串，从output.txt中可以得到n,e,d

现在的关键问题就是求出p和q来

(python 2)

```
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 = 16352578963372306131642407541567045533766691177138375676491913897592458965544068296813122740126583082
 e = 65537
 d = 94599283799736674301380685280594381390923686253390792532895605779853044350622131213982318758322648944
 p,q = getpq(n,e,d)
        print p
        print q
        print "Flag: flag{%s}" %md5(str(p + q)).hexdigest()
if __name__ == '__main__':
 main()
```

8，提取私钥中的信息

```
python RsaCtfTool.py --key private.pem  --dumpkey
```

9,利用公钥pub.key/pub.pem文件生成 私钥文件

```
python RsaCtfTool.py --publickey pubkey.pem  --private > private.pem


或

python2 RsaCtfTool.py --publickey pub.key  --private > private.key
```

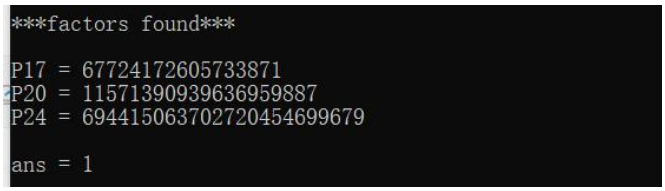
10，n分解出多个不同的因子时 ，求明文m

15年山东大学生爱好者线上赛

```
n= 544187306850902797629107353619267427694837163600853983242783
e= 39293
c= 43925489581832041340882702239805368586734326797171233201 1972
m=???
```

对n进行质因数分解，得到了3个质因数，（这里知道欧拉公式的性质的话 就很好解)

φ(x * y * z) = φ(x) * φ(y) * φ(z)=(x-1)(y-1)(z-1)



(python2)

```
import gmpy2
from Crypto.Util.number import long_to_bytes

n= 544187306850902797629107353619267427694837163600853983242783
e= 39293
c= 43925489581832041340882702239805368586734326797171233201 1972
p1 = 67724172605733871
p2 = 11571390939636959887
p3 = 6944150637027204546 99679
phi = (p1-1)*(p2-1)*(p3-1)
d = gmpy2.invert(e, phi)
m = pow(c, d, n)
print long_to_bytes(m)
```

三，爆破攻击方法：

1，低加密指数分解攻击（比如 e=2,e=3）

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。

下面就是e选取的太小导致存在的安全问题：

（1）e=2把密文c开平方求解

RSA加密，由于e只有2，相当于把明文m平方而已，得到的c也比n小很多。尝试把c开根号看能否得到明文。一般的python开根号方法精度较低，对大整数开出来的根号准确度低。

发现使用gmpy2库可以对大整数开根号。

题目：01-西湖论剑rsa

```
已知:

e=2

c=92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321676580866698866

求明文m?
```

```
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}
```

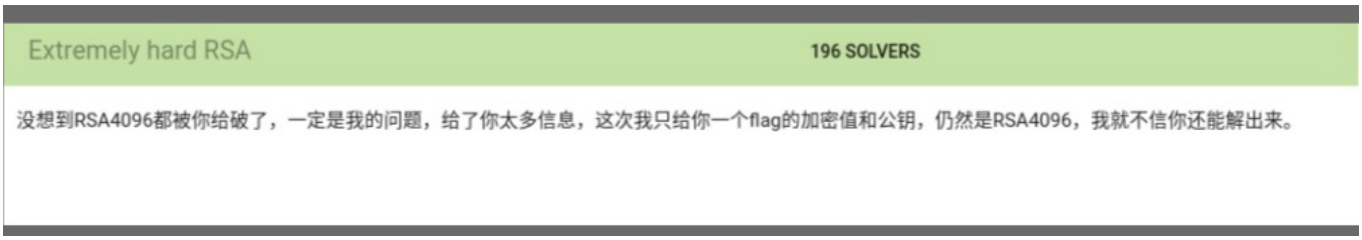
（2）e=3 小明文攻击

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

2.Roll按行加密（加密是按行进行的）

顾名思义，这里的的加密是按行进行的。

题目: 04-实验吧---RSAROLL

应该按行进行解密

根据给出的文件这里猜测(真的就是猜测，太坑了):

n 为920139713

e 为19

首先把加密的部分另存为一份文件roll.txt

```
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
```


解密脚本：

```
import gmpy2
from Crypto.Util.number import long_to_bytes

n = 920139713
p = 49891
q = 18443
e = 19
phi = (p-1)*(q-1)
d = gmpy2.invert(e,phi)
m = ""
with open('roll.txt','r') as f:
    for c in f.readlines():
        c = int(c)
        #m += long_to_bytes(pow(int(c), d, n))

        md = str(pow(c, d, n))
        m += chr(int(md))
print(m)

#flag{13212je2ue28fy71w8u87y31r78eu1e2}
```

3.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

N is
18674375108313094928585156581138941368570022222190945461284402673204018075354069827186 08

e is 65537

message is
0x8BD7BF995BF9E16A0D04ADB49A2411C74FFDB0DB4F35DB3A79A1B44691947C9824085BC4CA5F7F

N is
20071978783607427283823783012022286910630968751671103864055982304683197064862908267206 04

e is 65537

message is
0x8C3CF3161AA3E37831030985C60566A7604688B73E5B1D3B36E72EF06ED4F71289EFE80E0D94BD75

求明文?

由于 不能直接 分解 n ， 只能先找出 n1，n2 的公因数作为 q ， 再拿n1 ， n2除以 q 得到 p1 和p2

然后在解密 ：（这个只是个 例子，并且是一个 python2 脚本 ， 最后输出的结果太大导致 程序崩掉, 借鉴一下思路即可）

```
#!/usr/bin/python
#coding:utf-8
#@Author:醉清风

import gmpy2
from Crypto.Util.number import long_to_bytes

lines = open('tmp.txt','r').readlines()

c1 = int(lines[2],16)
c2 = int(lines[6],16)
n1 = int(lines[0])
n2 = int(lines[4])

p1 = gmpy2.gcd(n1, n2)
assert (p1 != 1)
p2 = n1 / p1
p3 = n2 / p1
e = 0x10001
d1 = gmpy2.invert(e, (p1 - 1) * (p2 - 1))
d2 = gmpy2.invert(e, (p1 - 1) * (p3 - 1))
m1 = pow(c1, d1, n1)
m2 = pow(c2, d2, n2)
print long_to_bytes(m1)+long_to_bytes(m2)
```

判定 x 和 y 是否互素：

```
#判断两个数是否互素

def gcd(a, b):  # 判断来两个数是否互素,辗转相除法
    if (b == 0):
        return a
    else:
        return gcd(b, a % b)


def main():
    x = 17  # x,y的值根据需要修改即可
    y = 65537
    if gcd(x, y) == 1:  # 如果两个数的最大公约数是1，那么两数互素。
        print(str(x) + " " + str(y) + "  两个数互素")
    else:
        print(str(x) + " " + str(y) + " 两个数不互素")


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

4.共模攻击（m，n相同；e，c不同，且e1 和 e2互质）

适用情况：明文m、模数n相同，公钥指数e、密文c不同，gcd(e1,e2)==1也就是e1和e2互质。

对同一明文的多次加密使用相同的模数和不同的公钥指数可能导致共模攻击。

题目: 06-Jarvis OJ -Crypto-very hard RSA

前几题因为N太小，都被你攻破了，出题人这次来了个RSA4096，是否接受挑战就看你了。

这个题目就比较有意思了，4096位的RSA加密，要不是这里存在共模攻击说不定你死活都解不开 哈哈 要是有量子计算机的话说不定能解开。

题目给出了3个文件，其中2个明文是分开加密后的密文，另一个veryhardRSA.py则是加密脚本，我们通过分析加密脚本进而写出解密脚本。

查看加密脚本发现明文m、模数n相同，但是公钥指数e1和e2不同，而且e1与e2互素（上面给过判断2数是否互素的脚本）

（python 2脚本）(n是4096位的，不可能直接分解，除非用天河二号超级计算机)

```
import gmpy2
from Crypto.Util.number import long_to_bytes

e1 = 17
e2 = 65537
n = 0x00b0bee5e3e9e5a7e8d00b493355c618fc8c7d7d03b82e409951c182f398dee3104580e7ba70d383ae5311475656e8a964d38
c1=int(open('./flag.enc1','rb').read().encode('hex'),16)
c2=int(open('./flag.enc2','rb').read().encode('hex'),16)


_, r, s = gmpy2.gcdext(e1, e2)

m = pow(c1, r, n) * pow(c2, s, n) % n
print long_to_bytes(m)
```

5.低解密指数攻击（e过大或过小，一般e过大时使用）

在RSA中d也称为解密指数，当d比较小的时候，e也就显得特别大了。

适用情况：e过大或过小（一般e过大时使用）

在e过大或过小的情况下，可使用算法从e中快速推断出d的值，进而求出m

具体我也没看懂，参照：https://www.tr0y.wang/2017/11/06/CTFRSA/index.html#低解密指数攻击

题目: 我不是个人

n =
46065781388428960989637205658554417248531811702624626389974432923749270182062721955600 77

e =
35461110244130757205657218182792589919834535022875373093108939327546391654445662689424 54

c =
38230991316229399651823567590692301060044620412191737764632384680546256228451518238842 96

求明文m?

首先需要需要下载工具rsa-wiener-attack（附件里面的rsa工具中有）：git clone https://github.com/pablocelayes/rsa-wiener-attack

然后把exp.py放入这个目录中运行即可（Python2）实测没有问题

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

import gmpy2
from Crypto.PublicKey import RSA
import ContinuedFractions, Arithmetic
from Crypto.Util.number import long_to_bytes

def wiener_hack(e, n):
    # firstly git clone https://github.com/pablocelayes/rsa-wiener-attack.git !
    frac = ContinuedFractions.rational_to_contfrac(e, n)
    convergents = ContinuedFractions.convergents_from_contfrac(frac)
    for (k, d) in convergents:
        if k != 0 and (e * d - 1) % k == 0:
            phi = (e * d - 1) // k
            s = n - phi + 1
            discr = s * s - 4 * n
            if (discr >= 0):
                t = Arithmetic.is_perfect_square(discr)
                if t != -1 and (s + t) % 2 == 0:
                    print("Hacked!")
                    return d
    return False
def main():
    n = 46065781388428960989637205658554417248531811702624626389974432923749270182062721955600778820059011 9
    e = 35461110244130757205657218182792589919834535022875373093108939327546391654445662689424541509610783 4
    c = 38230991316229399651823567590692301060044620412191737764632384680546256228451518238842965221394711 8
    d = wiener_hack(e, n)
    m = pow(c,d,n)
    print long_to_bytes(m)
if __name__=="__main__":
    main()
```

6，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。

例如下面的就是e=k=3

$$c_1 \equiv m^e \bmod n_1$$

$$c_2 \equiv m^e \bmod n_2$$

$$c_3 \equiv m^e \bmod n_3$$

使用不同的模数n，相同的公钥指数e加密相同的信息。就会得到多个(m^e) ==ci (mod ni)，将(m^e)视为一个整体M，

这就是典型的中国剩余定理适用情况。按照本文的中国剩余定理小节容易求得m^e的值，当e较小时直接开e方即可，可使用

题目：12-Jarvis OJ -2018强网杯nextrsa-Level9

题目给出

n1,n2,n3,

c1,c2,c3,

e,

求明文m的值。（附件里面也有题目，截图清晰一点,这里就没给数据）

```
e = 3
c1 =
0x9e84763bdbe246fad0a9cd52fda6233e6128a6210efaf3e6dea4fe272f78ad1f8f5cc7022f62f4f542341128e42d6fd10e67c5f96edbd243917c0151289f7228
26b69cab36cc61e4ac094832b4299bbaf4630b722a0fb4f1997053be97e926f94afb55a0bb6ef00ab694e2f595d9eb8ca96c49f5cbbe194529f68a1aaf6f515148
74dc68e255c363579a80d27ce5090873ac719ba59f2492c91fd28bcce26b6a02bae005cbbd2a4cfe5b93442be8664d2313d412e7e09f545c64b7b74bbc408b6e57
aca9992ede644d856eb4cfcda562a75743e4b491L
c2 =
0x9817fdc7b31a8f9cde1794096d3aa2bc6fe06fe34d4b7c9ca9a77982adf67fd4a7e636659553f4168a16757dc3a75e54ff850b9a94a5270f4f75502c7055a3a3
7253c0f343f127e0ff162a349bb14eb4c1453fc6daace19bba4940d77c435686ef3b59f732072cde2e148d1a64f9682b3f1ceb9a000d87e180a1f9eb20c59dbebc
c878a45d99909ab2a3e4cdb74aa68890c941315ae289d6667200c53f9a32c8a64bfc74e62898ac03c460f945a13f11ee28860a3cd07526c30aa92eb89442a76549
a586db719f8610eb5d4a8f5bd1e481b5ef6e96efL
c3 =
0xb0c5ee1ac47c671c918726287e70239147a0357a9638851244785d552f307ed6a049398d3e6f8ed373b3696cfbd0bce1ba88d152f48d4cea82cd5dafd50b9843
806202e45821ad6622935393cd996968fc5e251aa3539ed593fe893b15d21ecbe6893eba7fe77b9be935ca0aeaf2ec53df7c7086349eb12792aefb7d34c31c18f3
a5d7ace90a282facf2d2760e6b5d98f0c70b23a6db654d10085be9dcc670625646a153b52c6c710efe8eb876289870bdd69cb7b45813e4fcfce815d191838926e9
2e077a5ee720caedc1617bf6a0bb072bbd2dab0L

n1 =
0x43d819a4caf16806e1c540fd7c0e51a96a6dfdbe68735a5fd99a468825e5ee55c4087106f7d1f91e10d50df1f2082f0f32bb82f398134b0b8758353bdabc5ba2
7d073f346d6adb2684a9d28b658dddc75b3c5d10a22a3e85c6c12549d0ce7577e79a068405d3904f3f6b9cc408c4cd8595bf67fe672474e0b94dc99072caaa4f86
52adef71649684f1a72c910ec5ca7909cc10aef85d43a57ec91f096a2d4794299e967fcd5add6e9cfb5baf7751387e24b93dbc1f37315ce573dc063ecddd4ae6fb
590c8b2f5bd06dd392fbc51e5d059cffbcb85555L
n2 =
0x60d175fdb0a96eca160fb0cbf8bad1a14dd680d353a7b3bc77e620437da70fd9153f7609efde652b825c4ae7f25decf14a3c8240ea8c5892003f1430cc88b0de
e23fbffb7cfe431ff3d802f5a54ab76257a86aeec1cf47d482fec970fc27c5b376fbf2cf993270bba9b78174395de3346d4e221d1eafdb8eecc8edb953d1ccaa5f
a6e72ce4fee37e77faaf5597d780ad5f0a7623edb08ce76264f72c3ff17afc932f5812b10692bcc941a18b6f3904ca31d038baf3fc1968d1cc0588a656d0c53cd5
7f524c2027adce84fd4d0e018dc88ca4d5d26867L
n3 =
0x280f992dd63fcabdcb739f52c5ed1887e720cbfe73153adf5405819396b28cb54423d196600cce76c8554cd963281fc4b153e3b257e96d091e5d99567dd1fa9a
3822316d751e788dc935d63916075530d7fb89cbec9b02c01aef19c39b4ecaa1f7fe2faf990aa938eb89730eda30558e669da54
ed1c74f380fd29ebdd28618d60c36e6973fc87c066cae05e9e270b5ac25ea5ca0bac5948de0263d8cc89d91c4b574202e71811
cfbe3370cd7b4187c023c35671de3888a1ed1303L
```

（python 2）

```
#!/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 = [856452939859749605250987551348123451190557129360825359177435234523787673329310883120372300895836722448
c = [200109715577899319481307989830302019500384502691441045328210306679244007888699202385797295146726302218


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

题目: 存货6

怕你觉得上面解出来的m是纯数字看着不舒服，特意给你找了个带flag的题型，还有就是这个e=9也不大呀。

题目给出n1,n2,n3,c1,c2,c3,e,求明文m的值。

```
e = 3
c1 =
0x9e84763bdbe246fad0a9cd52fda6233e6128a6210efaf3e6dea4fe272f78ad1f8f5cc7022f62f4f542341128e42d6fd10e67c5f96edbd243917c0151289f7228
26b69cab36cc61e4ac094832b4299bbaf4630b722a0fb4f1997053be97e926f94afb55a0bb6ef00ab694e2f595d9eb8ca96c49f5cbbe194529f68a1aaf6f515148
74dc68e255c363579a80d27ce5090873ac719ba59f2492c91fd28bcce26b6a02bae005cbbd2a4cfe5b93442be8664d2313d412e7e09f545c64b7b74bbc408b6e57
aca9992ede644d856eb4cfcda562a75743e4b491L
c2 =
0x9817fdc7b31a8f9cde1794096d3aa2bc6fe06fe34d4b7c9ca9a77982adf67fd4a7e636659553f4168a16757dc3a75e54ff850b9a94a5270f4f75502c7055a3a3
7253c0f343f127e0ff162a349bb14eb4c1453fc6daace19bba4940d77c435686ef3b59f732072cde2e148d1a64f9682b3f1ceb9a000d87e180a1f9eb20c59dbebc
c878a45d99909ab2a3e4cdb74aa68890c941315ae289d6667200c53f9a32c8a64bfc74e62898ac03c460f945a13f11ee28860a3cd07526c30aa92eb89442a76549
a586db719f8610eb5d4a8f5bd1e481b5ef6e96efL
c3 =
0xb0c5ee1ac47c671c918726287e70239147a0357a9638851244785d552f307ed6a049398d3e6f8ed373b3696cfbd0bce1ba88d152f48d4cea82cd5dafd50b9843
806202e45821ad6622935393cd996968fc5e251aa3539ed593fe893b15d21ecbe6893eba7fe77b9be935ca0aeaf2ec53df7c7086349eb12792aefb7d34c31c18f3
a5d7ace90a282facf2d2760e6b5d98f0c70b23a6db654d10085be9dcc670625646a153b52c6c710efe8eb876289870bdd69cb7b45813e4fcfce815d191838926e9
2e077a5ee720caedc1617bf6a0bb072bbd2dab0L

n1 =
0x43d819a4caf16806e1c540fd7c0e51a96a6dfdbe68735a5fd99a468825e5ee55c4087106f7d1f91e10d50df1f2082f0f32bb82f398134b0b8758353bdabc5ba2
7d073f346d6adb2684a9d28b658dddc75b3c5d10a22a3e85c6c12549d0ce7577e79a068405d3904f3f6b9cc408c4cd8595bf67fe672474e0b94dc99072caaa4f86
52adef71649684f1a72c910ec5ca7909cc10aef85d43a57ec91f096a2d4794299e967fcd5add6e9cfb5baf7751387e24b93dbc1f37315ce573dc063ecddd4ae6fb
590c8b2f5bd06dd392fbc51e5d059cffbcb85555L
n2 =
0x60d175fdb0a96eca160fb0cbf8bad1a14dd680d353a7b3bc77e620437da70fd9153f7609efde652b825c4ae7f25decf14a3c8240ea8c5892003f1430cc88b0de
e23fbffb7cfe431ff3d802f5a54ab76257a86aeec1cf47d482fec970fc27c5b376fbf2cf993270bba9b78174395de3346d4e221d1eafdb8eecc8edb953d1ccaa5f
a6e72ce4fee37e77faaf5597d780ad5f0a7623edb08ce76264f72c3ff17afc932f5812b10692bcc941a18b6f3904ca31d038baf3fc1968d1cc0588a656d0c53cd5
7f524c2027adce84fd4d0e018dc88ca4d5d26867L
n3 =
0x280f992dd63fcabdcb739f52c5ed1887e720cbfe73153adf5405819396b28cb54423d196600cce76c8554cd963281fc4b153e3b257e96d091e5d99567dd1fa9a
3822316d751e788dc935d63916075530d7fb89cbec9b02c01aef19c39b4ecaa1f7fe2faf990aa938eb89730eda30558e669da5
ed1c74f380fd29ebdd28618d60c36e6973fc87c066cae05e9e270b5ac25ea5ca0bac5948de0263d8cc89d91c4b574202e718
cfbe3370cd7b4187c023c35671de3888a1ed1303L
```

（python2 ）

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

import gmpy2
import time
from Crypto.Util.number import long_to_bytes

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 = 9
n = [1427824243688496747719766719551761878349324170274680064790380583855500424222801587265617122592056166266
c = [8503386841878430857367370996070077735031442642767762731969734681112374234235907217022042887495299698844
data = zip(c, n)
x, n = CRT(data)
m = gmpy2.iroot(gmpy2.mpz(x), e)[0].digits()
print long_to_bytes(m)



#Tr0y{e=3_1s_danger0us!}
```

参考了几篇写得很好的的文章：

1，https://zhuanlan.zhihu.com/p/76006823

2，https://cloud.tencent.com/developer/article/1076086

3，https://willv.cn/2018/07/21/RSA-ATTACK/

4，https://err0rzz.github.io/2017/11/14/CTF%E4%B8%ADRSA%E5%A5%97%E8%B7%AF/