Buu Crypto

原创

iH0lmes 于 2021-07-31 15:09:09 发布 105 收藏

分类专栏： CTF Crypto 文章标签： python

本文链接：https://blog.csdn.net/iH0lmes/article/details/119275941

Buu RSA做题记录

RSA

题目

```
在一次RSA密钥对生成中，假设p=473398607161，q=4511491，e=17
求解出d作为flga提交
```

解

```
import gmpy2

p = 473398607161
q = 4511491
e = 17
phi = (p-1) * (q-1)
d = gmpy2.invert(e,phi)
print(d)

'''
125631357777427553
'''
```

rsarsa

题目

```
Math is cool! Use the RSA algorithm to decode the secret message, c, p, q, and e are parameters for the RSA algorithm.


p =  964842302901051567659055174001042653494573763923573980064398935203985250729849139956103500916342705037010757
073363335091169128029777716020062528166537848 3
q =  118748438379802970320924058486536568527609101545433809076500401907042833589092085782510630477324439922306479
038875100655479473135432993032619860534865694 07
e =  65537
c =  832082989951746041747735902982036393605400248712561268928896613457424033149298619391004926660564731664657648652621745700637684228086972858172674640158370589994176821413874225968933484073563355305388764184765117377625182029308721288567018036740680740676592363897316137581739273774783276275169010442386901903 4

Use RSA to find the secret message
```

解

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

p =  964842302901051567659055174001042653494573763923573980064398935203985250729849139956103500916342705037010757073363335091169128029777716020062528166537848 3
q =  118748438379802970320924058486536568527609101545433809076500401907042833589092085782510630477324439922306479038875100655479473135432993032619860534865694 07
e =  65537
c =  832082989951746041747735902982036393605400248712561268928896613457424033149298619391004926660564731664657648652621745700637684228086972858172674640158370589994176821413874225968933484073563355305388764184765117377625182029308721288567018036740680740676592363897316137581739273774783276275169010442386901903 4
n = p * q
phi = (p-1) * (q-1)

d = gmpy2.invert(e,phi)
m = gmpy2.powmod(c,d,n)
print(m)

'''
557744663355446577768879988
'''
```

RSA1

题目

```
p = 863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515811350222774520620532769093950403299469990205322 9
q = 126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470392840300825934907766305725849599542053368802284 69
dp = 650079570221683462110904235119326153065004384105625293093094966335862501688183284072806602615026469307610935487409984138045488171609777830726811691058292 9
dq = 783472263673553449019532580386470672380574033551303889137911760438881683674556098098256795673512201963002175438762767516968043599582527539160811120550041
c = 247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735304376570850796761573502053519452229893513160764865735995760419783398722659250627643185360890073102702785261596789374319038628924007479155251189839599706079341429747366757843259934459420313721073421038 52
```

dp,dq泄露

解

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

def decode(p,q,dp,dq,c,n):
    invertP = gmpy2.invert(p,q)
    m1 = gmpy2.powmod(c,dp,p)
    m2 = gmpy2.powmod(c,dq,q)
    m =( ( (m2-m1) * invertP ) * p + m1 ) % n
    print(long_to_bytes((m)))

p = 863763376725700856709965348654109117132049150943361544753916243791124417588566780639841179052408355344515811350222774520620532769093950403299469990205322 9
q = 126406749739964727691760479371708834209270508214800105815931371353724738805956137373376306297525773461470392840300825934907766305725849599542053368802284 69
dp = 650079570221683462110904235119326153065004384105625293093094966335862501688183284072806602615026469307610935487409984138045488171609777830726811691058292 9
dq = 783472263673553449019532580386470672380574033551303889137911760438881683674556098098256795673512201963002175438762767516968043599582527539160811120550041
c = 247223054038873820735673164676490806626315529059602293990791079956021544181760563358006388875276141640735304376570850796761573502053519452229893513160764865735995760419783398722659250627643185360890073102702785261596789374319038628924007479155251189839599706079341429747366757843259934459420313721073421038 52
n = p * q

decode(p,q,dp,dq,c,n)

'''
noxCTF{W31c0m3_70_Ch1n470wn}
'''
```

RSA3

题目

```
c1=223220352756632370416468937704519335093247019134843033380762106035426127589562628696408224864701211494244855713610074212936755163388221952803137949911360481409188424712198402635633888625049268273943641001343665116172072585548486669008478872134955566201987908150111322299612330553300932596437779889270316152185280595681121956388331289633015629862167468435391954755812792092570684280891476219901105495581653497767526739500957534782038707348392842506653636148277489237096952074030428745655550893337278232750656901077253749754176431142905221629119893209261779264525390147891080159287820356486111891204546495983256605136 1
e1=11187289

n=227080788158850114624620490643391858987124392772268310734578884031293785473502924202670165518190524307790047558466490440010241414852832864831307026160572746984736111495087988697063475019315831176327107007872280164801276773936499295304165986860273542164225659344590151619276136079028315428579778596125962823536793277733037270044072621972315863245991819835726224045903540845417880622621645101406058681224103880901744201477524085541297897609023008980462739090078528184740307706996476473630151021189567376739413542176926960449696953085064365731425655734875835070373569448480398643823392162666706735674888715089253111548 01
c2=187020100451870155565486916423949828356692621472302127313099386752264585552104259724294184492734105353879859310367118542656239050668056657518032691068807467690034789007910995902395139254497488140759040174715855728484735564905654500626647064491284158347879619472662597897859629222387011340797204142284140661930714953046123410529874556159300235368238014992697733571860874527475008406404193650115544211830375056534612867327409837027408226711480456194976671845861236572856040618756539095678223289140653377977334446403515187754876498199782623636172657979828431796308887294072384966509877204287082171152579890078673316983 97
e2=9647291
```

共模攻击

解

```
from Crypto.Util.number import *
import gmpy2 as g2

def egcd(a,b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, gcd = egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, gcd

c1=22322035275663237041646893770451933509324701913484303338076210603542612758956262869640822486470121149424485571361007421293675516338822195280313794991136048140918842471219840263536338886250492682739436410013436651161720725855484866690084788721349555662019879081501113222996123305533009325964377798892703161521852805956811219563883312896330156298621674684353919547558127920925706842808914762199011054955816534977675267395009575347820387073483928425066536361482774892370969520740304287456555508933372782327506569010772537497541764311429052216291198932092617792645253901478910801592878203564861118912045464959832566051361
e1=11187289
n=22708078815885011462462049064339185898712439277226831073457888403129378547350292420267016551819052430779004755846649044001024141485283286483130702616057274698473611149508798869706347501931583117632710700787228016480127677393649929530416598686027354216422565934459015161927613607902831542857977859612596282353679327773303727004407262197231586324599181983572622404590354084541788062262164510140605868122410388090174420147752408554129789760902300898046273909007852818474030770699647647363015102118956737673941354217692696044969695308506436573142565573487583507037356944848039864382392162666706735674888715089253111548014
c2=18702010045187015556548691642394982835669262147230212731309938675226458555210425972429418449273410535387985931036711854265623905066805665751803269106880746769003478900791099590239513925449748814075904017471585572848473556490565450062664706449128415834787961947266259789785962922238701134079720414228414066193071495304612341052987455615930023536823801499269773357186087452747500840640419365011554421183037505653461286732740983702740822671148045619497667184586123657285604061875653909567822328914065337797733444640351518775487649819978262363617265797982843179630888729407238496650987720428708217115257989007867331698397
e2=9647291

xy = egcd(e1,e2)
x = xy[0]
y = xy[1]
if x < 0:
    x = -x
    c1 = g2.invert(c1,n)
if y < 0:
    y = -y
    c2 = g2.invert(c2,n)

m = g2.powmod(c1,x,n) * g2.powmod(c2,y,n) % n
print(long_to_bytes(m))

'''
b'flag{49d91077a1abcb14f1a9d546c80be9ef}'

我们解出的y经常是负数，而在数论模运算中，要求一个数的负数次幂，与常规方法并不一样。比如此处要求c2的y次幂，就要先计算c2模n的逆元，然后求逆元的-y次幂。
'''
```

RSA2

题目

```
e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764361981420093067961210988553380133534844502375167047843707305554472428068473329805159916766030364518314616149748535863368149212966880240206579778990555048954764511878726660192942972413316776846530966590611 3
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036483444630319693920705664292714809329037444021050365 7

c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749428095741020195893573736038080184545382929399743341418883872575179626170262202858721156035336284719106030657851051138096516213347269871306359262102895916707278148256267368309059052121421807116028766518075 1
```

dp泄露

解

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

def decode(e,n,c,dp):
    for x in range(1, e):
        if (dp*e-1) % x == 0:
            if n % (((e*dp-1)//x)+1) == 0:
                p = (e*dp-1)//x+1
                q = n // p
                phi = (p-1) * (q-1)
                d = gmpy2.invert(e,phi)
                m = gmpy2.powmod(c,d,n)
                print(long_to_bytes(m))

e = 65537
n = 24825400785152624117772152669890180298583276617622160961225887737162058006043310153832803030521991869764361981420093067961210988553380133534844502375167047843707305554472428068473329805159916766030364518314616149748535863368149212966880240206579778990555048954764511878726660192942972413316776846530966590611 3
c = 14042367097625269680753367358620940057566428210068411978420352712452118899640382659743688376604187906749428095741020195893573736038080184545382929399743341418883872575179626170262202858721156035336284719106030657851051138096516213347269871306359262102895916707278148256267368309059052121421807116028766518075 1
dp = 90507449805234690464302513287951833069192517457305400462187725331868267505542197094355201669552856036483444630319693920705664292714809329037444021050365 7
decode(e,n,c,dp)

'''
b'flag{wow_leaking_dp_breaks_rsa?_98924743502}'
'''
```

RSA

http://ctf.ssleye.com/pub_asys.html

解

```
import rsa

e = 65537
n = 869344822960481191906660620034948005889056560172030256172166540583783221035 17
p = 285960468890451637935629440372639283459
q = 304008741604601924494328155975272418463
d = 81176168860169991027846870170527607562179635470395365333547868786951080991441

key = rsa.PrivateKey(n,e,d,q,p)
with open('flag.enc','rb')as flag:
    flag = flag.read()
    print(rsa.decrypt(flag,key))
```

RSAROLL

RSA roll！roll！roll！
Only number and a-z
（don't use editor
which MS provide）

{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
306220148

```
from Crypto.Util.number import *
import gmpy2 as g2

n = 920139713
e= 19
q = 18443
p = 49891
d = g2.invert(e, (p-1)*(q-1))
M = [704796792,752211152,274704164,18414022,368270835,483295235,263072905,459788476,483295235,459788476,663551792,475206804,459788476,428313374,475206804,459788476,425392137,704796792,458265677,341524652,483295235,534149509,425392137,428313374,425392137,341524652,458265677,263072905,483295235,828509797,341524652,425392137,475206804,428313374,483295235,475206804,459788476,306220148]
for c in M:
    m = g2.powmod(c,d,n)
    letter = long_to_bytes(m)
    print(letter.decode('utf-8'),end='') #消除b''

'''
flag{13212je2ue28fy71w8u87y31r78eu1e2}
'''
```

Dangerous RSA

```
#n:  0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793L
#e:0x3
#c:0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365
so,how to get the message?
```

低加密指数，直接开c的三次方根

```
from Crypto.Util.number import *
import gmpy2 as g2
n = 0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793
e = 0x3
c = 0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365
c = g2.iroot(c,3)#c是一个元组
print(long_to_bytes(list(c)[0]))
```

[HDCTF2019]basic rsa

题目

```
import gmpy2
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex

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

p = 262248800182277040650192055439906580479
q = 262854994239322828547925595487519915551

e = 65533
n = p*q


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

print c

# 27565231154623519221597938803435789010285480123476977081867877272451638645710
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2
p = 262248800182277040650192055439906580479
q = 262854994239322828547925595487519915551
c = 27565231154623519221597938803435789010285480123476977081867877272451638645710
e = 65533
n = p*q
d = g2.invert(e,(p-1)*(q-1))

m = g2.powmod(c,d,n)
print(long_to_bytes(m))

'''
b'flag{B4by_Rs4}'
'''
```

[GUET-CTF2019]BabyRSA

题目

```
p+q : 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea
(p+1)(q+1) : 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740
e : 0xe6b1bee47bd63f615c7d0a43c529d219
d : 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5
enc_flag : 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2

P = 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea #p+q
Q = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740 #(p+1)(q+1)
e = 0xe6b1bee47bd63f615c7d0a43c529d219
d = 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5
enc_flag = 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a
n = Q - P - 1

m = g2.powmod(enc_flag,d,n)
print(long_to_bytes(m))

'''
b'flag{cc7490e-78ab-11e9-b422-8ba97e5da1fd}'
'''
```

rsa2

题目

```
N = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383693740305816082770283909611956355972181848077519920922059268376958811713365106925235218265173085

import hashlib
flag = "flag{" + hashlib.md5(hex(d)).hexdigest() + "}"
```

e过大，考虑维纳攻击

```
from Crypto.Util.number import *
import gmpy2 as g2
from RSAwienerHacker import hack_RSA
import hashlib

N = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383697403058160827702839096119563559721818480775199209220592683769588117133651069252352182651730852
d = hack_RSA(e,N)
print(d)
Hash = hashlib.md5()
Hash.update(bytes(str(d), encoding='utf-8'))
flag = "flag{" + Hash.hexdigest() + "}"
print(flag)

'''
flag{2cf5122fc7b4e0e24c93d8c2f8323b10}由于python版本原因，所以导致hashlib.md5()的结果会不一样
'''
```

RSA5

题目

```
m = xxxxxxxx
e = 65537
========== n c ==========
n = 2047491889405177853330526234560188092808828447112182375404972535407247715587377884805507384334582069788664108684261248654125018396596600159134203156295356179333234164133430284799610841746636068813986650517968951658930563690213721018562465085490678003720441220630994919908000557692277577372243886376211775042932758579209344742398000240120061330294383421282090926971387668346581736915858582229467505697897061220288542643607195021453826292107740907616041743669983613880116262131484560879687020683470411670776316984738722330782890857094498441697301942752979002908976626494907803866952346524383767526385806285473908363420 7
c = 9744639082433308657289787692135954007820533985968977413162757225964150189129250863739385091922496927176638871002519503989696195606289557006214694773634034292797499261667889337274426195417287349087880548324119634588172116407865115606711995781642276852444202568807946265675560598210417400163534587402213304540234401004596111172015199041203447775585180276906930906901873854185413018369220475876142712127998200299393974534369567190001529679063746488033737551153642479689099652668120063308684103632039584772593574475799301335280465057506813612929559130656921330015633365091079594680082006749414336488584289629112613732 0
n = 2091881996064889134943826304695490221095914640786098074216593025378131875928569249251147526323424200250941907954564405175525131139263576341255349974450642156607472126882233732163726594222679034383985618210057553984535887749371833423758582126338818112654518972342926214963065128944655340219053113552083610421716026834968852516837521346257021361284589898969432426941020249687168864997837028466101739905690393184065675733085962618377339657405641301736760644654019997315563046623945363723293690406370655116065029503127338561947074059351026728595790580156636250226275775062916293737372129178952765953149943523526162030975 9
c = 1581963620197118553869488050512046933258215856714070824521803121848292387556864177196229718923770810072104155432038682511434979353089791861087415144087855679134383396897817458726543883093567600325204596156649305930352575274039425470836355002691145864435755333821133969266951545158052745938252574301327696822347115053614052423028835532509220641378760800693351542633860702225772638930501021571415907348128269681224178300248272689705308911282208685459668200507057183420662959113956077584781737983254788703048275698921427029884282557468334399677849962342196140864403989162117738206246183665814938783122909930082802031855

n = 2503325462590675727236960911921420203316212862517124643663957061526394915736327321312155682587873792326529057955187382437487095746716398954206348941663671365464248671721923122507411526968411942808635253547168335948624820364446146593550051790151323373915288294301017727654512830841293455583008777612835512593291484645947022110200766691221199231053889065439648711170538573050284358972728982969215217713475309864978141224706566063782628205516999182
```

409911091657685618887697562137660663425892778402578714226336715294710872075722244668641562747970366603187163565631428272705118919088900876305581168004031527707892806881649 1
c = 41853085294168740058312307810140924071984513859556773996685018339026234783956692794048839907251843327091524433725837010761987866352917393567708572867021071567300200043589556225110614106610589826220551997368208082038414467963052843946517144309186903894869205608346723161581464531837894121409390290293247560353580817544266451600332629243302486752161082709801570497054886202634851294809528147640028652800191851276624493183242793832777664162581422751439235321687984130110282715430852490290489974522125031117423023020654010514580665853953604684474606586729528516435471938227752183878536234536380254923891222045075559088621 2
n = 21206968097314131007183427944486801953583151151443627943113736996776787181111063957960698092696800555044199156765677935373149598221184792286812213296417749834607696302116136745662816658117055427803315230042700695125718401646810484873064775005221089174056824724922160855810527236751389605017579545235876864998419873065217294820244730785120525126565815560229001887622837549118168081685183371092395128598125004730268910276024806808565802081366898904032509920453785997056150497645234925528883879419642189109649009132381586673390027614766605038951015853086721168018787523459264932165046816881682774229243688581614306480751 3
c = 45210380110447584418911284684672330884938857508505889857085199111547780905971361261502890418934541266744681413934726623373503617122126948673116229704407077279411132638323571731417758552279737425710889745934763020841117706257642228383662775595608870429488598921385514726806545178149166092797483655806107122598566777405184770865315922331071754700682919036075057994329319896637074770179046114262137702383970057437303860800319556941584665584755997519402450391676291265767840244823484528683134174715429567782855677794359402671406799066865318624676272384010034591016371912972094224703881218025365697614144576182583435506131 4
n = 22822039733049388110936778173014765663663303811791283234361230649775805923902173438553927805407463106104699773994158375704033093471761387799852168337898526980521753614307899669015931387819927421875316304591521901592823814417756447695701045846773508629371397013053684553042185725059996791532391626429712416994990889693732805181947970071429309599614973772736556299404246424791660679253884940021728846906344198854779191951739719342908761330661910477119933428550774242910420952496929605686154799487839923424336353747442153571678064520763149793294360787821751703543288696726923909670396821551053048035619499706391118145067 5
c = 15406498580761780108625891878008526815145372096234083936681442225155097299264808624358826686906535594853622687379268969468433072388149786607395396424104318820879443743112358706546753935215756078345959375299650718555759698887852318017597503074317356745122514481807843745626429797861463012940172797612589031686718185390345389295851075279278516147076602270178540690147808314172798987497259330037810328523464851895621851859027823681655934104713689539848047163088666896473665500158179046196538210778897730209572708430067658411755959866033531700460551556380993982706171848970460224304996455600503982223448904878212849412357 6
n = 21574139855341432908474064784318462018475296809327285532337706940126942575349507668289214078026102682252713757703081553093108823214063791518482289846780197329821139507974763780260290309600884920811959842925540583967085670848765317877441480914852329276375776405689784571404635852204097622600656222714808541872252335877037561388406257181715278766652824786376262249274960467193961956690974853679795249158751078422296580367506219719738762159965958877806187461070689071290948181949561254144310776943334859775121650186245846031720507944987838489723127897223416802436021278671237227993686791944711422345000479751187704426369 7
c = 20366856150710305124583065375297661819795242238376485264951185336996083744604593418983336285185491197426018595031444652123288461491879021096028203694136683203441692987069563513026001861435722117985559909692670907347563594578265880806540396777223906955491026286843168637367593400342814725694366078337030937104035993569672959361347287894143027186846856772983058328919716702982222142848848117768499996617588305301483085428547267337070998767412540225911508196842253134355901263861121500650240296746702967594224401650220168780537141654489215019142122284308116284129004257364769474080721001708734051264841350424152506027932 8
n = 25360227412666612490102161131174584819240931803196448481224305250583841439581008528535930814167338381983764991296575637231916547647970573758269411168219302370541684789125112505021148506809643081950237623703181025696585998044695691322012183660424636496897073045557400768745943787342548267386564625462143150176113656264450210023925571945961405709276631990731602198104287528528055650050486159837612279600415259486306154947514005408907590083747758953115486124865486720633820559135063440942528031402951958557630833503775112010715604278114325528993771081233535247118481765852273252404963430792898948219539473312462979849137 9
c = 19892772524651452341027595619482734356243435671592398172680379981502759695784087900669089919987705675899945658648623800090272599154590123082189645021800958076861518397325439521139995652026377132368232502108620033400051346127757698623886142621793423225749240286511666556091787851683978017506983310073524398287279737680091787333547538239920607761080988243639547570818363788673249582783015475682109984715293163137324439862838574460108793714172603672477766831356411304446881998674779501188163600664488032943639694828698984739492200699684462748922883550002652913518229322945040819064133350314536378694523704793396169065179 10

n = 22726855244632356029159691753451822163331519237547639938779517751496498713174588935566576167329576494790219360727877166074136496129927296296996970048082870488804456564986667129388136556137013346228118981936899510687589585286517151323048293150257036847475424044378109168179412287889340596394755257704938006162677656581509375471102546261355748251869048003600520034656264521931808651038524134185732929570384705918563982065684145766427962502261522481994191989820110575981906998431553107525542001187655703534683231777988419268338249547641335718393312295800044734534761692799403469479540628978562990312574547359458674911910
c = 60401197951758564075410823600235322046147238586886367248227127175727597939602463418003081497398098712343130496297329347975697810530006861856663748339784032905250725987740017313502447445907727957010651295618981165764999841859206612711236653561327191936654742355968842391080306058827778688561223782226811405705191803212869769471540422726224113039810113025862256308598927317246405746581254782871151984062538473679798837680008126053954829526986896044777194789475954421859214806526378683356732332006621006210250615008957296053056658646931229525573618715231653002060703256603530955927780377673953603292313313228236100600060

n = 23297333791443053297363000786835336095252290818461950054542658327484507406594632785712767459958917943095522594228205423428207345128899745800927319147257669773812669542782839237744305180098276578841929496345963997512244219376701787616046235397139381894837435562662591060768476997333538748065294033141610502252325292801816812268934171361934399951548627267791401089703937389012586581080223313060159456238857080740699528666411303029934807011214953984169785844714159627792016926490955282697877141614638806397689306795328344778478692084754216753425842557818899467945102646776342655167655384224860504086083147841252232760941
c = 54181203012083787131158894655799642578718141145150460960909601597378590768292585169203615778539039259541984068437573036875578483023022002292959169024302057378436018067007382347566985757086124249284804408687391200758886816720622065291565664212766111078029174189936250296906271968138303263698742497776192396033006058768659675157190797971159105786535627878990193101399459049580248824178337363048947654334894762345753567552751472565773870228733489069001496349407471045138501541181069911370726433086202846631082830522457509452289953878034321288421522515492926989474076636438958534326500293520920183728344570542711028169340
n = 28873667904715682722987234293493200306976947898711255064125115933666968678742598858722431426218914462903521596341771131695619382266194233561677824357379805303885993804266436810606263022097900266975250431575654686915049693091467864820512767070713267708993899899011156106766178906700336111712803362113039613548672937053397875663144794018087017731949087794894903737682383916173267421403408140967713071026001874733487295007501068871044649170615709897145185679223231552669622016184274266477858128732131874820243146650894890274531437229979956162518695523467301209821091974587988226851265693171432678233521108957689731059149134
c = 99198804637868366849879579790915274774714449963923752440755278418655091601816665430163176349635124375103241987024163228413774894170295723884744500758014629968252446575302861074281863541728367165028176090705909297692619323242753532899393025364403106286983492448720640057006445202237276709507879242960042968830329789412008833626539933516385458602071790224724926712566304272284618526681180353170214286759548749470151977459169181977251211222363693827415339830234622559139246928062493874490166298658233164023660176578441669198466834978518423880582838562199005355674271036038699550661934255013852553220979015314021038838691
n = 22324685947539653722499932469409607533065419157347813961958075689047690465266404384199483683908594787312445528159635527833904475801890381455653807265501217328757871352731293000303438205315816792663917579066674842307743845261771032363928568844669895768092515658328756229245837025261744260614860746997931503548788509983868038349720225305730985576293675269073709022350700836510054067641753713212999954307022524495885583361707378513742162566339010134354907863733205921845038918224463903789841881400814074587261720283879760122070901466517118265422863420376921536734845502100251460872499122236686832189549698020737176683019
c = 14915270502032949898828292485603951848049772777471261431039572191646241875284410478373512635804406864747673804640055402646279101264831299306683440958145475921150610578434701314980750604203951110086190271990370199257012366601665630682456839757877628043595201647016916909164825910261385827055582468694961627597808784371379608230000439882273030038764105031213701633037116033594307645393375978668625084515281582851032518100587418796878752183841602825061727066133594776572154207348160493933395937554892185887966070602618979052334532686714116106310473404594879374795119334503694622137957389330190014718031576077917385384671
n = 27646746423759020111007828653264027999257847645666129907789026054594393648800236117046769112762641778865620892443423100189619327585811384883515424918752749559627553637785037359639801125213256163008431942593727931931898199727552768626775618479833029101249692573716030706695702510982283555740851047022672485743432464647772882314215176114732257497240284164016914018689044557218920300262234652840632406067273375269301008409860193180822366735877288205783314326102263756503786736122321348320031950012144905869556204017430593656052867939493633163499580242224763404338807022510136217187779084917996171602737036564991036724299

c = 21991524128957260536043771284854920393105808126700128222125856775506885721971193109361315961129190814674647133
6464887087893990660894961612838205086401018885457667488911898654270235561980111174603323721280911197488286585269
3568495792630434563163194764958886962193442198665168611876541805092478812512512789193462671299047392773862892403
9438457512433113565594351383100993402339745708218469973773438882376330680532643039584993577021381753338723548630
7008892410920611669932693018165569417445885810825749609388627231235840912644654685819620931663346297596334834498
661789016450371769203650109994771872404185770230172934013971

n = 2054548740581692873173898837447501268682793370978978439185570683513627027093340120301932913693765087838611718
7776530639342572123237188053978622697282521473917978282830432161153221216194169879669541998840691383025487220850
8720754360643084999249585179797279544029656121960814043416515173263640415192501250364248226343542687738954656989
2088343922299658122635859587399397660469983061393232072055413001167129794443351504718056548449519100388759989128
9037982010216357831078328159028953222056918189365840711588671093333013117454034313622855082795813122338562446223
041211192277089225078324682108033843023903550172891959673551
c = 1422743918819102946125047669279053965461919988848731942911441455797537630868890802814081715720557980405978380
7641305577385724758530138514972962209062230576107406142402603484375626077345190883094097636019771377866339531511
9651366505674123638891831596161884492637524753286632453110599883379960473592632888374363055888480445729377594244
6658687028051242433680706472989451584055240475687959069879704633333644546512044508758762174390662427962177963477
2378802959109714400516183718323267273824736540168545946444437586299214110424738159957388350785999348535171553569
373088251552712391288365295267665691357719616011613628772175

n = 2735972771158427723489715772405585279401921684522979893865581426946004638435356813859856775539255965346094944
4557879120040796798142218939251844762461270251672399546774067275348291003962551964648742053215424620256999345448
3988052785927770496682815583128717739799313430978068787011140560300415066904769542540065925552753425795296252311
9432135790466851212153951488070404696997489841209567508258531545826759101673492464629435766692429390841834550890
2112711075232047998775303603175363964055048589769318562104883659754974955561725694779754279606726358588862479198
815999276839234952142017210593887371950645418417355912567987
c = 3788529784248255027081674540877016372807848222776887920453488878247137930578296797437647922494510483767651150
4929333560932889659437415702689438619870242766107127174091399464095139630431144639331460884300042377471634228029
5925029660257064936301615158136400679589422659958470807258269699674051888760678546077585102981428035938576309107
8902301957226484620428513604630585131511167015763190591225884202772840456563643159507805711004113901417503751181
05082363820780353331114295109116161608513917547544347648195680548508238109011598212978497900056461021293540357353
50124476838786661542089045509656910348676742844957008857457

n = 2754593760375173724878522089173579646897332973807620914407992144996729257234942453901050228756403011683126126
8197384650511043068738911429169730640135947800885987171539267214611907687570587001933829208655100828045651391618
0896032884565703345005331786952384076847022512526715793710186516750543686062825246733699830346823305783087698864
5633581873382723729457047685367355268536168914426155289575826652239300411601784939734625911922106382166328093582
0440671825601452417487330105280889520007917979115568067161590058277418371493228631232457972494285014767469893647
892888681433965857496916110704944758070268626897045014782837
c = 1406911297060889573241703997754273266579660189376240150087878687168064579875478331569351126174005972517134240
4186571066972546332813667711135661176659424619936101038903439144294886379322591635766682645179888058617577572409
3074847081711444887084105434629720081799945940874739356380266126793897597568114905241271956287412628713044279084
8121499247118285930882877811900575092893576492796721234352650341051579371720136036043798132257679805627665714036
3332700714732224848346808963992302409037706094588964170239521193589470070839790404597252990818583717869140229811
712295005710540476356743378906642267045723633874011649259842

n = 2574616207569791156026318179121643306257417857242460033685627817611273305443146325390343312823270905414160710
0891177804285813783247735063753406524678030561284491481221681954564804141454666928657549670266775659862814924386
5841487854536473168649359427729191405635063056662078168976018627130928092344290965847532637078288997809792231181
8100929365556314652679238891346255730643366429696633146990642866512743882939970300286780026994785586926203671425
6550075520193125987011945192273531732276641728008406855871598678936585324782438668746810516660152018244253008092
470066555687277138937298747951929576231036251316270602513451
c = 1734428486027548947749152581992285532679227512871970940129254560812285982982746208839004461223496755168287995
4301458425842831995513832410355328065562098763660326163262033200347338773439095709944202252494552172589503915965
9315243265236632897775831526647222419208005378673310306239066740818522962323063362715428327284108036311702296427
1752494233239084246703514363150440114072708327073246423744391526386588058030877611121971896174637884292464414212
7243573824972533819479079381023103585862099063382129757560124074676150622288706094110075567706403442920696472627
797607697962873026112240527498308535903232663939028587036724

n = 23288486934117120315036919418588136227028485494137930196322371533620884932783396569389467056721797172792124383

```
n = 2328848693411712031503691941858813622702848549413793019632371533620884932783396569389467056721797172792124383
9129969128783853015760155446770590696037582684845937132790047363216362087277861336964760890214059732779383020349
2048032057258702254299859395701415082200412868578100481646967070186637584168077089106714774073660988834308118619
3301497340939017994857771257974935229944031054368903565146539986790842888554123777614340437633344294939706324922
3702355051571790555151203866821867908531733788784978667478707672984539512431549558672467752712004519300318999208
10207673250141258910490473498378989535875366407748689452949 9
c = 1073825441811407654807144884496404646814162174060321438498635418910523697707100142927156063642807597045989095
8274941762528116445171161040040833357876134689749846940052619392750394683504816081193432350669452446113285638982
5517625866563291090072140199449758164348277688827046304600012094522391628965761918763246623331538355339566002952
5515837702519842695094404064323543021101106358603246772432973578594737205175904213817105416585484247299058380089
9984893232549092766400510300083585513014171220423103452292891496141806956300396540682381668367564569427813092064
053993103537635994311143010708814851867239706492577203899024

n = 1959144138395852943559872911393634665700135257835790934765725723977754042481174981778306123323581791656068913
8344041497732749011519736303038986277394036718790971374656832741054547056417771501234494768509780369075443550907
8472982462757174205623751144060557336202587779052221697020364940450860173810842724961627702599558111744404901265
1474787666131775064948877499234800504438908110168601644621926406997137064631954642978290481006302032470413849560
8761532563310699753322444871060383693044481932265801505819646998535192083036872551683405766123968487907648980900
712118052346174533513978009131757167547595857552370586353973
c = 3834917098887202931981968704659119341624432294759361919553937551053499607440333234018189141970246302299385742
5482785898960332828949812003532706371272134831721825298904959034256491167559016311016658763017998656127177503600
8908517914275066460345419364205301638471451585586836872350892227176719028552113778568807562283292482924836277447
6456232826885801046969384519549385428259591566716890844604696258783639390854153039329480726205147199247183621535
1724508259790471324954396038408065012549971670511424271573817998907253237655588038080301094680486822520287202413
574786147046100891208103671924143520341774846885023640228 87

n = 1925424257158843017130819175787126107535852115862474570274405755605465233249596119679536963048478293029200323
8730267396462491733557715379956969694238267908985251699834707734400775311452868924330866502429576951934279223234
6766547492729327691073909763212086055162995325600540813018294406887969046354469860816911568422712680599707620042
5921903675317490994234320443279507637743210763020362175455280412440879235822007186236944320158415571189338887735
0138023238624566616551246804054720492816226651467017802504094070614892556444425915920269485861799532473383304622
064493223627552558344088839860178294589481899206318863310603
c = 6790553533991297205804561991225493105312398825187682250780197510784765226429663284220400480563039341938599783
3467240510762112656634686438264301090132450140358111782950819399586870874773128677202899645060978197620952444791
2935999886767181118197381966878846966804634586613743109946107600094742641157502049208755274344864375366235896845 1
9411519100170291423367424938566820315486507444202022408003879118465761273916755290898112991525546114191064022991
3297243700646325699038561892361778940077666907826302474438953588939837358228242434871818510987872712702567808910
944051219476310887299173983176523204977651017901326791718 89

n = 2680970025117127910297496294918441113645937226762053519842144983329844809258049748530195379661918533931606438
7798092220298630428207556482805739803420279056191194360049651767412572609187680508073074653291350998253938793269
2142304571171944348538887653034033858247862318594503512124494048707763202974197124865748047943256027603473064329
2728171616036883018794494012890797102783851007951946684617610656516473096398889240024006308939772041492139893639
9927948235195085202171264728816184532651138221862240969655185596628285814057082448321749567943946273776184657698
104465062749244327092588237927996419620170254423837876806659
c = 3862135566084340137698647271238794120419912715289905285485074512106926189866528704246322194246016775242650110
4314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980064667
1301748901528132153712913301179254879877441322285914544974519727307311002330350534857867516466612474769753577858
6600758305928914035518672460573978396883291725301771870422290286858620361407790657710619335281374230194073114735
8183240589908970925174700278803200209449537961468654467296907324930970348255638602462281473101576781004296981375
254861746497491571442559535194026607702167240985864542734 6
```

20组n和c，找最大公因子

解

```
from Crypto.Util.number import *
import gmpy2 as g2
e = 65537
```

n1 = 204749188940517785333052623456018809280882844711218237540497253540724771558737788480550738433458206978866410
868426124865412501839659660015913420315629535617933323416413343028479961084174663606881398665051796895165893056 3
690213721018562465085490678003720441220630994919908000557692277577372243886376211775042932758579209344742398000 2
401200613302943834212820909269713876683465817369158585822294675056978970612202885426436071950214538262921077409 0
761604174366998361388011626213148456087968702068347041167077631698473872233078289085709449844169730194275297900 2
908976626494907803866952346524383767526385806285473908363420 7
c1 = 974463908243330865728978769213595400782053398596897741316275722596415018912929508637393850919224969271766388
710025195039896961956062895570062146947736340342927974992616678893372744261954172873490878805483241196345881721 1
640786511560671199578164227685244420256880794626567556059821041740016353458740221330454023440100459611117201519 9
041203447775585180276906930906901873854185413018369220475876142712127998200299393974534369567190001529679063746 4
880337375511536424798909965266812006330868410363203958477259357447579930133528046505750681361292955913065692133 0
015633365091079594680082006749414336488584289629112613732 0

n2 = 209188199606488913494382630469549022109591464078609807421659302537813187592856924925114752632342420025094190
795456440517552513113926357634125534997445064215660747212688223373216372659422267903438398561821005755398453588 7
749371833423758582126338818112654518972342926214963065128944655340219053113552083610421716026834968852516837521 3
462570213612845898989694324269410202496871688649978370284661017399056903931840656757330859626183773396574056413 0
173676064465401999731556304662394536372329369040637065511606502950312733856194707405935102672859579058015663625 0
226275775062916293737372129178952765953149943523526162030975 9
c2 = 158196362019711855386948805051204693325821518567140708245218031218482923875568641771962297189237708100721041
554320386825114349793530897918610874151440878556791343833968978174587265438830935676003252045961566493059303525 7
527403942547083635500269114586443575533382113396926695154515805274593825257430132769682234711505361405242302883 5
532509220641378760800693351542633860702225772638930501021571415907348128269681224178300248272689705308911282208 6
854596682005070571834206629591139560775847817379832547887030482756989214270298842825574683343996778499623421961 4
086440398916211773820624618366581493878312290993008280203185 5

n3 = 250332546259067572723696091192142020331621286251712464366395706152639491573632732131215568258787379232652905
795518738243748709574671639895420634894166367136546424867172192312250741152696841194280863525354716833594862482 0
364446146593550051790151323373915288294301017727654512830841293455583008777612835512593291484645947022110200766 6
912211992310538890654396487111705385730502843589727289829692152177134753098649781412247065660637826282055169991 8
240991109165768561888769756213766066342589277840257871422633671529471087207572224466864156274797036660318716356 5
631428272705118919088900876305581168004031527707892806881649 1
c3 = 418530852941687400583123078101409240719845138595567739966850183390262347839566927940488399072518433270915244
337258370107619878663529173935677085728670210715673002000435895562251106141066105898262205519973682080820384144 6
796305284394651714430918690389486920560834672316158146453183789412140939029029324756035358081754426645160033262 9
243302486752161082709801570497054886202634851294809528147640028652800191851276624493183242793832777664162581422 7
514392353216879841301102827154308524902904899745221250311174230230206540105145806658539536046844746065867295285 1
643547193822775218387853623453638025492389122204507555908862

n4 = 212069680973141310071834279444868019535831511514436279431137369967767871811110639579606980926968005550441991
567656779353731495982211847922868122132946177498346076963021161367456628166581170554278033152300427006951257184 0
164681048487306477500522108917405682472492216085581052723675138960501757954523587686499841987306521729482024473 0
785120525126565815560229001887622837549118168081685183371092395128598125004730268910276024806808565802081366898 9
040325099204537859970561504976452349255288838794196421891096490091323815866733900276147666050389510158530867211 6
801878752345926493216504681688168277422924368858161430648075 1
c4 = 452103801104475844189112846846723308849388575085058898570851991115477809059713612615028904189345412667446814
139347266233735036171221269486731162297044070772794111326383235717314177585522797374257108897459347630208411177 0
625764222838366277559560887042948859892138551472680654517814916609279748365580610712259856677740518477086531592 2
331071754700682919036075057994329319896637074770179046114262137702383970057437303860800319556941584665584755997 5
194024503916762912657678402448234845286831341747154295677828556777943594026714067990668653186246762723840100345 9
101637191297209422470388121802536569761414457618258343550613

n5 = 228220397330493881109367781730147656636633038117912832343612306497758059239021734385539278054074631061046997
739941583757040330934717613877998521683378985269805217536143078996690159313878199274218753163045915219015928238 1
441775644769570104584677350862937139701305368455304218572505999679153239162642971241699499088969373280518194797 0
071429309599614973772736556299404246424791660679253884940021728846906344198854779191951739719342908761330661910 4
771199334285507742429104209524969296056861547994878399234243363537474421535716780645207631497932943607878217517 0
354328869672692390967039682155105304803561949970639111814506 7
c5 = 154064985807617801086258918780085268151453720962340839366814422251550972992648086243588266869065355948536226

873792689694684330723881497866073953964241043188208794437431123587065467539352157560783459593752996507185557596988878523180175975030743173567451225144818078437456264297978614630129401727976125890316867181853903453892958510752792785161470766022701785406901478083141727989874972593300378103285234648518956218518590278236816559341047136895398480471630886668964736655001581790461965382107788977302095727084300676584117559598660335317004605515563809939827061718489704602243049964556005039822344890487821284941235 7

n6 = 215741398553414329084740647843184620184752968093272855323377069401269425753495076682892140780261026822527137577030815530931088232140637915184822898467801973298211395079747637802602903096008849208119598429255405839670856708478653178774414809148523292763757764056897845714046358522040976226006562227148085418722523358770375613884062571817152787666528247863762622492749604671939619566909748536797952491587510784222965803675062197197387621599659588778061874610706890712909481819495612541443107769433348597751216501862458460317205079449878384897231278972234168024360212786712372279936867919447114223450004797511877044263691 0

c6 = 203668561507103051245830653752976618197952422383764852649511853369960837446045934189833362851854911974260185950314446521232884614918790210960282036941366832034416929870695635130260018614357221179855599096926709073475635945782658808065403967772239069554910262868431686373675934003428147256943660783370309371040359935696729593613472878941430271868468567729830583289197167029822221428488481177684999966175883053014830854285472673370709987674125402591150819684225313435590126386112150065024029674670296759422440165022016878053714165448921501914212228430811628412900425736476947408072100170873405126484135042415250602793 2

n7 = 253602274126666124901021611311745848192409318031964484812243052505838414395810085285359308141673383819837649912965756372319165476479705737582694111682193023705416847891251125050211485068096430819502376237031810256965859980446956913220121836604246364968970730455574007687459437873425482673865646254621431501761136562644502100239255719459614057092766319907316021981042875285280556500504861598376122796004152594863061549475140054089075900837477589531154861248654867206338205591350634409425280314029519585576308335037751120107156042781143255289937710812335352471184817658522732524049634307928989482195394733124629798491 37

c7 = 198927725246514523410275956194827343562434356715923981726803799815027596957840879006690899199877056758999456586486238000902725991545901230821896450218009580768615183973254395211399956520263771323682325021086200334000513461277576986238861426217934232257492402865116665560917878516839780175069833100735243982872797376800917873335475382399206077610809882436395475708183637886732495827830154756821099847152931631373244398628385744601087937141726036724777668313564113044468819986747795011881636006644880329436396948286989847394922006996844627489228835500026529135182293229450408190641333503145363786945237047933961690651 79

n8 = 227268552446323560291596917534518221633315192375476399387795177514964987131745889355665761673295764947902193607278771660741364961299272962969969700480828704888044565649866671293881365561370133462281189819368995106875895852865171513230482931502570368474754240443781091681794122878893405963947552577049380061626776565815093754711025462613557482518690480036005200346562645219318086510385241341857329295703847059185639820656841457664279625022615224819941919898201105759819069984315531075255420011876557035346832317779884192683382495476413357183933122958000447345347616927994034694979540628978562990312574545735945867491 91

c8 = 604011979517585640754108236002353220461472385868863672482271271757275979396024634180030814973980987123431304962973293479756978105300068618566637483397840329052507259877400173135024474459077279570106512956189811657649998418592066127112366535613271919366547423559688423910803060588277768856122378222681140570519180321286976947154042272622411303981011302586225630859892731724640574658125478287115198406253847367979883768000812605395482952698689604477719478947595442185921480652637868335673233200662100621025061500895729605305665864693122952557361871523165300206070325660353095592778037767395360329231331322823610060006

n9 = 232973337914430532973630007868353360952522908184619500545426583274845074065946327857127674599589179430955225942282054234282073451288997458009273191472576697738126695427828392377443051800982765788419294963459639975122442193767017876160462353971393818948374355626625910607684769973335387480652940331416105022523252928018168122689341713619343999515486272677914010897039373890125865810802233130601594562388570807406995286664113030299348070112149539841697858447141596277920169264909552826978771416146388063976893067953283447784786920847542167534258425578188994679451026467763426551676553842248605040860831478412522327609 41

c9 = 541812030120837871311588946557996425787181411451504609609096015973785907682925851692036157785390392595419840684375730368755784830230220022929591690243020573784360180670073823475669857570861242492848044086873912007588868167206220652915656642127661110780291741899362502969062719681383032636987424977619239603300605876865967515719079797115915057865356278789901931013994590495802488241783373630489476543348947623457535675527514725657738702287334890690014963494074710451385015411810699113707264330862028466310828305224575094522899538780343212884215225154929269894740766364389585343265002935209201837283445705427110281693 4

n10 = 288736679047156827229872342934930200306976947898711255064125115933666968678742598858722431426218914462903521596341771131695619382266194233561677824357379805303885993804266436810606263022097900266975250431575654686915049 6

930914678648205127670707132677089938998990111561067661789067003361117128033621130396135486729370533978756631447940180870177319490877948949037376823839161732674214034081409677130710260018747334872950075010688710446491706157098914518567922323155266962201618427426647785812873213187482024314665089489027453143722997995616251869552346730120982109197458798822685126569317143267823352110895768973105914910
c10 = 991988046378683668498795797909152747747144499639237524407552784186550916018166654301631763496351243751032419870241632284137748941702957238847445007580146299682524465753028610742818635417283671650281760907059092976926193232427535328993930253644031062869834924487206400570064452022372767095078792429600429688303297894120088336265399335163854586020717902247249267125663042722846185266811803531702142867595487494701519774591691819772512112223636938274153398302346225591392469280624938744901662986582331640236601765784416691984668349785184238805828385621990053556742710360386995506619342550138525532209790153140210388386910
n11 = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947873124455281596355278339044758018903814556538072655012173287578713527312930003034382053158167926639175790666748423077438452617710323639285688446698957680925156583287562292458370252617442606148607469979315035487885099838680383497202253057309855762936752690737090223507008365100540676417537132129999543070225244958855833617073785137421625663390101343549078637332059218450389182244639037898418814008140745872617202838797601220709014665171182654228634203769215367348455021002514608724991222366868321895496980207371766830191
c11 = 14915270502032949898828292485603951848049772777471261431039572191646241875284410478373512635804406864747673804640055402646279101264831299306683440958145475921150610578434701314980750604203951110086190271990370199257012366601665630682456839757877628043595201647016916909164825910261385827055582468694961627597808784371379608230000439882273030038764105031213701633037116033594307645393375978668625084515281582851032518100587418796878752183841602825061727066133594776572154207348160493933395937554892185887966070602618979052334532686714116106310473404594879374795119334503694622137957389330190014718031576077917385384671
n12 = 27646746423759020111007828653264027999257847645666129907789026054594393648800236117046769112762641778865620892443423100189619327585811384883515424918752749559627553637785037359639801125213256163008431942593727931931898199727552768626775618479833029101249692573716030706695702510982283555740851047022672485743432464647772882314215176114732257497240284164016914018689044557218920300262234652840632406067273375269301008409860193180822366735877288205783314326102263756503786736122321348320031950012144905869556204017430593656052867939493633163499580242224763404338807022510136217187779084917996171602737036564991036724299
c12 = 219915241289572605360437712848549203931058081267001282221258567755068857219711931093613159611291908146746471364648870878939906608949616128382050864010188854576674889118986542702355619801111746033237212809111974882865852693568495792630434563163194764958886962193442198665168611876541805092478812512512789193462671299047392773862892403943845751243311356559435138310099340233974570821846997377343888237633068053264303958499357702138175333872354863070088924109206116699326930181655694174458858108257496093886272312358409126446546858196209316633462975963348344986617890164503717692036501099947718724041857702301729340139711
n13 = 2054548740581692873173898837447501268682793370978978439185570683513627027093340120301932913693765087838611718777653063934257212323718805397862269728252147391797828283043216115322121619416987966954199884069138302548722085087207543606430849992495851797972795440296561219608140434165151732636404151925012503642482263435426877389546569892088343922299658122635859587399376604699830613932320720554130011671297944433515047180565484495191003887599891289037982010216357831078328159028953222056918189365840711588671093333013117454034313622855082795813122338562446223041211192277089225078324682108033843023903550172891959673551
c13 = 142274391881910294612504766927905396546191998884873194291144145579753763086889080281408171572055798040597838076413055773857247585301385149729622090622305761074061424026034843756260773451908830940976360197713778663395315119651366505674123638891831596161884492637524753286632453110599883379960473592632888374363055888480445729377594244665868702805124243368070647298945158405524047568795906987970463333364454651204450875876217439066242796217796347723788029591097144005161837183232672738247365401685459464444375862992141104247381599573883507859993485351715535693730882515527123912883652952676656913577196160116136287721751
n14 = 27359727711584277234897157724055852794019216845229798938655814269460046384353568138598567755392559653460949444557879120040796798142218939251844762461270251672399546774067275348291003962551964648742053215424620256999345448398805278592777049668281558312871773979931343097806878701114056030041506690476954254006592555275342579529625231194321357904668512121539514880704046969974898412095675082585315458267591016734924646294357666924293908418345508902112711075232047998775303603175363964055048589769318562104883659754974955561725694779754279606726358588862479198815999276839234952142017210593887371950645418417355912567987
c14 = 37885297842482550270816745408770163728078482227768879204534888782471379305782967974376479224945104837676511504929333560932889659437415702689438619870242766107127174091399464095139630431144639331460884300042377471634228029592502966025706493630161515813640067958942265995847080725826969967405188876067854607758510298142803593857630911

07890230195722648462042851360463058513151116701576319059122588420277284045656364315950780571100411390141750375118105082363820780353311142951091161616085139175475443476481956805485082381090115982129784979000564610212935403573535012447683878666154208904550965691034867674284495700885745 7

n15 = 27545937603751737248785220891735796468973329738076209144079921449967292572349424539010502287564030116831261268197384650511043068738911429169730640135947800885987171539267214611907687570587001933829208655100828045651391618089603288456570334500533178695238407684702251252671579371018651675054368606282524673369983034682330578307869886456335818733827237294570476853673552685361689144261552895758266522393004116017849397346259119221063821663280935820440671825601452417487330105280889520007917979115568067161590058277418371493228631232457972494285014767469893647892888681433965857496916110704944758070268626897045014782837

c15 = 14069112970608895732417039977542732665796601893762401500878786871680645798754783315693511261740059725171342404186571066972546332813667711135661176659424619936101038903439144294886379322591635766682645179888058617577572409307484708171144488708410543462972008179994594087473935638026612679389759756811490524127195628741262871304427908481214992471182859308828778119005750928935764927967212343526503410515793717201360360437981322576798056276657140363332700714732224848346808963992302409037706094588964170239521193589470070839790404597252990818583717869140229811712295005710540476356743378906642267045723633874011649259842

n16 = 257461620756979115602631817912164330625741785724246003368562781761127330544314632539034331282327090541416071008911778042858137832477350637534065246780305612844914812216819545648041414546669286575496702667756598628149243865841487854536473168649359427729191405635063056662078168976018627130928092344290658475326370782889978097922311818100929365556314652679238891346255730643366429696633146990642866512743882939970300286780026994785586926203671425655007552019312598701194519227353173227664172800840685587159867893658532478243866874681051666015201824425300809247006655568727713893729874795192957623103625131627060251345 1

c16 = 1734428486027548947749152581992285532679227512871970940129254560812285982982746208839004461223496755168287995430145842584283199551383241035532806556209876366032616326203320034733877343909570994420225249455217258950391596593152432652366328977758315266472224192080053786733103062390667408185229623230633627154283272841080363117022964271752494233239084246703514363150440114072708327073246423744391526386588058030877611121971896174637884292464414212724357382497253381947907938102310358586209906338212975756012407467615062228870609411007556770640344292069647262779760769796287302611224052749830853590323266393902858703672 4

n17 = 2328848693411712031503691941858813622702848549413793019632371533620884932783396569389467056721797172792124383912996912878385301576015544677059069603758268484593713279004736321636208727786133696476089021405973277938302034920480320572587022542998593957014150822004128685781004816469670701866375841680770891067147740736609888343081186193301497340939017994857771257974935229944031054368903565146539986790842888554123777614340437633344294939706324922370235505157179055515120386682186790853173378878497866747870767298453951243154955867246775271200451930031899920810207673250141258910490473498378989535875366407748689452949 9

c17 = 1073825441811407654807144884496404646814162174060321438498635418910523697707100142927156063642807597045989095827494176252811644517116104004083335787613468974984694005261939275039468350481608119343235066945244611328563898255176258665632910900721401994497581643482776888270463046000120945223916289657619187632466233315383553395660029525515837702519842695094404064323543021101106358603246772432973578594737205175904213817105416585484247299058380089998489323254909276640051030008358551301417122042310345229289149614180695630039654068238166836756456942781309206405399310353763599431114301070881485186723970649257720389902 4

n18 = 1959144138395852943559872911393634665700135257835790934765725723977754042481174981778306123323581791656068913834404149773274901151973630303898627739403671879097137465683274105454705641777150123449476850978036907544355090784729824627571742056237511440605573362025877790522216970203649404508601738108427249616277025995581117444049012651474787666131775064948877499234800504438908110168601644621926406997137064631954642978290481006302032470413849560876153256331069975332244487106038369304448193226580150581964699853519208303687255168340576612396848790764898090071211805234617453351397800913175716754759585755237058635397 3

c18 = 3834917098887202931981968704659119341624432294759361919553937551053499607440333234018189141970246302299385742548278589896033282894981200353270637127213483172182529890495903425649116755901631101665876301799865612717750360089085179142750664603454193642053016384714515855868368723508922271767190285521137785688075622832924829248362774476456232826885801046969384519549385428259591566716890844604696258783639390854153039329480726205147199247183621535172450825979047132495439603840806501254997167051142427157381799890725323765558803808030109468048682252028720241357478614704610089120810367192414352034177484688502364022887

n19 = 1925424257158843017130819175787126107535852115862474570274405755605465233249596119679536963048478293029200323873026739646249173355771537995696969423826790898525169983470773440077531145286892433086650242957695193427922323467665474927293276910739097632120860551629953256005408130182944068879690463544698608169115684227126805997076200425921903675317490994234320443279507637743210763020362175455280412440879235822007186236944320158415571189338887 7

```
350138023238624566616551246804054720492816226651467017802504094070614892556444425915920269485861799532473383304622064493223627552558344088839860178294589481899206318863310603
c19 = 67905535339912972058045619912254931053123988251876822507801975107847652264296632842204004805630393419385997833467240510762112656634686438264301090132450140358111782950819399586870874773128677202899645060978197620952444791293599988676718118197381966878846966804634586613743109946107600094742641157502049208755274344864375366235896845194115191001702914233674249385668203154865074442020224080038791184657612739167552908981129915255461141910640229913297243700646325699038561892361778940077666907826302474438953588939837358228242434871818510987872712702567808910944051219476310887299173983176523204977651017901326791718890
n20 = 26809700251171279102974962949184411136459372267620535198421449833298448092580497485301953796619185339316064387798092220298630428207556482805739803420279056191194360049651767412572609187680508073074653291350998253938793269214230457117194434853888765303403385824786231859450351212449404870776320297419712486574804794325602760347306432927281716160368830187944940128907971027838510079519466846176106565164730963988892400240063089397720414921398936399927948235195085202171264728816184532651138221862240969655185596628285814057082448321749567943946273776184657698104465062749244327092588237927996419620170254423837876806659
c20 = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752426501104314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980064667130174890152813215371291330117925487987744132228591454497451972730731100233035053485786751646661247476975357785866007583059289140355186724605739783968832917253017718704222902868586203614077906577106193352813742301940731147358183240589908970925174700278803200209449537961468654467296907324930970348255638602462281473101576781004296981375254861746497491571442559535194026607702167240985864542734 6

n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20]
c = [c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20]
for i in range(20):
    for j in range(20):
        if i != j:
            if g2.gcd(n[i],n[j]) != 1:
                N = n[i]
                p = g2.gcd(n[i],n[j])
                q = N // p
                d = g2.invert(e, (p-1)*(q-1))
                m = g2.powmod(c[i], d, N)
                print(long_to_bytes(m))

'''
b'flag{abdcbe5fd94e23b3de429223ab9c2fdf}'
'''
```

[NCTF2019]childRSA

题目

```
from random import choice
from Crypto.Util.number import isPrime, sieve_base as primes
from flag import flag

def getPrime(bits):
    while True:
        n = 2
        while n.bit_length() < bits: #bit_length()，二进制的位数
            n *= choice(primes) #n = n * (随机从sieve_base列表中选取的一个数)
        if isPrime(n + 1): #如果n+1是个素数，返回该值
            return n + 1

e = 0x10001
m = int.from_bytes(flag.encode(), 'big') #big代表正常顺序，十六转十
p, q = [getPrime(2048) for _ in range(2)]
n = p * q
c = pow(m, e, n)

# n = 32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579
667346279456710633774501407292473006312537723894221717638059058796679686953564471994009285384798450493756900459225040360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044633282064056800913282016363415202171926089293431012379261585078566301060173689328363696699811123592090204578098276704877408688525618732848817623879899628629300385790344366046641825507767709276622692835393219811283244303899850483748651722336996164724553364097066493953127153066970594638491950199605713033004684970381605908909693802373826516622872100822213645899846325022476318425889580091613323747640467299866189070780620292627043349618839126919699862580579994887507733838561768581933029077488033326056066378869170169389819542928899483936705521710423905128732013121538495096959944889076705471928490092476616709838980562233255542325528398956185421193665359897664110835645928646616337700617883946369110702443135980068553511927115723157704586595844927607636003501038871748639417378062348085980873502535098755568810971926925447913858894180171498580131088992227637341857123607600275137768132347158657063692388249513
# c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108
```

sieve_base :包含了前10000个素数的列表

解

n可以直接用factordb分解

```
from Crypto.Util.number import *
import gmpy2 as g2

n = 32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579667346279456710633774501407292473006312537723894221717638059058796679686953564471994009285384798450493756900459225040360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044633282064056800913282016363415202171926089293431012379261585078566301060173689328363696699811123592090204578098276704877408688525618732848817623879899628629300385790344366046641825507767709276622692835932198112832443038998504837486517223369961647245533640970664939531271530669705946384919501996057130330046849703816059089096938023738265166228721008222136458998463250224763184258895800916133237476404672998661890707806202926270433496188391269196998625805799948875077338385617685819330290774880333260560663788691701693898195429288994839367055217104239051287320131215384950969599448890767054719284900924766167098389805622332555423255283989561854211936653598976641108356459286466163377006178839463691107024431359800685535119271157231577045865958449276076360035010388717486394173780623480859808735025350987555688109719269254479138588941801714985801310889922276373418571236076002751377681323471586570636923882495131307597161367011014142898823526027674354555037785878481711602257307508985022577801782788769786800015984410443717799994642236194840684557538917849420967360121509675348296203886340264385224150964642958965438801864306187503790100281099130863977710204660546799128755418521327290719635075221585824217487386227004673527292281536221958961760681032293340099395863194031788435142296085219594866635192464353365034089592414809332183882423461536123972873871477755949082223830049594561329457349537703926325152949582123419049073013144325689632055433283354999265193117288252918515308767016885678802217366700376654365502867
c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108
e = 0x10001
d = g2.invert(e,(p-1)*(q-1))
m = g2.powmod(c,d,n)
print(long_to_bytes(m))

'''
b'NCTF{Th3r3_ar3_1ns3cure_RSA_m0duli_7hat_at_f1rst_gl4nce_appe4r_t0_be_s3cur3}'
'''
```

[HDCTF2019]bbbbbbrsa

题目

```
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:
  break;
 else:
  e -= 1;

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

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

# 23737406995293649917635893242000934662067855618361018403816222372255122346 32

p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
c = ==gMzYDNzljMxUTNyIzNzljMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwljN2QzM5ADMwlDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2
import base64

p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
q = n // p
c = '==gMzYDNzljMxUTNylzNzljMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwljN2QzM5ADMwlDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM'
c = base64.b64decode(c[::-1])
c = int(c.decode('utf-8'))

for e in range(50000,70000):
    if g2.gcd(e,(p-1)*(q-1)) == 1:
        d = g2.invert(e,(p-1)*(q-1))
        m = g2.powmod(c,d,n)
        m = long_to_bytes(m)
        if b'flag' in m:
            print(m)

'''
b'flag{rs4_1s_s1mpl3!#}'
'''
```

[BJDCTF2020]RSA

题目

```
from Crypto.Util.number import getPrime,bytes_to_long

flag=open("flag","rb").read()

p=getPrime(1024)
q=getPrime(1024)
assert(e<100000)
n=p*q
m=bytes_to_long(flag)
c=pow(m,e,n)
print c,n
print pow(294,e,n)

p=getPrime(1024)
n=p*q
m=bytes_to_long("BJD"*32)
c=pow(m,e,n)
print c,n

'''
output:
126416356178037461503322326463545962927078614802002075371991411836244383037571205700967412480202366669657557980096565477386163990253001230437662555185961493489304445998206752300464233730530516319325572308490834268594901837323037517440048741830625948568703186142899916759800635483164994869089232096275638715548756127020791005670186989929358182061090875681660973923141057175554829261410305056395717088762131671121879625844840653215457275941351753692339259225077949996073235369768241831629233850056699304034488534651414058468359198429084697875473417523654718924952043076445861613932287760420155341479138883383162441691201350877410446020974330671403454670413724762734498113346180195347973601702140172581880846289837599476737562774949483967194454382240305997807381312244140761253065816894298782025678658300694700171174923019354237057090570553016792170283562712240147525103900077501738163390022247472739682370869506313624611565262225976963459130942176126954826098442614882464128501073098321537750925501129873782762161115803297642001166254785451561059795562889807356968415822567833347454392032653289344684980811283747668439003097647205390506985552229785068802696070118654342813984378390762431727479692624882954341346475412720884307033106303738163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699878976943319063836649085085377577273214792371548775204594097887078898598463892440141577974544939268247818937936607013100808169758675042264568547764031628431414727922168580998494695800403043312406643527637667466318473669542326169218665366423043579003388486634167642663495896607282155808331902351188500197960905672207046579647052764579411814305689137519860880916467272056778641442758940135016400808740387144508156358067955215018979153370552535153498477459720877329811204688208387543826122582132404214848454954722487086658061408795223805022202997613522014736983452121073860054851302343517756732701026667062765906277626879215457936330799698812755973057557620930172778859116538571207100424990838508255127616637334499680058645411786925302368790414768248611809358160197554369255458675450109457987698749584630551177577492043403656419968285163536823819817573531356497236154342689914525321673807925458651854768512396355389740863270148775362744448115581639629326362342160548500035000156097215446881251055505465713854173913142040976382500435185442521721 1280621090306136836905430957515936037402234477454745934521690712819395759293807181586595407328753254594737067183837214480653975382948435606491935728562330520960068057097522463921439680512435086277215927236277876803684463476091761270872178732015931843245605080622778443509116111998261398730325599554316539542665805946211005643139251754871744789808491516766117236298425120168863946965228345230771282139885701648759079499654446882670560033220853520144332226729874711752888298595537524642481261647832718239946170997889346409324513553013543000784222338936021280343985086761512114805003488776758469360877632325223325426104 7
'''
```

爆破e

解

```
from Crypto.Util.number import *
import gmpy2 as g2

M = bytes_to_long(b'BJD'*32)
C = 97915337055253515349847745972087732981120468820838754382612258213240421484845495472248708665806140879522380502220299761352201473698345212107386005485130234351775673270102666706276590627762687921545793633079969881275597305755762093017277885911653857120710042499083850825512761663733449968005864541178692530236879041476824861180935816019755436925545867545010945798769874958463055117757749204340365641996828516353682381981757353135649723615434268991452532167380792545865185476851239635538974086327014877536274444811558163962932636234216054850003500015609721544688125105550546571385417391314204097638250043518544252172 1
N = 12806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
#factordb(N)
Q = 99855353761764939308265951492116976798674681282941462516956577712943717850048051273358745095906207085170915794187749954588685850452162165059831749303473106541930948723000882713453679904525655327168665295207423257922666721077747911860159181041422993030618385436504858943615630219459262419715816361781062898911
P = 12824761438044119862164649541094830994699723979595166193127307292737403525975999214935790989614172618704654898641799470216704634421446450064793599725862853183779233471272196346700840716668786892900597116836186002453103681720507605496930442102268030498629595920834415742494350003282830101667563141502388677597 7

for E in range(1,100000):
    if g2.gcd(E,(P-1)*(Q-1)) == 1:
        if g2.powmod(M,E,N) == C:
            e = E
            print('e是：',e)

c = 12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755798009656547738616399025300123043766255518596149348930444599820675230046423373053051631932557230849083426859490183732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018698992935818206109087568166097392314105717555482926141030505639571708876213167112187962584484065321545727594135175369233925922507794999607323536976824183162923385005669930403448853465141405846835919842908469787547341752365471892495204307644586161393228776042015534147913888338316244169120
n = 13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037
#factordb(n)
p = 99855353761764939308265951492116976798674681282941462516956577712943717850048051273358745095906207085170915794187749954588685850452162165059831749303473106541930948723000882713453679904525655327168665295207423257922666721077747911860159181041422993030618385436504858943615630219459262419715816361781062898911
q = 13528342342754565102391613415651971710970939911355390783298877025940222669588052419908789637730363186679019200852965871637668432803207583609415615081102516333668116342087545174738986854920308174356190737926024066515316692750405937907655555870427565913313590682730618904080432357446881955340190512799952367606 7
d = g2.invert(e,(p-1)*(q-1))
m = g2.powmod(c,d,n)

print(long_to_bytes(m))

'''
e是： 52361
b'BJD{p_is_common_divisor}'
'''
```

[BJDCTF2020]rsa_output

题目

{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,2767}

{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,3659}

message1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599

message2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227

共模攻击，套用上面的脚本

解

```
from Crypto.Util.number import *
import gmpy2 as g2

def egcd(a,b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, gcd = egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, gcd

c1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599
e1=2767
n=21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111
c2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227
e2=3659

xy = egcd(e1,e2)
x = xy[0]
y = xy[1]
if x < 0:
    x = -x
    c1 = g2.invert(c1,n)
if y < 0:
    y = -y
    c2 = g2.invert(c2,n)

m = g2.powmod(c1,x,n) * g2.powmod(c2,y,n) % n
print(long_to_bytes(m))

'''
b'BJD{r3a_C0mmoN_moD@_4ttack}
'''
```

[ACTF新生赛2020]crypto-rsa0

题目

```
#解伪加密
output：
90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992371538834949647434361938539784599470602104117547005673877738257835729760037765213340036696350766324229143613179932145122130685778504062410137043635958208805698698169847293520149572605026492751740223509962069259610194152560033947435941060614738650327920730359549258750560797626266484523488562555758401666405193348626900639493165157502565459374982134762866374558034528907812644460307323698710448703598385686181765862060410550002979817332728160898060144008463923077420655593318749722748449920478494722033903501
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2

p = 90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992371538834949647434361938539784599470602104117
q = 75470056738777382578357297600377652133400366963507663242291436131799321451221306857785040624101370436359582088056986981698472935201495726050264927517402231
c = 50996206925961019415256003394743594106061473865032792073035954925875056079762626648452348856255575840166640519334862690063949316515750256545937498213476286637455803452890781264446030732369871044870359838568618176586206041055000297981733272816089806014400846392307742065559331874972274844992047849472203390350
e = 0x10001
n = q * p
d = g2.invert(e,(p-1)*(q-1))
m = g2.powmod(c,d,n)
print(long_to_bytes(m))

'''
b'actf{n0w_y0u_see_RSA}'
'''
```

[GWCTF 2019]BabyRSA

题目

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

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

assert(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()

N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348
5892529574488072710160684141364000652761487311065141015589377654873782315294379788472913014975827912743004473925
4000426610922834573094957082589539445610828279428814524313491262061930512829074466232633130599104490893572093943
8327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089
4893017835533139093353677106579181764397876304503083371232616288381063812002937833709293866217411974768789948460
3628344079493556601422498405360731958162719296160584042671057160241284852522913676264596201906163
m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987
2601271228867045888437397141984275092928765864026621968664695692987211578217309397974295874512167192856870946852
6098715927189829600497283118051641107305128852697032053368115181216069626606165503465125725204875578701237789292
9662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647
8162169987041255782240438121380402668583122143072829075559781925933961665015867471324884165433851519940553200317
3732520457813901170264713085107077001478083341339002069870585378257051150217511755761491021553239
m2=4874439857574051734266281883756571176042355079369675229932579721088722836983052384544657232142268714142767888
9120581861970398212429127367428240806276809718025112069143946721592402069107358506519993161000146910672957081386
3936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891
3675466181551005527065309515364950610137206393257148357659666687091662749848560225453826362271704292692847596339
5332290880388205320861094211585758410776012687131750978740835362490060189487894132387839228456334940236088652560
7196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954 6
```

先解出c1,c2,再用立方和公式，得到F1*F2的值，最后解方程

解

```
from Crypto.Util.number import *
import gmpy2 as g2
from sympy import *

N=636585149594574746909030160182690866222909256464847291783000651837227921337237899651287943597773270944384034858925957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061631
m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532394m2=48744398575740517342662818837565711760423550793696752299325797210887228369830523845446572321422687141427678891205818619703982124291273674282408062768097180251120691439467215924020691073585065199931610001469106729570813863936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891367546618155100552706530951536495061013720639325714835765966668709166274984856022545382636227170429269284759633953322908803882053208610942115857584107760126871317509787408353624900601894878941323878392284563349402360886525607196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954646e = 0x10001#factordb(N)p = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377747699q = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377748737d=g2.invert(e,(p-1)*(q-1))

c1 = g2.powmod(m1,d,N)#c1 = F1 + F2
c2 = g2.powmod(m2,d,N)

F1_F2 = (pow(c1,2) - (c2//c1)) // 3
#print(F1_F2)
x = Symbol('x')
result = solve([x**2 - x*c1 + F1_F2])#result为列表,其中的元素为字典
F1 = list(result[0].values())
F2 = list(result[1].values())

F1 = int(F1[0])
F2 = int(F2[0])
flag1 = long_to_bytes(F2)
flag2 = long_to_bytes(F1)
print(flag1 + flag2)

'''
b'GWHT{f709e0e2cfe7e530ca8972959a1033b2}'
'''
```

SameMod

共模攻击

题目

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

message1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311 1537349
message2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323 8309535
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2
'''
def egcd(a,b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, gcd = egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, gcd

c1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311 1537349
e1=773
n=626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9
c2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323 8309535
e2=839

xy = egcd(e1,e2)
x = xy[0]
y = xy[1]
if x < 0:
    x = -x
    c1 = g2.invert(c1,n)
if y < 0:
    y = -y
    c2 = g2.invert(c2,n)

m = g2.powmod(c1,x,n) * g2.powmod(c2,y,n) % n
print(m)

m=102108971031231191041011101191011161041051101071051161051151121111151151059810810 1125
#ASCII码
'''
list1 = [102,108,97,103,123,119,104,101,110,119,101,116,104,105,110,107,105,116,105,115,112,111,115,115,105,98,108,101,125]
for i in list1:
    print(chr(i),end='')

'''
flag{whenwethinkitispossible}
'''
```

[WUSTCTF2020]babyrsa

题目

```
c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 730698867716256428074357836610140626042647684817351458735088469257355216951 59
e = 65537
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2

c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 730698867716256428074357836610140626042647684817351458735088469257355216951 59
#factordb
e = 65537
p = 189239861511125143212536989589123569301
q = 386123125371923651191219869811293586459
d = g2.invert(e,(p-1)*(q-1))
print(long_to_bytes(g2.powmod(c,d,n)))

'''
b'wctf2020{just_@_piece_0f_cak3}'
'''
```

RSA4

低加密指数广播攻击

题目

```
N = 331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400220031202
142322434104143104244241214204444443323000244130122022422310201104411044030113302323014101331214303223312402430 4
024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134102101102 2
123324130302443133000130340402010444244312013000033411004243201020340144040401000344200122304221144200141300 4
c = 310020004234033304244200421414413320341301002123030311202340222410301423440312412440240244110200112141140201 2
240324022321312042130123032044220033000040114341021413212233112432420100141404224113423043222012411124021322031 0
113122122300402200312000211023002334114320140431134031113423014023141220133333314240242313433321130210241311111 1
424430032440123340034044314223400401224111323000242234420441240411021023100222003123214343030122032301042243

N = 302240000040421410144422133334143140011011044322223144412002220243001141141114123223331331304421113021231204 3
222331201214444342100412322141444132444344243023112221432244023024321022421322440320100201132240111210432321432 2
120342424313404431402221202434310004234200243233114430021421241403341412000434421133022402030122303333432424403 1
204240122301242232011303211220044222411134403012132420311110302442344021122101224411230002203344140143044114 c
= 112200203404013430330214124004404232100413210430003032331414233441442223434010422003340332031240300114400142 1
011210323444031213403212340044434414423302013011013404210222030200241332110202241413044304114424031012102010031 0
104334204234412411424420321211112232031121330310333414423433343322024400121200333330432223421433344122023012440 1
301341401423202210124024431040013414313121123433424113113414422043330422002314144111134142044333404112240344

N = 332200324410041111434222123043121331442103233332422341041340412034230003314420311333101344231212130200312041 0
443244311410330043331100210130201400200112220123000200413420400040022202102231221113141121243332111322303321240 2
242314121403130314444413440302442011142324442403003000334021303212130321334302040130424333000131402303012103411 3
333404044042124224011310320301334123133000433204030244001132400413032403432343014310240144013024232142402032 3
c = 100134441201411303224332041240022422243323340111242100124402414023421004103311314413032420110021013230404033 1
112042130442222220032440224424332242244441404334213011111133002221320303032442210113303221204204224310143434220 3
204121042113212104212423330331134311311111414320001124000211131212223434000340331204040104302143311203133432432 2
233041123400140301320214321011302112411344224134423120130421412120031022113003214040430121243320132404312 42
```

用中国剩余定理解

解

```
from Crypto.Util.number import *
import gmpy2 as g2

def CRT(n1,n2,n3,c1,c2,c3):
    M = n1*n2*n3
    M1 = n2*n3
    M2 = n1*n3
    M3 = n1*n2
    t1 = g2.invert(M1,n1)
    t2 = g2.invert(M2,n2)
    t3 = g2.invert(M3,n3)
    me = (c1*t1*M1+c2*t2*M2+c3*t3*M3) % M
    return me

n1 = 331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400220031202
142322434104143104244241214204444443323000244130122022422310201104411044030113302323014101331214303223312402430 4
024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134102101102 2
123324130302443133000130340402010444244312013000033411004243201020340144040401000344200122304221144200141300 4
c1 = 310020004234033304244200421414413320341301002123030311202340222410301423440312412440240244110200112141140201
224032402232131204213012303204422003300004011434102141321223311243242010014140422411342304322201241112402132203 1
011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343332113021024131111 1
142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230104224 3

n2 = 302240000040421410144422133334143140011011044322223144412002220243001141141114123223331331304421113021231204
322233120121444434210041232214144413244434424302311222143224402302432102242132244032010020113224011121043232143 2
212034242431340443140222120243431000423420024323311443002142124140334141200043442113302240203012230333343242440 3
120424012230124223201130321122004422241113440301213242031111030244234402112210122441123000220334414014304411 4
c2 = 112200203404013430330214124004404423210041321043000303233141423344144222343401042200334033203124030011440014
210112103234440312134032123400444344144233020130110134042102220302002413321102022414130443041144240310121020100 3
101043342042344124114244203212111122320311213303103334144234333433220244001212003333304322234214333441220230124 4
001304140142320221012402443104001341431312112343342411311341442204333042200231414411113414204433340411224034 4

n3 = 332200324410041111434222123043121331442103233332422341041340412034230003314420311333101344231212130200312041
044324431141033004333110021013020140020011222012300020041342040004002220210223122111314112124333211132230332124 0
224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230301210341 1
333440444042124224011310320301334123133000433204030244001132400413032403432343014310240144013024232142402032 3
c3 = 100134441201411303224332041240022422243323340111242100124402414023421004103311314413032420110021013230404033
111204213044222222003244022442433224224444140433421301111113300222132030303244221011330322120420422431014343422 0
320412104211321210421242333033113431131111141432000112400021113121222343400034033120404010430214331120313343243 2
123304112340014030132021432101130211241134422413442312013042141212003102211300321404043012124332013240431242
#注意到这里面没有5及以上的数，可能是5进制
n1 = int(str(n1),5)
c1 = int(str(c1),5)
n2 = int(str(n2),5)
c2 = int(str(c2),5)
n3 = int(str(n3),5)
c3 = int(str(c3),5)
e = 3

m_e = CRT(n1,n2,n3,c1,c2,c3)
m = g2.iroot(m_e,3)#元组
print(long_to_bytes(list(m)[0]))#元组转化为列表

'''
b'noxCTF{D4mn_y0u_h4s74d_wh47_4_b100dy_b4s74rd!}'
'''
```

[BJDCTF2020]easyrsa

题目

```
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:
792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957238967608439270640608430765700010466569665440915500631320395729288574379171519878197420557865479212319158495766529320839045374836918233315280988231245335970614780819892291676277372172668158897710387745411904374488916452938318807749919493290964391869664876907327364751380953182517883134591810800848971719184808713694342985458103006676013451912221080252735948993692674899399826084848622145815461035
321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764345875193333001424730103446948038851685575488012024959332262154377633292802421135565244984575595628729008116020569444239674037776233069618807576132463287296166430326289640729312720858669280459737993747118468251577810569651641785052325242458091792356075715671742288225616978886459685593436083753319880971571452643576267381416465563535009949241158757481983180362968986040970009382721959030567335658801505402753692396377939759233295987160033503082593214367525792910003555604315422296997599551411529147083624944821531074516133689541340669000932476620078917924889695194204723544890161235112845930914582554756929847982110124909416186720768653760704744796870875899095013638092474735905257054959409856997063285435182595072975256350228484926373012758638252270395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257340593418867512432699727777523828792840374332429770515173252464121351630658529772219078008818070507035946971986934393910652920479828595751686077438400189277752591616774327241995857205533223205609597944815508246597778148259837199479887191751476750839473044797477032996768176762549539444 1

'''
```

z是无用的

解

```
from Crypto.Util.number import *
import gmpy2 as g2

n = 153107451613368954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012490941618672076865376070474479687087589909501363809247473590525705495940985699706328543518259507297525635022848492637301275863825227039598933923293337609276373530522502741958214690234014438413950964102318435921014265918825734059341886751243269972777752382879284037433242977051517325246412135163065852977221907800881807050703594697198693439391065292047982859575168607743840018927775259161677432724199585720553322320560959794481550824659777814825983719947988719175147675083947304479747703299676817676254953944 41
#factordb(n)
p = 105909195259921349656664570904199242969110902804477734660927330311460997899731622163728968380757294196277263615386525795293086103142131020215128282050307177125962302515483190468569376643751587606016315185736245896434947691528567696271911398179288329609207435393579332931583829355558784305002360873458907029141
q = 144564833334456076455156647979862690498796694770100520405218930055633597500009574663803955456004439398699669751249623406199542605271188909145969364476344963078599240058180033000440459281558347909876143313940657252737586803051935392596519226965519859474501391969755712097119163926672753588797180811711004203301
c =792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957238967608439270640608430765700010466569665440915500631320395729288574379171519878197420557865479212319158495766529320839045374836918233315280988231245335970614780819892291676277372172668158897710387745411904374488916452938318807749919493290964391869664687690732736475138095318251788313459181080084897171918480871369434298545810300667601345191222108025273594899369267489939982608484862214581546103 5
e = 65537
d = g2.invert(e,(p-1)*(q-1))
m = g2.powmod(c,d,n)

print(long_to_bytes(m))

'''
b'BJD{Advanced_mathematics_is_too_hard!!!}'
'''
```

[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 = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849
200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
# c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218 04
```

只给出了e和d，可以根据

$ed \ mod \ \phi(n) = 1$,也就是$ed = k * \phi(n) + 1$ 爆破k得到phi，又因为p和q很相近，所以对phi开方，

解

```
from Crypto.Util.number import *
import gmpy2 as g2
import sympy

c = 538272316807382811069616855829420668175799114902277821127563301413483223874527233300721180839298617076705685
041174247415826157096583055069337393987892262764211225227035880754417457056723909135525244957935906902665679771
011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329
325989862799714580389275398925561527314030002104065450590144278781065362652430570631666316934179720575293875559
056568986738227803487467274114398257187962140796551136220532809687606867385639367743705527511680719955380746377
631156468689844150878381460560990755652899449340045313521804
d = 192757789460378997180354554381755091757239114661274621545069165641015199236033089003314276019834768862558492
033237408199644297630705859739088116815586223853301862194473329920810818581417946684450446816320036999656426592
102288867006255450475851245321743477782046804949431381829172705040075255171655040364714819714888440826468684669
421183872177535169634497538098603540476192567878694002978585681397003965675194698253985751038854876244634244299
130177295856208771681716034441114646928413796611120751233993432706102722878652008803981935732608482686334619834
015031227070217852728240847398084414687146397303110709214913
e = 0x10001

for k in range(1,100000):
    if (e * d - 1) % k == 0:
        phi = (e * d - 1) // k
        q = sympy.nextprime((list(g2.iroot(phi, 2))[0]))
        p = sympy.prevprime(q)
        if e * d == k * (q - 1) * (p - 1) + 1:
            break
n = p * q
m = g2.powmod(c, d, n)
print(long_to_bytes(m))

'''
b'NCTF{70u2_nn47h_14_v3ry_gOO0000000d}'
'''
```

[ACTF新生赛2020]crypto-rsa3

题目

```
from flag import FLAG
from Cryptodome.Util.number import *
import gmpy2
import random

e=65537
p = getPrime(512)
q = int(gmpy2.next_prime(p))
n = p*q
m = bytes_to_long(FLAG)
c = pow(m,e,n)
print(n)
print(c)

'''
177606504836499246970959030226871608885969321778211051080524634084516973331441644993898029573612290095853069264036530459253652875586279468778310551475469102271005664966581483818346830373661345538480119032512527264740476612742231377276886895358235330467787931319021434444087356108211678387174888599022428636831457390378511382354771000540945361168984775052693073641682375071407490851289703070905749525830483035988737117653971428424612332020925926617395558868160380601912498299922825914229510166957910451841730028919883807634489834128830801407228447221775264711349928156290102782374379406719292116047581560530382210049
'''
```

解

```
from Crypto.Util.number import *
import gmpy2 as g2

n = 17760650483649924697095903022687160888596932177821105108052463408451697333144164499389802957361229009585306926403653045925365287558626794687783105514754691022710056649665814838183468303736613455384801190325125272647404766127422313772768868953582353304677879313190214344440873561082116783871748885990224286368 3
c = 14573903785113823547710005409453611689847750526930736416823750714074908512897030709057495258304830359887371176539714284246123320209259266173955588681603806019124982999228259142295101669579104518417300289198838076344898341288308014072284472217752647113499281562901027823743794067192921160475815605303822100 49
q = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179231
p = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179293
e = 65537
d = g2.invert(e,(p-1)*(q-1))

print(long_to_bytes(g2.powmod(c,d,n)))

'''
b'actf{p_and_q_should_not_be_so_close_in_value}'
'''
```

[AFCTF2018]可怜的RSA

题目

```
GVd1d3vilXFfcHapEYuo5fAvliUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfVyz2Z5GdTlaknxvHDUGf5HCukokyPwK/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N8O6i9pc4Gg3O8soOlciBG/6/xdfN3SzSStMYIN8nfZZMSq3xDDvz4YB7TcTBh4ik4wYhuC77gmT+HWOv5gLTNQ3EkZs5N3EAopy11zHNYU80yv1jtFGcluNPyXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNqSVlDs12EqXEqqJ0uA==

-----BEGIN PUBLIC KEY-----
MIIBJDANBgkqhkiG9w0BAQEFAAOCAREAMIIBDAKCAQMlsYv184kJfRcjeGa7Uc/4
3pIkU3SevEA7CZXJfA44bUbBYcrf93xphg2uR5HCFM+Eh6qqnybpIKl3g0kGA4rv
tcMlJ9/PP8npdpVE+U4Hzf4lcgOaOmJiEWZ4smH7LWudMlOekqFTs2dWKbqzlC59
NeMPfu9avxxQ15fQzljhvcz9GhLqb373XDcn298ueA80KK6Pek+3qJ8YSjZQMrFT
+EJehFdQ6yt6vALcFc4CB1B6qVCGO7hICngCjdYpeZRNbGM/r6ED5Nsozof1oMbt
Si8mZEJ/Vlx3gathkUVtlxx/+jlScjdM7AFV5fkRidt0LkwosDoPoRz/sDFz0qTM
5q5TAgMBAAE=
-----END PUBLIC KEY-----
```

这道题对rsa模块有些不了解，所以有些地方还是没看懂

解

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
import rsa
import base64

e = 65537
n = 79832181757332818552764610761349592984614744432279135328398999801627880283610900361281249973175805069916210179560506497075132524902086881120372213626641879468491936860976686933630869673826972619938321951599146744807653301076026577949579618331502776303983485566046485431039541708467141408260220098592761245010678592347501894176269580510459729633673468068467144199744563731826362102608811033400887813754780282628099443490170016087838606998017490456601315802448567772411623826281747245660954245413781519794295336197555688543537992197142258053220453757666537840276416475602759374950715283890232230741542737319569819793988431443
p = 3133337
q = 254783260649374192922001721363994977190818429145282283164559062116931183219713999360047291348411629741442462714864396957860365881174246118819559509962196468073788222782856382615820991083394389495730341012151411561564087428438200480668308638143623798857203950823184628500029016056897618763191511473527300909575569408421442998873946787436077669378280944783364011594490358783068537162165483742734623865083073677131120730040113834189678949305540675824532489810220119228833744427368480459206763413618712317871634414675330768900817218821793691687872877247696426653999925560521448458786001262839688902730675753420611776244939
d = 40685323095637968945062081571376887101071282583953641068796265067780089581800389371225962228147745329208814617384003682732251813145363057622997620852359361894981877789705925642659156053278463569719075292492371037594961695406980434257386725363097812363238479558795136548210346872238413308479861486387077589791592947525897418830092737691183376310561638616788181330174858523356304969379437064297632669267222363890816482210483241578857794531426423253194786057696662915045699551293223226488108061800669870067752911145450890058278542054946679802045148816861503525629297739069240138879046006632734770010934163999215947575503644 9
flag = 'GVd1d3vilXFfcHapEYuo5fAvliUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfVyz2Z5GdTlaknxvHDUGf5HCukokyPwK/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N8O6i9pc4Gg3O8soOlciBG/6/xdfN3SzSStMYlN8nfZZMSq3xDDvz4YB7TcTBh4ik4wYhuC77gmT+HWOv5gLTNQ3EkZs5N3EAopy11zHNYU80yv1jtFGcluNPyXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNqSVlDs12EqXEqqJ0uA=='
flag = base64.b64decode(flag)
rsa_components=(n,e,d,p,q)
arsa=RSA.construct(rsa_components)
rsakey = RSA.importKey(arsa.exportKey())
rsakey = PKCS1_OAEP.new(rsakey)
print(rsakey.decrypt(flag))
```

[RoarCTF2019]babyRSA

题目

```
import sympy
import random

def myGetPrime():
    A= getPrime(513)
    print(A)
    B=A-random.randint(1e3,1e5)
    print(B)
    return sympy.nextPrime((B!)%A)
p=myGetPrime()
#A1=21856963452461630437348278434191434000066076750419027493852463513469865262064340836613831066602300959772632397773487317560339056658299954464169264467234407
#B1=21856963452461630437348278434191434000066076750419027493852463513469865262064340836613831066602300959772632397773487317560339056658299954464169264467140596

q=myGetPrime()
#A2=16466113115839228119767887899308820025749260933863446888224167169857612178664139545726340867406790754560227516013796269941438076818194617030304851858418927
#B2=16466113115839228119767887899308820025749260933863446888224167169857612178664139545726340867406790754560227516013796269941438076818194617030304851858351026

r=myGetPrime()

n=p*q*r
#n=85492663786275292159831603391083876175149354309327673008716627650718160585639723100793347534649628330416631255660901307533909900431413447524262332232659153047067908693481947121069070451562822417357656432171870951184673132554213690123308042697361969986360375060954702920656364144154145812838558365334172935931441424096270206140691814662318562696925767991937369782627908408239087358033165410020690152067715711112732252038588432896758405898709010342467882264362733
c=pow(flag,e,n)
#e=0x1001
#c=75700883021669577739329316795450706204502635802310731477156998834710820770245219468703245302009998932067080383977560299708060476222089630209972629755965140317526034680452483360917378812244365884527186056341888615564335560765053550155758362271622330017433403027261127561225585912484777829588501213961110690451987625502701331485141639684356427316905122995759825241133872734362716041819819948645662803292418802204430874521342108413623635150475963121220095236776428
#so,what is the flag?
```

有阶乘，经过查找可以知道用威尔逊定理，

威尔逊定理

```
当且仅当p为素数时：(p-1)! ≡ -1 (mod p)
```

看了一下加密函数可以知道，p和q是$B!\ modA$的下一个素数，A是素数，由威尔逊定理可得

$$(A-1)!\ modA=-1$$

可以化为:(A-1)(A-2)(A-3)⋯(B+1)B! mod A = A-1 ,化简可以得到(A-2)(A-3)⋯(B+1)B! mod A = 1

所以这个时候只需要求出(A-2)(A-3)⋯(B+1)模A的逆，就是B!值了

解

```
from Crypto.Util.number import *
import gmpy2 as g2
import sympy

def Wilson(A,B):
    X = 1
    for i in range(B+1,A-1):
        X = X*i
        X = X%A
    X_inv = g2.invert(X,A)
    not_pq = X_inv % A
    pq = sympy.nextprime(not_pq)
    return pq

c = 75700883021669577739329316795450706204502635802310731477156998834710820770245219468703245302009998932067080383977560299708060476222089630209972629755965140317526034680452483360917378812244365884527186056341888615564335560765053550155758362271622330017433403027261127561225585912484777829588501213961110690451987625502701331485141639684356427316905122995759825241133872734362716041819819948645662803292418802204430874521342108413623635150475963121220095236776428

A1 = 21856963452461630437348278434191434000066076750419027493852463513469865262064340836613831066602300959772632397773487317560339056658299954464169264467234407
B1 = 21856963452461630437348278434191434000066076750419027493852463513469865262064340836613831066602300959772632397773487317560339056658299954464169264467140596
p = Wilson(A1,B1)

A2 = 16466113115839228119767887899308820025749260933863446888224167169857612178664139545726340867406790754560227516013796269941438076818194617030304851858418927
B2 = 16466113115839228119767887899308820025749260933863446888224167169857612178664139545726340867406790754560227516013796269941438076818194617030304851858351026
q = Wilson(A2,B2)

n = 85492663786275292159831603391083876175149354309327673008716627650718160585639723100793347534649628330416631255660901307533909900431413447524262332326591530470679086934819471210690704515628224173576564321718709511846731325542136901233080426973619699863603750609547029206563641441541458128385583653341729359314414240962702061406918146623185626969257679919373697826279084082390873580331654100206901520677157111127322520385884328967584058987090103424678822642362733
r = n // p // q
phi = (p-1)*(q-1)*(r-1)
e = 0x1001
d = g2.invert(e,phi)

m = g2.powmod(c,d,n)
print(long_to_bytes(m))

'''
b'RoarCTF{wm-CongrAtu1ation4-1t4-ju4t-A-bAby-R4A}'
'''
```