BUUCTF RSA题目全解2

原创

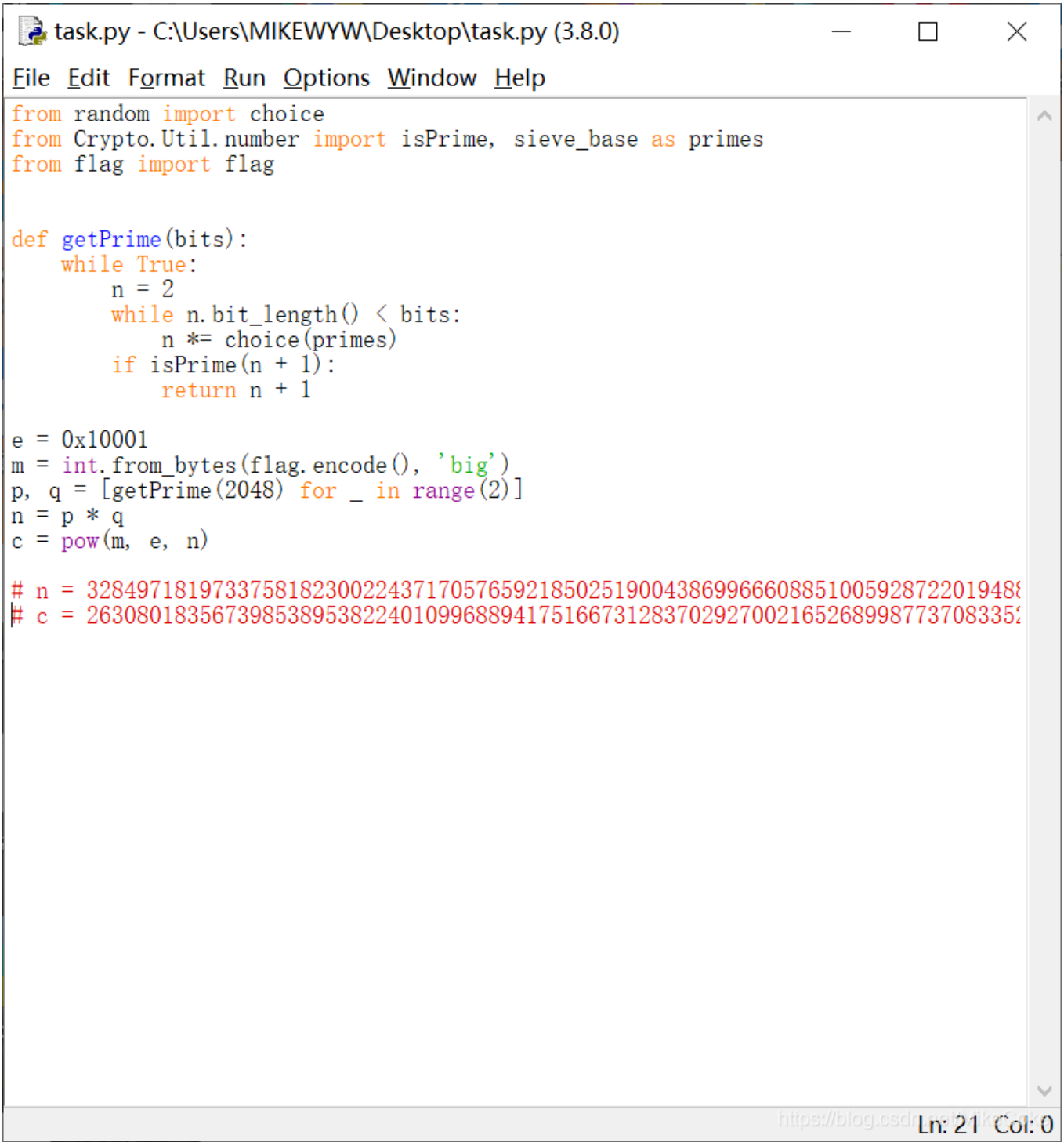
宁嘉 于 2020-08-12 15:15:37 发布 2947 收藏 16

分类专栏： RSA加密 文章标签： 密码学

本文链接：https://blog.csdn.net/MikeCoke/article/details/107206707

RSA

1.[NCTF2019]childRSA



别人的解法，利用Crypto.Util.number 中的sieve_base

我的解法：

解题思路：

1.**py文件给了，n , c , e。为了求m 由算法pow(c,d,n)，可知要先求出d来**

2. **由算法**
   gmpy2.invert(e,N) # N = (q-1)*(p-1)
   可知我们要先求出p，q来，即分解n

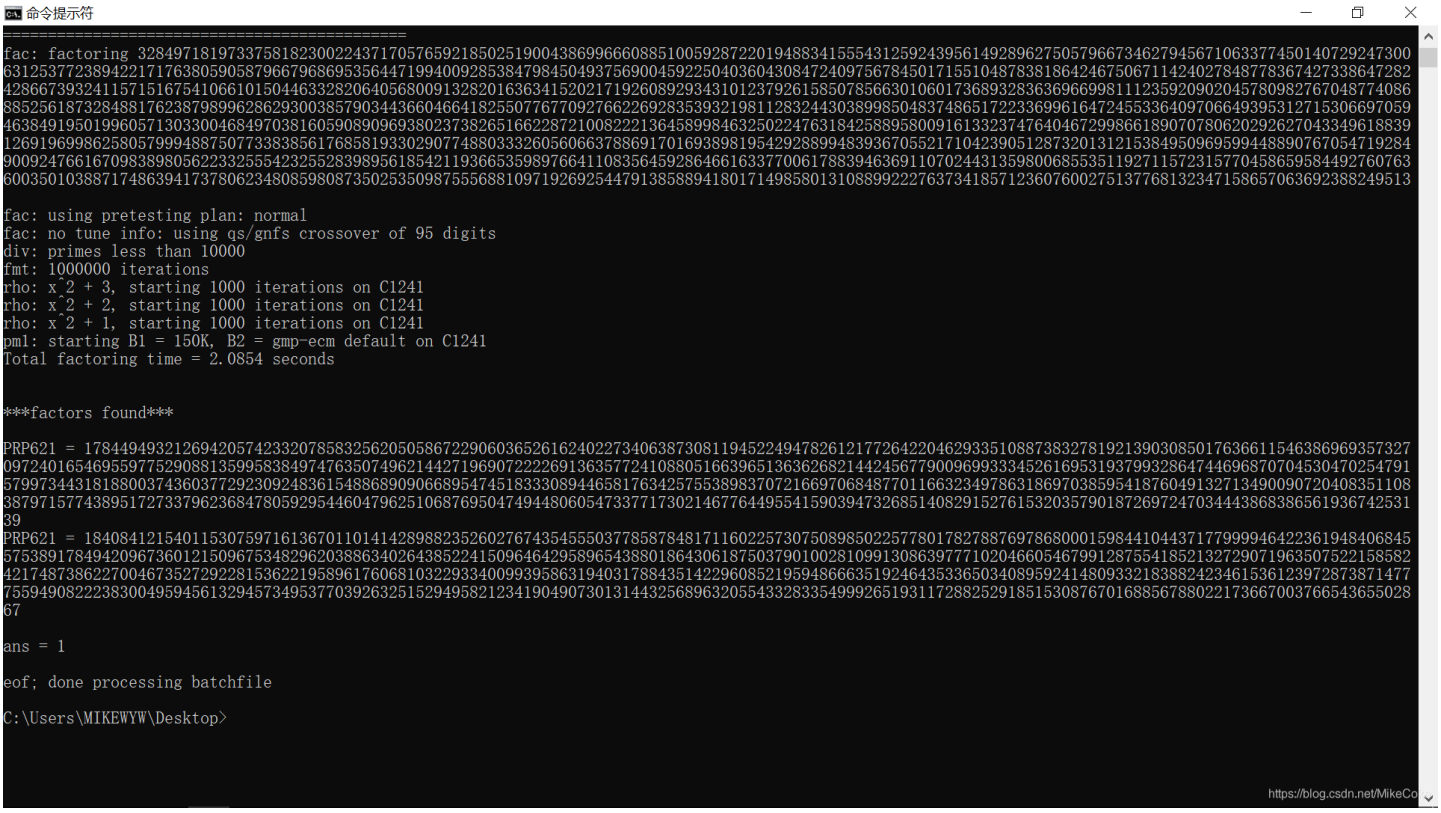
用yafu进行素数分解，先把模n新建一个txt文件，文件末尾要进行换行

进入cmd，命令 `cd Desktop` 进入桌面

命令

`yafu-x64 "factor(@)" -batchfile 1.txt` 进行10进制大整数分解

当数比较小时 用命令 `yafu-x64 factor(n)`



得到

```
import gmpy2 as gp
import libnum

n=32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579667
34627945671063377450140729247300631253772389422171763805905879667968695356447199400928538479845049375690045922504
40360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044463
32820640568009132820163634152021719260892934310123792615850785663010601736893283636966998111235920902045780982767
70487740868852561873284881762387989962862930038579034436604664182550776770927662269283539321981128324430389985048
37486517223369961647245533640970664939531271530669705946384919501996057130330046849703816059089096938023738265166
22872100822213645899846325022476318425889580091613323747640467299866189070780620292627043349618839126919699862
58057999488750773383856176858193302907748803332605606637886917016938981954292889948393670552171042390512873201312
15384950969599448890767054719284900924766167098389805622332555423255283989561854211936653598976641108356459286466
16337700617883946369110702443135980068553511927115723157704586595844927607636003501038871748639417378062348085
98087350253509875556881097192692544791385889418017149858013108899222763734185712360760027513776813234715865706369
2388249513
c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108

p=17844949321269420574233207858325620505867229060365261624022734063873081194522494782612177264220462933510887383
27819213903085017636611546386969357327097240165469559775290881359958384974763507496214427196907222269136357724108
88051663965136362682144245677900969933345261695319379932864744696870704530470254791579973443181880037436037729230
92483615488689090668954745183330894465817634257553898370721669706848770116632349786318697038595418760491327134900
90072040835110838797157743895172733796236847805929544604796251068769504749448060547337717302146776449554159039473
26851408291527615320357901872697247034443868386561936742531391
q = 18408412154011530759716136701101414289882352602767435455503778587848171160225730750898502257780178278876978680
00015984410443717799994642236194840684557538917849420967360121509675348296203886340264385224150964642958965438801
86430618750379010028109913086397771020466054679912875541852132729071963507522158582421748738622700467352729228153
62219589617606810322933400993958631940317884351422960852195948666351924643533650340895924148093321838824234615361
23972873871477755949082223830049594561329457349537703926325152949582123419049073013144325689632055433283354999265
19311728825291851530876701688567880221736670037665436550286 7

N=(p-1)*(q-1)

e = 65537


d = gp.invert(e,N)

m = pow(c,d,n)

print(libnum.n2s(m))
```

flag{Th3r3_ar3_1ns3cure_RSA_m0duli_7hat_at_f1rst_gl4nce_appe4r_t0_be_s3cur3}

2.[HDCTF2019]bbbbbbrsa

enc.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
p = 177077389675257695042507998165006460849
n =
374218295098877962748971622493673294009886471456133253673379680633413727260 61
c =
==gMzYDNzIjMxUTNyIzNzIjMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwIjN2QzM5ADMwIDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM
```

第 3 行，第 109 列 100% Windows (CRLF) UTF-8

encode (1).py - C:\Users\MIKEWYW\Desktop\encode (1).py (3.8.0)

File Edit Format Run Options Window Help

```
from base64 import b64encode as b32encode
from gmpy2 import invert,gcd,iroot
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex
import random

flag = "******************************"

nbit = 128

p = getPrime(nbit)
q = getPrime(nbit)
n = p*q

print p
print n

phi = (p-1)*(q-1)

e = random.randint(50000,70000)

while True:
        if gcd(e,phi) == 1:
```

```
        if gcd(e,phi) == 1:
            break;
        else:
            e -= 1;

c = pow(int(b2a_hex(flag),16),e,n)

print b32encode(str(c))[::-1]

# 23737406995293649917635893242000934662067855618361018403816222372255122346 32
```

Ln: 2 Col: 0

解题思路：

1.第一张图给了 **p，n，c，所以我们能够求出q**

2.**为了求出明文m,我们先要求出d来**

3.第二种图告诉我们 **e 的范围在(50000,70000)**，我们可以对e进行遍历，爆破求出d

4. **对于每个符合条件的 e，都会得到一个与之对应的 m，我们知道 一般的 m中都会含有flag字符，所以用search()进行查找**

```
import gmpy2 as gy
import re
import libnum

p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
c =23737406995293649917635893242000934662067855618361018403816222372255122346 32

q = n // p

N = (p-1)*(q-1)

for e in range(50000,70000):
    try:
        d = gy.invert(e,N)
        flag = libnum.n2s(pow(c,d,n))
        if re.search('flag',flag):          #扫描整个字符串，返回第一个成功的匹配
            print(flag)
    except:
        pass
```

`flag{rs4_1s_s1mpl3!#}`

3.SameMod

共模攻击，不多讲了，数学原理

附件.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

{626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9,773}
{626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9,839}

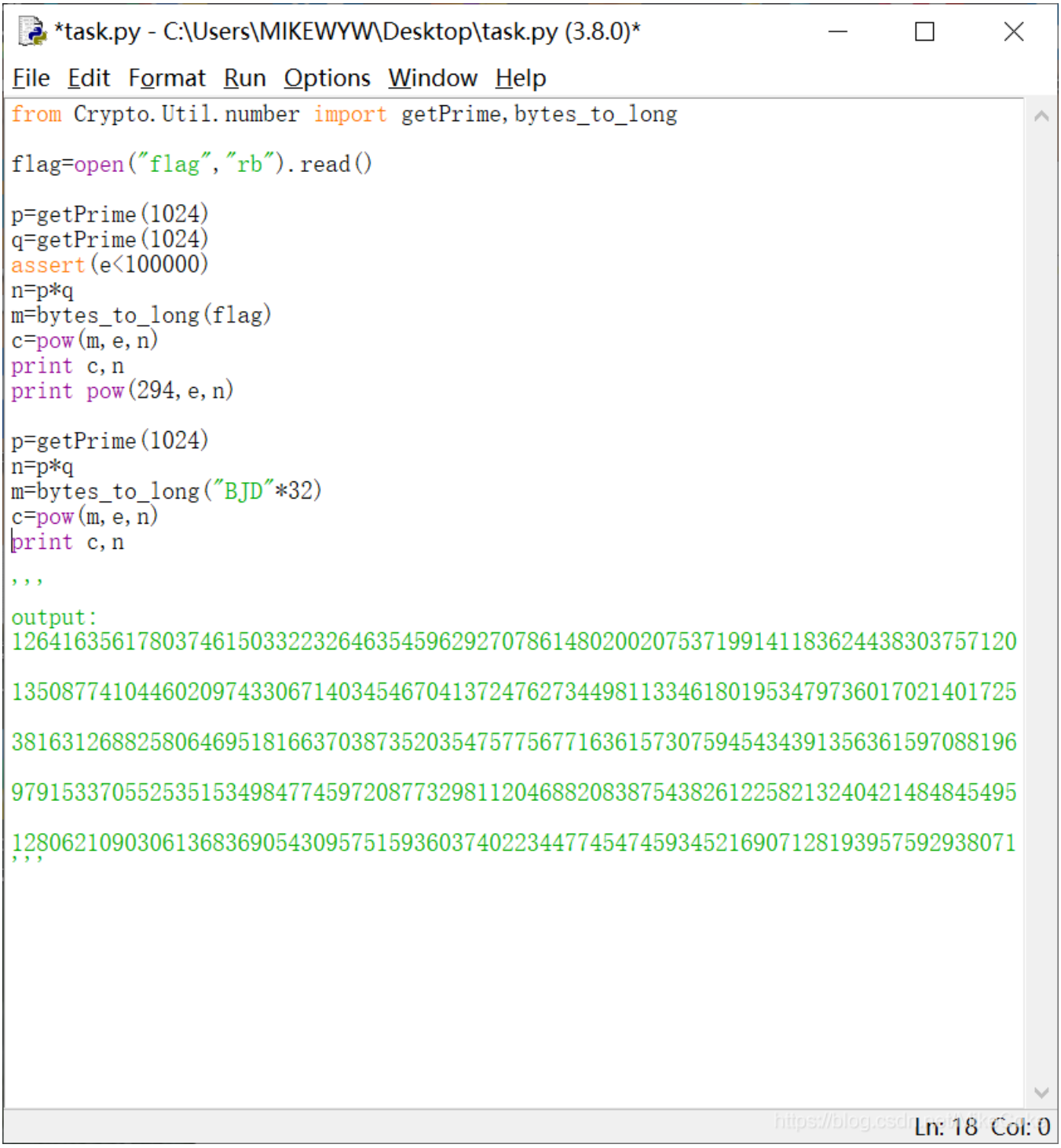
message1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311153734 9
message2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323830 9535

第 1 行，第 1 列 100% Unix (LF) UTF-8

```
import gmpy2 as gp

def exgcd(a, b):
    if b==0:
        return 1, 0, a
    x2, y2, r = exgcd(b, a%b)
    x1 = y2
    y1 = x2-(a//b)*y2
    return x1, y1, r

def get_flag(string):
    flag=''
    i=0
    j=1
    while i < len(string):
        if int(string[i:i+j]) >= 33 and int(string[i:i+j]) <=126:
            flag+=chr(int(string[i:i+j]))
            i=i+j
            j=1
        else:
            j+=1
    print(flag)

if __name__ == '__main__':

    e1=773
    e2=839
    n=626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9
    message1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398111315531115373 49
    message2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323830953 5
    r1, r2, t = exgcd(e1, e2)
    m = gp.powmod(message1, r1, n) * gp.powmod(message2, r2, n) % n
    get_flag(str(m))
```

`flag{whenwethinkitispossible}`

4.[BJDCTF2020]RSA



解题思路：

1.题目告知输出了两种不同的密文 **c** 和 **n** 值，但值 **n1** 和 **n2** 共用相同的 **p**，所以用gcd()算法求最大公约数可以得到 **p值**，进而求出 **q**。

2.对 e 进行遍历，爆破得到其值。

```python
from gmpy2 import*
import libnum

n1 =13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037
c1 =12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755798009656547738616399025300123043766255518596149348930444599820675230046423373053051631932557230849083426859490183732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018698992935818206109087568166097392314105717555482926141030505639571708876213167112187962584484065321545727594135175369233925922507794999607323536976824183162923385005669930403448853465141405846835919842908469787547341752365471892495204307644586161393228776042015534147913888338316244169120

n2 =12806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
c2 = 97915337055253515349847745972087732981120468820838754382612258213240421484845495472248708665806140879522380502220299761352201473698345212107386005485130234351775673270102666706276590627762687921545793633079969881275597305755762093017277885911653857120710042499083850825512761663733449968005864541178692530236879041476824861180935816019755436925545867545010945798769874958463055117757749204340365641996828516353682381981757353135649723615434268991452532167380792545865185476851239635538974086327014877536274444811558163962932636234216054850003500015609721544688125105550546571385417391314204097638250043518544252172

pzm = 38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889859846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501 8

p = gcd(n1,n2)
#p =99855353761764939308265951492116976798674681282941462516956577712943717850048051273358745095906207085170915794187749954588685850452162165059831749303473106541930948723000882713453679904525655327168665295207423257922666721077747911860159181041422993030618385436504858943615630219459262419715816361781062898911

q =n1//p

for i in range(100000):             # 爆破e
    res = pow(294,i,n1)
    if(res == pzm):
        e = i
        break

phi = (q-1)*(p-1)
d = invert(e,phi)
m = pow(c1,d,n1)
print(libnum.n2s(m))  # 数值转字符串
```

`BJD{p_is_common_divisor}`

5.[GKCTF2020]babycrypto

```
# n:0xb119849bc4523e49c6c038a509a74cda628d4ca0e4d0f28e677d57f3c3c7d0d876ef07d7581fe05a060546fedd7d061d3bc70d679b
6c5dd9bc66c5bdad8f2ef898b1e785496c4989daf716a1c89d5c174da494eee7061bcb6d52cafa337fc2a7bba42c918bbd3104dff62ecc9d
3704a455a6ce282de0d8129e26c840734ffd302bec5f0a66e0e6d00b5c50fa57c546cff9d7e6a978db77997082b4cb927df9847dfffef551
38cb946c62c9f09b968033745b5b6868338c64819a8e92a827265f9abd409359a9471d8c3a2631b80e5b462ba42336717700998ff38536c2
436e24ac19228cd2d7a909ead1a8494ff6c3a7151e888e115b68cc6a7a8c6cf8a6c005L
# e:65537
# enc:142256658448019987871466305146814351366793421621336673344205910652945193107827146036333588705419957795067 9
102659270179475911101747625120544429262334214483688332111552004535828182425152965223599160129610990036911146029 1
700335920557689834279048353958504146346595650921914608759002377115974212723120327964409485097244920272473761132 1
867818344322236453166998512803297125679253201505182904123020381409019461104117277536835719785445120126092711779 2
277559690205342515437625417792867692280849139537687763919269337822899746924269847694138899165820004160319118749 2
980310658005308695627046714357095789219014956881240423025003 61
# p>>128<<128:0xe4e4b390c1d201dae2c00a4669c0865cc5767bc444f5d310f3cfc75872d96feb89e556972c99ae20753e3314240a52df
5dccd076a47c6b5d11b531b92d901b2b512aeb0b263bbfd624fe3d52e5e238beeb581ebe012b2f176a4ffd1e0d2aa8c4d3a2656573b727d4
d3136513a931428b00000000000000000000000000000000L
```

Coppersmith 算法

P高位已知，低位未知

用Sagemath，攻击脚本Zui-Qing-Feng/RSA。
关于Sagemath

编译sagemath脚本，用sage-online解决

```
n = 0xb119849bc4523e49c6c038a509a74cda628d4ca0e4d0f28e677d57f3c3c7d0d876ef07d7581fe05a060546fedd7d061d3bc70d679b
6c5dd9bc66c5bdad8f2ef898b1e785496c4989daf716a1c89d5c174da494eee7061bcb6d52cafa337fc2a7bba42c918bbd3104dff62ecc9d
3704a455a6ce282de0d8129e26c840734ffd302bec5f0a66e0e6d00b5c50fa57c546cff9d7e6a978db77997082b4cb927df9847dfffef551
38cb946c62c9f09b968033745b5b6868338c64819a8e92a827265f9abd409359a9471d8c3a2631b80e5b462ba42336717700998ff38536c2
436e24ac19228cd2d7a909ead1a8494ff6c3a7151e888e115b68cc6a7a8c6cf8a6c005L
p_fake = 0xe4e4b390c1d201dae2c00a4669c0865cc5767bc444f5d310f3cfc75872d96feb89e556972c99ae20753e3314240a52df5dccd
076a47c6b5d11b531b92d901b2b512aeb0b263bbfd624fe3d52e5e238beeb581ebe012b2f176a4ffd1e0d2aa8c4d3a2656573b727d4d3136
513a931428b00000000000000000000000000000000L

pbits = 1024
kbits = 128
pbar = p_fake & (2^pbits-2^kbits)
print ("upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)  )

PR.<x> = PolynomialRing(Zmod(n))
f = x + pbar

x0 = f.small_roots(X=2^kbits, beta=0.4)[0]  # find root < 2^kbits with factor >= n^0.3
print (int(x0 + pbar))
```

```
pbits = 1024
kbits = 128
pbar = p_fake & (2^pbits-2^kbits)
print ("upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)  )

PR.<x> = PolynomialRing(Zmod(n))
f = x + pbar

x0 = f.small_roots(X=2^kbits, beta=0.4)[0]  # find root < 2^kbits with factor >= n^0.3
print (int(x0 + pbar))
```



```
upper 896 bits (of 1024 bits) is given
160734387026849747944319274262095716650717626398118440194223452208652532694713113062084219512359968722796763029072117463281356654614167941930993838521563406
```


得

到 p=160734387026849747944319274262095716650717626398118440194223452208652532694713113062084219512359968722796763029072117463281356654614167941930993838521563406258263299846297499190884495560744873319814150988520868951045961906000066805136724505347218275230562125457122462589771119429631727404626489634314291445667

由q=n//p得到q

```
>>> 0xb119849bc4523e49c6c038a509a74cda628d4ca0e4d0f28e677d57f3c3c7d0d876ef07d7581fe05a060546fedd7d061d3bc70d679b6c5dd9bc66c5bdad8f2ef898b1e785496c4989daf716a1c89d5c174da494eee7061bcb6d52cafa337fc2a7bba42c918bbd3104dff62ecc9d3704a455a6ce282de0d8129e26c840734ffd302bec5f0a66e0e6d00b5c50fa57c546cff9d7e6a978db77997082b4cb927df9847dfffef55138cb946c62c9f09b968033745b5b6868338c64819a8e92a827265f9abd409359a9471d8c3a2631b80e5b462ba42336717700998ff38536c2436e24ac19228cd2d7a909ead1a8494ff6c3a7151e888e115b68cc6a7a8c6cf8a6c005 // 160734387026849747944319274262095716650717626398118440194223452208652532694713113062084219512359968722796763029072117463281356654614167941930993838521563406258263299846297499190884495560744873319814150988520868951045961906000066805136724505347218275230562125457122462589771119429631727404626489634314291445667
139091353059018128421744751525080056530307965918298875691299992775484064426591581456998968315582349027071987206340653988925923465225471661893944397744293391269274124345189028818977002600599732469824164218366399726233373069742839737062004061244787413638290767590029376062508897417109117189614458570241407458359
>>>
```


接下来就是RSA常规操作了

```
from gmpy2 import*
import libnum
n=22356763374676421464625378500213339933332772809897207920729779273423674391734609826525432054721219700275907299
13247151892160932731719352256765963175774684203024187469291409835456431180619208073489564952078977888011546099997
13973202684541940857690744940359412410680906226760273075221532248260114209496048785258860756023841150910290983997
48434123617015174382209747228326250301273950316316969957774360584069874655921898737853921369255937089219232551862
82515777996509326779993612528103615281644689464568237409082282767318227236298791238683706176542426759149262625 3
49498709445342710799386836175120162674849965878446213480453
c=14225665844801998787146630514681435136679342162133667334420591065294519310782714603633358870541995779506791026
59270179475911101747625120544429262334214483688332111552004535828182425152965223599160129610990036911146029170033
35920557689834279048353958504146346595650921914608759002377115974212723120327964409485097244920272473761132186 78
18344322236453166998512803297125679253201505182904123020381409019461104117277536835719785445120126092711779227 75
59690205342515437625417792867692280849139537687763919269337822899746924269847694138899165820004160319118749298 03
10658005308695627046714357095789219014956881240423025003 61
p=16073438702684974794431927426209571665071762639811844019422345220865253269471311306208421951235996872279676302
90721174632813566546141679419309938385215634062582632998462974991908844955607448733198141509885208689510459619 06
00006680513672450534721827523056212545712246258977111942963172740462648963431429144 5667
e=65537

q=n//p
N=(q-1)*(p-1)
d=invert(e,N)
m=pow(c,d,n)


print(libnum.n2s(m))
```

```
flag{3d0914a1-1e97-4822-a745-c7e20c5179b9}
```

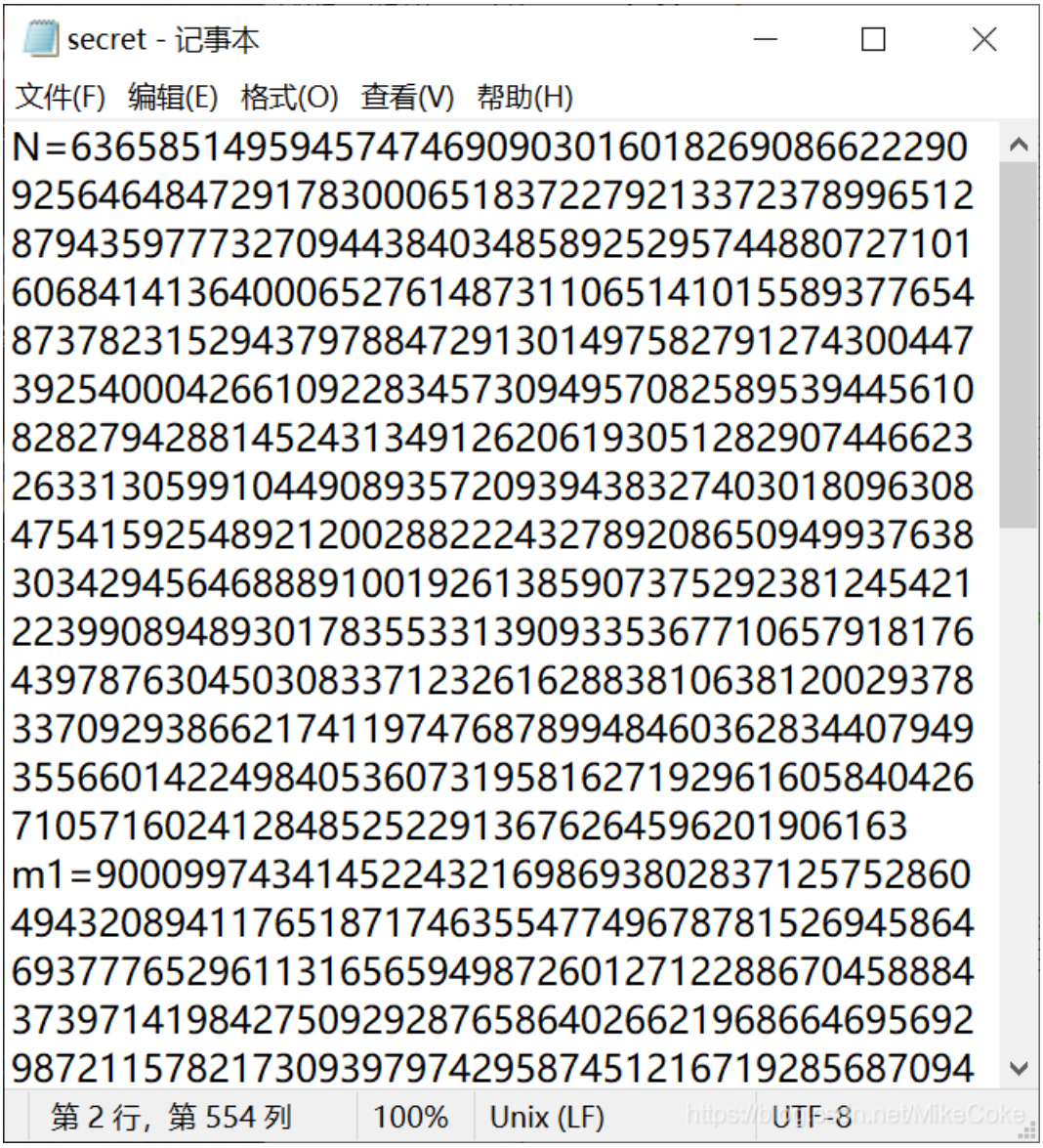
6.[GWCTF 2019]BabyRSA

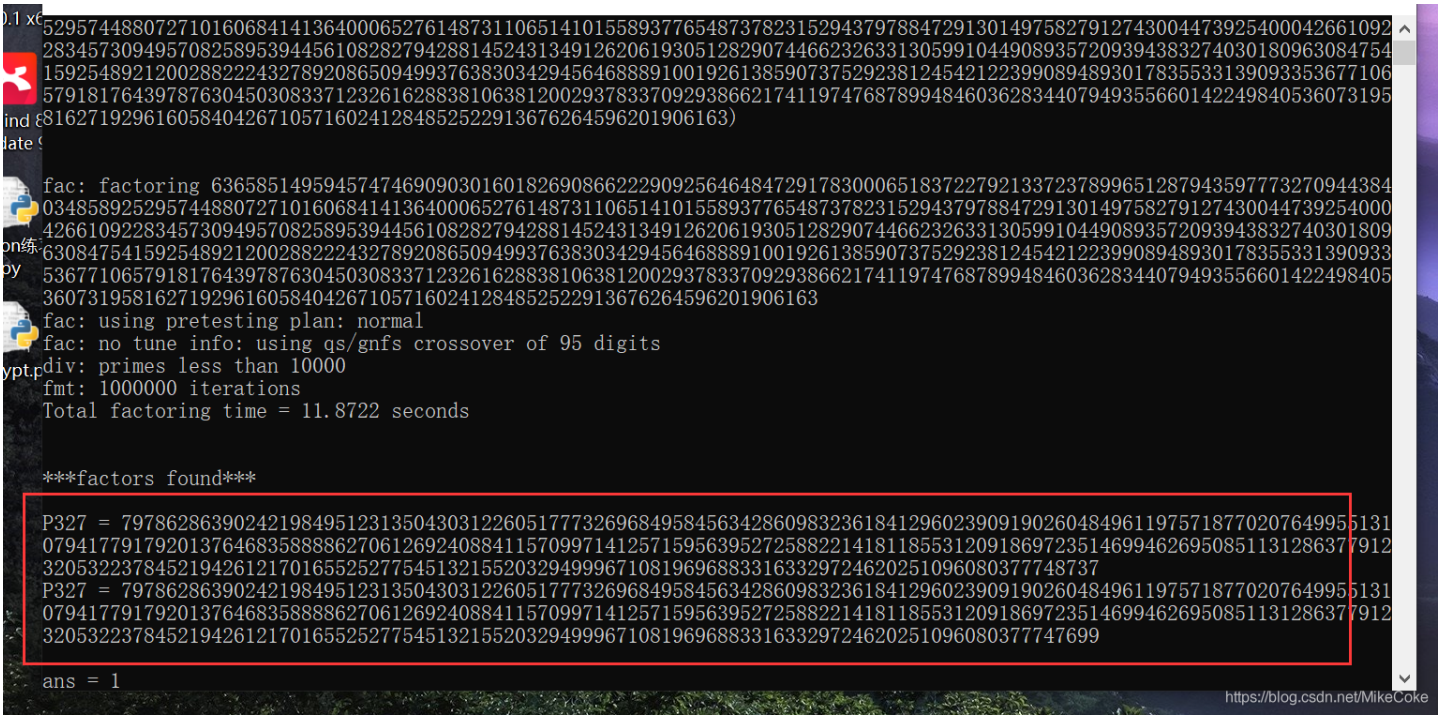
题目

```python
import hashlib
import sympy
from Crypto.Util.number import *

flag = 'GWHT{******}'
secret = '******'

0assert(len(flag) == 38)

half = len(flag) / 2

flag1 = flag[:half]
flag2 = flag[half:]

secret_num = getPrime(1024) * bytes_to_long(secret)

p = sympy.nextprime(secret_num)
q = sympy.nextprime(p)

N = p * q

e = 0x10001

F1 = bytes_to_long(flag1)
F2 = bytes_to_long(flag2)

c1 = F1 + F2
c2 = pow(F1, 3) + pow(F2, 3)
assert(c2 < N)

m1 = pow(c1, e, N)
m2 = pow(c2, e, N)

output = open('secret', 'w')
output.write('N=' + str(N) + '\n')
output.write('m1=' + str(m1) + '\n')
output.write('m2=' + str(m2) + '\n')
output.close()
```

先分析一下题目

由 secret文件给的N值，我们能够通过yafu分解出，p,q,的值来


secret - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710639783045030833712326162883810638120029378337092938662174119747687899484603628344079493556601422498405360731958162719296160584042671057160241284852522913676264596201906163
m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094
第 2 行，第 554 列 100% Unix (LF) UTF-8


fac: factoring 6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710639783045030833712326162883810638120029378337092938662174119747687899484603628344079493556601422498405360731958162719296160584042671057160241284852522913676264596201906163
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 11.8722 seconds
factors found
P327 = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377748737
P327 = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377747699
ans = 1

上代码

在使用符号之前，先要利用 SymPy 的 abc 子模块导入所需要的变量

Python科学计算利器——SymPy库

```
# 利用 SymPy 的 abc 子模块新建符号 x, y
from sympy.abc import x, y
```

```
================ RESTART: G:/CTF python脚本/[GWCTF 2019]BabyRSA.py ============
==
[(114155321203115613061978950846377251335007090 9, 159095629059803302986255661163
0426044507841845), (159095629059803302986255661163042604450784 1845, 114155321203
115613061978950846377251335007090 9)]
>>>
================ RESTART: G:/CTF python脚本/[GWCTF 2019]BabyRSA.py ============
==

================ RESTART: G:/CTF python脚本/[GWCTF 2019]BabyRSA.py ============
==
30ca8972959a1033b2}
GWHT{f709e0e2cfe7e5
>>>
================ RESTART: G:/CTF python脚本/[GWCTF 2019]BabyRSA.py ============
==
30ca8972959a1033b2}
GWHT{f709e0e2cfe7e5
>>>
```


`flag{f709e0e2cfe7e530ca8972959a1033b2}`

7.[BJDCTF2020]rsa_output

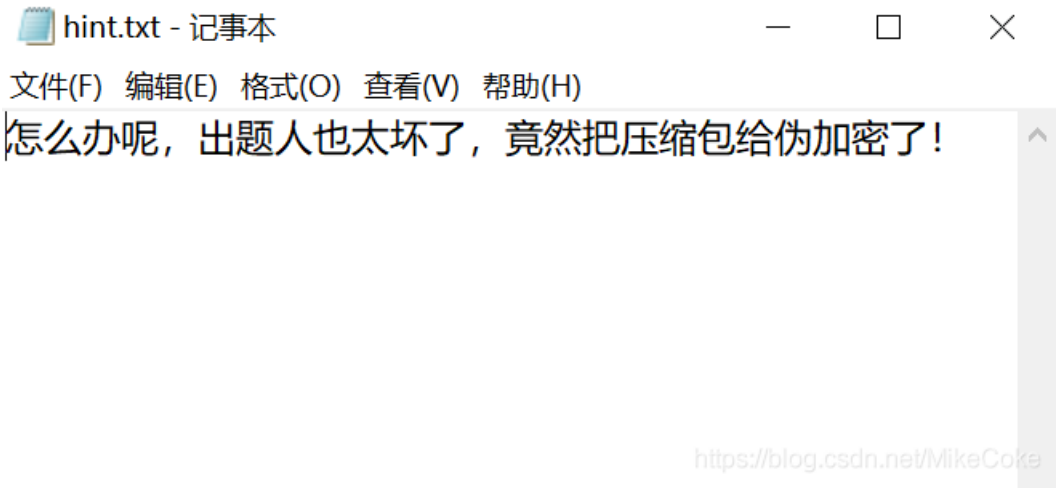
共模攻击

```
import libnum
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 = 21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111
  c1 = 20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599
  c2 = 11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227
  e1 = 2767
  e2 = 3659
  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 (libnum.n2s(m))

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

import libnum
m=102108971031231191041011101191011161041051101071051161051151121111151151059810810112 5

print(libnum.n2s(m))
'''
```

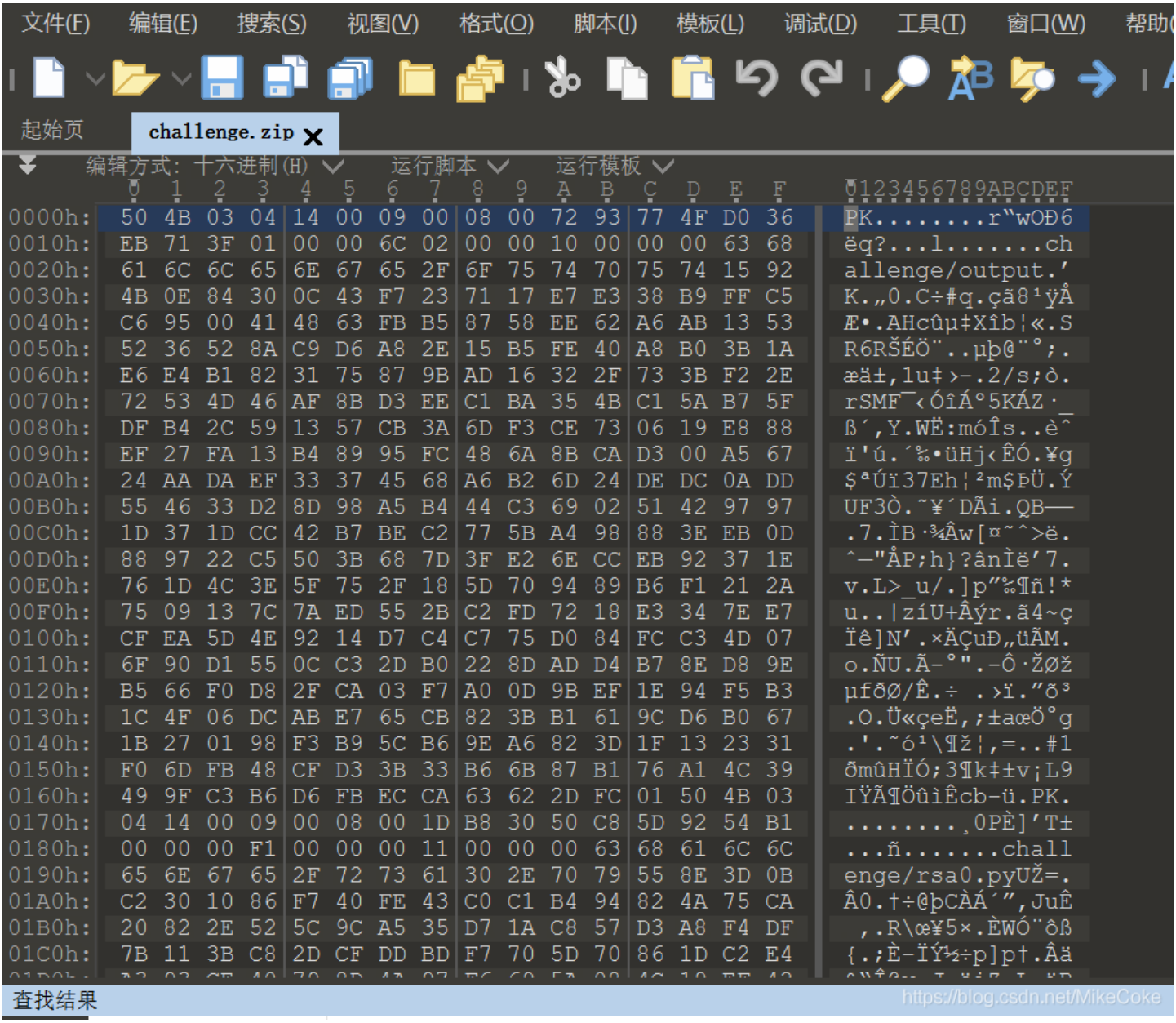
```
flag{r3a_C0mmoN_moD@_4ttack}
```

8.[ACTF新生赛2020]crypto-rsa0

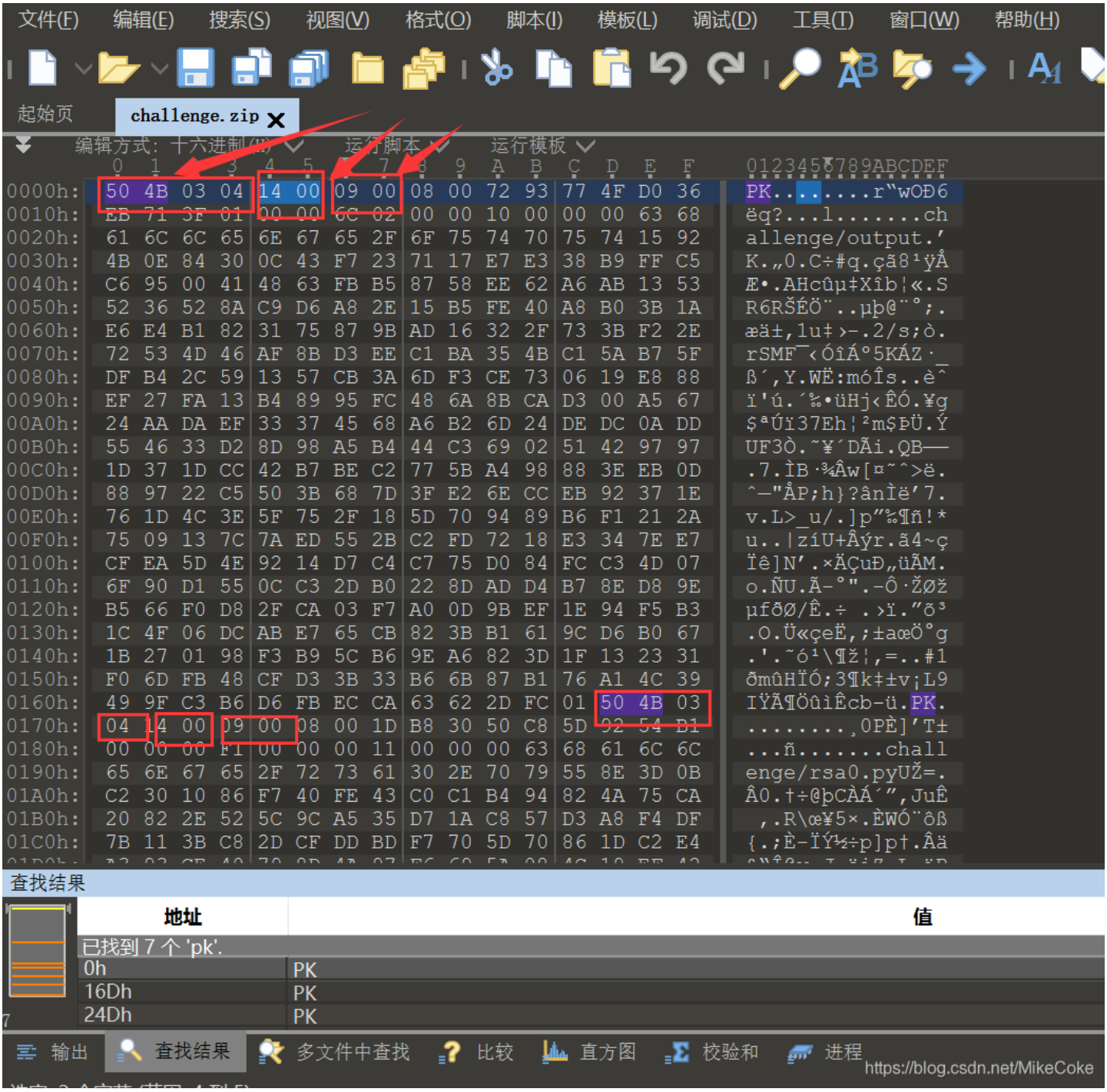
打开下载的文件，解压后得到 challenge.zip和hint.txt两个文件



通过hint文件可知，zip文件被伪加密了



将14 00 后面的全部改为偶数就可以了，不懂伪加密原理的自己去Google(我懒得找具体是哪里伪加密了，所以把全部的伪加密点都改了)


文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 脚本(I) 模板(L) 调试(D) 工具(T) 窗口(W) 帮助(H)
起始页
challenge.zip
编辑方式：十六进制(H)
运行脚本
运行模板
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 50 4B 03 04 14 00 09 00 08 00 72 93 77 4F D0 36 PK.....r"wOÐ6
0010h: EB 71 3F 01 00 00 6C 02 00 00 10 00 00 00 63 68 ëq?...l.....ch
0020h: 61 6C 6C 65 6E 67 65 2F 6F 75 74 70 75 74 15 92 allenge/output.'
0030h: 4B 0E 84 30 0C 43 F7 23 71 17 E7 E3 38 B9 FF C5 K.„0.C÷#q.çã8¹ÿÅ
0040h: C6 95 00 41 48 63 FB B5 87 58 EE 62 A6 AB 13 53 Æ•.AHcûµ‡Xîb¦«.S
0050h: 52 36 52 8A C9 D6 A8 2E 15 B5 FE 40 A8 B0 3B 1A R6RŠÉÖ¨..µþ@¨°;.
0060h: E6 E4 B1 82 31 75 87 9B AD 16 32 2F 73 3B F2 2E æä±‚1u‡›-.2/s;ò.
0070h: 72 53 4D 46 AF 8B D3 EE C1 BA 35 4B C1 5A B7 5F rSMF¯‹ÓîÁº5KÁZ·_
0080h: DF B4 2C 59 13 57 CB 3A 6D F3 CE 73 06 19 E8 88 ß´,Y.WË:móÎs..è^
0090h: EF 27 FA 13 B4 89 95 FC 48 6A 8B CA D3 00 A5 67 ï'ú.´‰•üHj‹ÊÓ.¥g
00A0h: 24 AA DA EF 33 37 45 68 A6 B2 6D 24 DE DC 0A DD $ªÚï37Eh¦²m$ÞÜ.Ý
00B0h: 55 46 33 D2 8D 98 A5 B4 44 C3 69 02 51 42 97 97 UF3Ò.˜¥´DÃi.QB——
00C0h: 1D 37 1D CC 42 B7 BE C2 77 5B A4 98 88 3E EB 0D .7.ÌB·¾Âw[¤˜ˆ>ë.
00D0h: 88 97 22 C5 50 3B 68 7D 3F E2 6E CC EB 92 37 1E ˆ—"ÅP;h}?ânÌë’7.
00E0h: 76 1D 4C 3E 5F 75 2F 18 5D 70 94 89 B6 F1 21 2A v.L>_u/.]p”‰¶ñ!*
00F0h: 75 09 13 7C 7A ED 55 2B C2 FD 72 18 E3 34 7E E7 u..|zíU+Âýr.ã4~ç
0100h: CF EA 5D 4E 92 14 D7 C4 C7 75 D0 84 FC C3 4D 07 Ïê]N’.×ÄÇuÐ„üÃM.
0110h: 6F 90 D1 55 0C C3 2D B0 22 8D AD D4 B7 8E D8 9E o.ÑU.Ã-°".-Ô·ŽØž
0120h: B5 66 F0 D8 2F CA 03 F7 A0 0D 9B EF 1E 94 F5 B3 µfðØ/Ê.÷ .›ï.”õ³
0130h: 1C 4F 06 DC AB E7 65 CB 82 3B B1 61 9C D6 B0 67 .O.Ü«çeË‚;±aœÖ°g
0140h: 1B 27 01 98 F3 B9 5C B6 9E A6 82 3D 1F 13 23 31 .'.˜ó¹\¶ž¦‚=..#1
0150h: F0 6D FB 48 CF D3 3B 33 B6 6B 87 B1 76 A1 4C 39 ðmûHÏÓ;3¶k‡±v¡L9
0160h: 49 9F C3 B6 D6 FB EC CA 63 62 2D FC 01 50 4B 03 IŸÃ¶ÖûìÊcb-ü.PK.
0170h: 04 14 00 09 00 08 00 1D B8 30 50 C8 5D 92 54 B1 ....¸0PÈ]’T±
0180h: 00 00 00 F1 00 00 00 11 00 00 00 63 68 61 6C 6C ...ñ.......chall
0190h: 65 6E 67 65 2F 72 73 61 30 2E 70 79 55 8E 3D 0B enge/rsa0.pyUŽ=.
01A0h: C2 30 10 86 F7 40 FE 43 C0 C1 B4 94 82 4A 75 CA Â0.†÷@þCÀÁ´”‚JuÊ
01B0h: 20 82 2E 52 5C 9C A5 35 D7 1A C8 57 D3 A8 F4 DF ‚.R\œ¥5×.ÈWÓ¨ôß
01C0h: 7B 11 3B C8 2D CF DD BD F7 70 5D 70 86 1D C2 E4 {.;È-ÏÝ½÷p]p†.Âä
查找结果
地址
值
已找到 7 个 'pk'.
0h PK
16Dh PK
24Dh PK
输出
查找结果
多文件中查找
比较
直方图
校验和
进程
https://blog.csdn.net/MikeCoke

简单的RSA题目

File Edit Format Run Options Window Help

```
from gmpy2 import*
from libnum import*

p=90185880664342063772402771624767392713862401730886765262953151639909683470229 2
q=75470056738777382578357297600377652133400366963507663242291436131799321451221 3

c=50996206925961019415256003394743594106061473865032792073035954925875056079762 6
e=65537
n=p*q

phi=(q-1)*(p-1)

d=invert(e,phi)

flag = pow(c,d,n)

print(n2s(flag))
```


`actf{n0w_y0u_see_RSA}`

9.[BJDCTF2020]easyrsa

题目，其中

Fraction(a,b) 相当于 a/b

Derivative(f(x),x) : 当x='x'时,求f(x)的导数值

arth(q)反双曲线正切函数

```python
from Crypto.Util.number import getPrime,bytes_to_long
from sympy import Derivative
from fractions import Fraction
from secret import flag

p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
m=bytes_to_long(flag)
c=pow(m,e,n)
print(c,z,n)
'''
output:
79225478668577614598074915026542162830127761777895115493506729581018102813484022840983101477965494306892538035109948774201355372685494106526544796208586913241103671820256487884070415999430913862275431821577462029470995723896760843927064060843076570001046656966544091550063132039572928857437917151987819742055786547921231915849576652932083904537483691823331528098823124533597061478081989229167627737217266815889771038774541190437448891645293831880774991949329096439186966468769073273647513809531825178831345918108008489717191848087136943429854581030066760134519122210802527359489936926748993998260848486221458154610353211574867762320966747162287218527507025792476601502007280526735983905939328431659588293337228973212727407643458751933330014247301034469480388516855754880120249593322621543776332928024211355652449845755956287290081160205694442396740377762330696188075761324632872961664303262896407293127208586692804597379937471184682515778105696516417850523252424580917923560757156717422882256169788864596855934360837533198809715714526435762673814164655635350099492411587574819831803629689860409700093827219590305673356588015054027536923963779397592332959871600335030825932143675257929100035556043154222969975995514115291470836249448215310745161336895413406690009324766200789179248896951942047235448901612351128459309145825547569298479821101249094161867207686537607047447968708758990950136380924747359052570549594098569970632854351825950729752563502284849263730127586382522703959893392329333760927637353052250274195821469023401443841395096410231843592101426591882573405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869343939106529204798285957516860774384001892777525916167743272419958572055332232056095979448155082465977781482598371994798871917514767508394730447974770329967681767625495394441

'''
```

Docs » SciPy函数求导数

17. SciPy求函数的导数

在SciPy里提供了很多的方法函数可以实现对某函数进行求导和求积分的操作。

SciPy的求导相对简单也容易理解。已知函数$f(x)$求其在x_0的导数即*[Math Processing Error]*。

- 方法一，使用scipy.misc模块下的derivative方法函数。

```
import numpy as np
from scipy.misc import derivative
def f(x):
    return x**5
for x in range(1, 4):
    print derivative(f, x, dx=1e-6)
```

程序的执行结果：

解题思路：

```
## 1.arctan（p）和 arth(q) ，求导后的值分别为1/(1+p** 2)  和 1/(1-q** 2)
## 2.通过构造方程组 z=p** 2+q**2, n = p*q,求出pq的值
```

解题py3代码

```
from sympy import*
from fractions import*
from sympy.abc import p,q
from gmpy2 import*
from libnum import*

e = 65537
c = 792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380
351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957 2
389676084392706406084307657000104665696654409155006313203957292885743791715198781974205578654792123191584957665 2
932083904537483691823331528098823124533597061478081989229167627737217266815889771038774541190437448891645293831 8
807749919493290964391869664687690732736475138095318251788313459181080084897171918480871369434298545810300667601 3
451912221080252735948993692674899399826084848622145815461035
z = 321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764
345875193333001424730103446948038851685575488012024959332262154377633292802421135565244984575595628729008116020 5
694442396740377762330696188075761324632872961664303262896407293127208586692804597379937471184682515778105696516 4
178505232524245809179235607571567174228822561697888645968559343608375331988097157145264357626738141646556353500 9
949241158757481983180362968986040970009382721959030567335658801505402753692396377939759233295987160033503082593 2
143675257929100035556043154222969975995514115291470836249448 2
n = 153107451613368954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012
490941618672076865376070474479687087589909501363809247473590525705495940985699706328543518259507297525635022848 4
926373012758638252270395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257 3
405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869 3
439391065292047982859575168607743840018927775259161677432724199585720553322320560959794481550824659777814825983 7
199479887191751476750839473044797477032996768176762549539444 1
'''
PQ=solve([pow(p,2)+pow(q,2)-z,p*q-n],[p,q])

print(PQ)
'''
p = 105909195259921349656664570904199242969110902804477734660927330311460997899731622163728968380757294196277263
615386525795293086103142131020215128282050307177125962302515483190468569376643751587606016315185736245896434947 6
915285676962719113981792883296092074353935793329315838293555587843050023608734589070291 41
q = 144564833334456076455156647979862690498796694770100520405218930055633597500009574663803955456004439398699669
751249623406199542605271188909145969364476344963078599240058180033000440459281558347909876143313940657252737586 8
030519353925965192269655198594745013919697557120971191639266727535887971808117110042033 01

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

m = pow(c,d,n)
print(n2s(m))
```

BJD{Advanced_mathematics_is_too_hard!!!}

10.[NCTF2019]babyRSA

题目

```
from Crypto.Util.number import *
from flag import flag

def nextPrime(n):
    n += 2 if n & 1 else 1
    while not isPrime(n):
        n += 2
    return n

p = getPrime(1024)
q = nextPrime(p)
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)

# d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
# c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218044
```

解题思路：

1.题目给了 e,d,c三个值，所以由公式 e*d=1 mod (q-1)*(p-1) 得到 (e*d-1)=k * (q-1)*(p-1) (设k为整数)

2.

分析题目可知p,q接近， p*q为2048位，

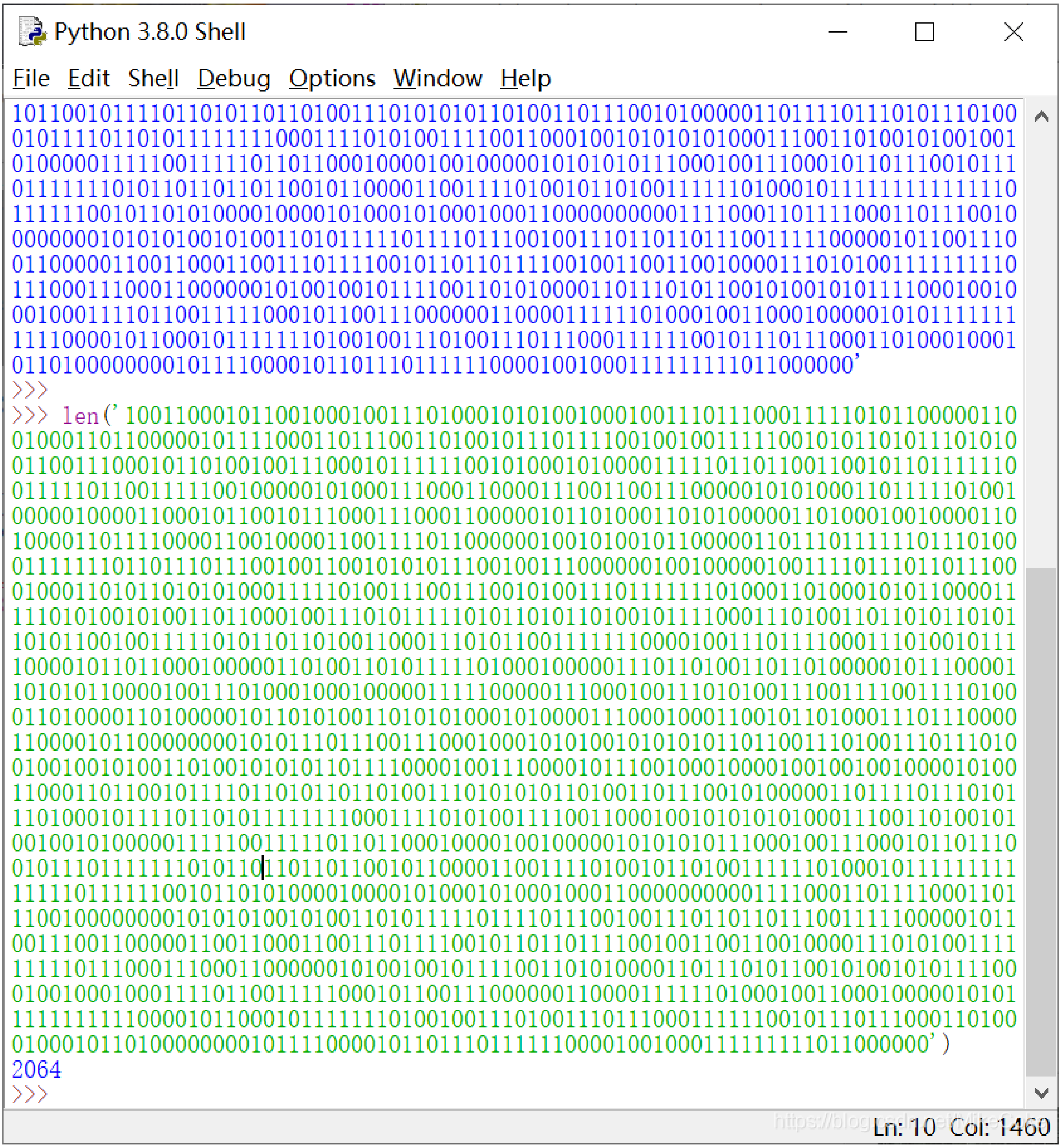
所以(p-1)*(q-1)<=2048计算可知 **e*d-1** 是 **2064** 位的，所以K的取值范围为（pow(2,15),pow(2,16)）

3.通过爆破K的值，可以得到(p-1)*(q-1) 的值，对phi 开平方，进而求得p,q。这里涉及的算法我通过举例来论证。

```
## 这个算法涉及两个点，开方后的取值的大小 Q，    Q<p<q，  或  p<Q<q
## 1.p<Q<q
假设 p=7,q=11,,则(p-1)*(q-1)=60,
iroot(60,2)[0]  = 7   （iroot()表示开根号，[0]表示取整数部分）
所以用sympy.next(7),就可以求得7后面的一个素数11

## 2.Q<p<q
假设 p=3,q=5，  则(p-1)*(q-1)=8,
iroot(8,2)[0]  = 2   （iroot()表示开根号，[0]表示取整数部分）
所以用sympy.next(2),就可以求得2后面的一个素数3

## 3.虽然Q取值范围不定，但是我们可以肯定的是，Q一定小于max(p,q),即p,q中最大的那个数。所以通过nextprime(Q),就可以求得p,q中任意的值。再用已知的 (e*d-1)//k  == (p-1)*(q-1),就可以求得（p-1）或是（q-1）的值了。最后再进行素性检测就可以了
```

File Edit Format Run Options Window Help

```
from gmpy2 import*
from sympy import*

e = 0x10001
d = 192757789460378997180354554381755091757239114661274621545069165641015199236O
c = 538272316807382811069616855829420668175799114902277782112756330141348322387 4


e_d_1=e*d-1

print(e_d_1)
```


Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
2064
Ln: 10 Col: 1460

爆破代码

```
from gmpy2 import*
from sympy import*
from libnum import*

e = 0x10001
d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553804763776311564686898441508783814605609907556528994493400453135218 04
```

```
for k in range(pow(2,15),pow(2,16)):
    if (e*d-1)%k == 0:
        phi = (e*d-1)//k

        P1 = iroot(phi,2)[0]    #  对(q-1)*(p-1)开平方

        p = nextprime(P1)     # 获取素数P

        q_1 = phi//(p-1)      # 得到(q-1)

        q = q_1+1
        if(isprime(q)):       # 对q进行素性检测
                    break
'''
print(p,q)

p=14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065343290987398659604248269967045171629674372752558643724846243232742336108081122507649 7
q=14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065343290987398659604248269967045171629674372752558643724846243232742336108081122507583 9
'''
n=p*q
m = pow(c,d,n)
print(n2s(m))
```

NCTF{70u2_nn47h_14_v3ry_gOO0000000d}