BUUCTF RSA题目全解3

原创

宁嘉 于 2020-08-21 17:08:00 发布 3231 收藏 11

分类专栏： RSA加密 文章标签： RSA

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/MikeCoke/article/details/107973068

版权

RSA加密 专栏收录该内容

12 篇文章 9 订阅

订阅专栏

1.[WUSTCTF2020]babyrsa

爆破n,就行了

```
from gmpy2 import*
from libnum import*

p = 189239861511125143212536989589123569301
q =  386123125371923651191219869811293586459

c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 73069886771625642807435783661014062604264768481735145873508846925735521695159
e = 65537

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

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

flag{just_@_piece_0f_cak3}

2.RSA4

参考文章

CTF ——crypto ——RSA原理及各种题型总结

RAS4

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

```
# 将5进制数转换为10进制数  int('',5)
N1 = int('3313103242120000300202143122442322224001424104234131044411402030032430021043332142020312022124034002200312021423224341041431042442412142044444433230002441301220224223102011044110440301133023230141013312143032233124024304024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134102101102221233241303024431330001303404020104442443120130000334110042432010203401440404010003442001223042211442001413004',5)
c1 = int('3100200042340333042442004214144133203413010021230303112023402224103014234403124124402402441102001121411402012240324022321312042130123032044220033000040114341021413212233112432420100141404224113423043222012411124021322031011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343332113021024131111142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230104224324100413300013034040201044424431201300003341100424320102034014404040100034420012230422114420014130041',5)

N2 = int('3022400000404214101444221333341431400110110443222231444120022202430011411411141232233313313044211130212312043222331201214444342100412322141444132444344243023112221432244023024321022421322440320100201132240111210432321432212034242431340443140222120243431000423420024323311443002142124140334141200043442113302240203012230333343242440312042401223012422320113032112200442224111344030121324203111103024423440211221012244112300022033441401430441141',5)
c2 = int('1122002034040134303302141240044044232100413210430003032331414233441442223434010422003340332031240300114400142101121032344403121340321234004443441442330201301101340421022203020024133211020224141304430411442403101210201003101043342042344124114244203212111122320311213303103334144234333433220244001212003333304322234214333441220230124400130414014232022101240244310400134143131211234334241131134144220433304220023141441111341420443334041122402344',5)

N3 = int('3322003244100411114342221230431213314421032333324223410413404120342300033144203113331013442312121302003120410443244311410330043331100210130201400200112220123000200413420400040022202102231221113141121243332111322303321240224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230301210341133344044404212422401131032030133412313300043320403024400113240041303240343234301431024014401302423214240203233',5)
c3 = int('1001344412014113032243320412400224222433233401112421001244024140234210041033113144130324201100210132304040331112042130442222220032440224424333242244441404334213011111133002221320303032442210113303221204204224310143434220320412104211321210421242333033113431131111414320001124000211131212223434000340331204040104302143311203133432412330411234001403013202143210113021124113442241344231201304214121200310221130032140404301212433201324043123242',5)

N = [N1,N2,N3]
c = [c1,c2,c3]


# 中国剩余定理算法
def chinese_remainder(modulus, remainders):
    Sum = 0
    prod = reduce(lambda a, b: a*b, modulus)
    for m_i, r_i in zip(modulus, remainders):
        p = prod // m_i
        Sum += r_i * (inverse(p,m_i)*p)
    return Sum % prod
e = 3

# print(chinese_remainder(N,c))
pow_m_e = chinese_remainder(N,c)

# pow_m_e = 17446992834638639179129969961058029457462398677361658450137832328330435503838651797276948890990069700515669656391607670623897280684064423087023742140145529356863469816868212911716782075239982647322703714504545802436551322108638975695013439206776300941300053940942685511792851350404139366581130688518772175108412341696958930756520037
m = iroot(pow_m_e,e)[0]

print(long_to_bytes(m))
```

flag{D4mn_y0u_h4s74d_wh47_4_b100dy_b4s74rd!}

3.[AFCTF2018]你能看出这是什么加密么

简单的RSA加密

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

p=int('0x928fb6aa9d813b6c3270131818a7c54edb18e3806942b88670106c1821e0326364194a8c49392849432b37632f0abe3f3c52e909b939c91c50e41a7b8cd00c67d6743b4f',16)

q=int('0xec301417ccdffa679a8dcc4027dd0d75baf9d441625ed8930472165717f4732884c33f25d4ee6a6c9ae6c44aedad039b0b72cf42cab7f80d32b74061',16)

e=int('0x10001',16)

c=int('0x70c9133e1647e95c3cb99bd998a9028b5bf492929725a9e8e6d2e277fa0f37205580b196e5f121a2e83bc80a8204c99f5036a07c8cf6f96c420369b4161d2654a7eccbdaf583204b645e137b3bd15c5ce865298416fd5831cba0d947113ed5be5426b708b89451934d11f9aed9085b48b729449e461ff0863552149b965e22b6',16)

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

d=invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```

```
flag{R54_|5_$0_$imp13}
```

4.[HDCTF2019]together（共模攻击）

先解析公钥

在线工具 SSL在线工具 SSL漏洞在线检测 工具网 买证书

公钥解析

获取公钥的加密类型、加密长度、其他参数，以及DER格式输出。

```
-----BEGIN PUBLIC KEY-----
MIIBITANBgkqhkiG9w0BAQEFAAOCAQ4AMIIBCQKCAQB1qLiqKtKVDprtS+NGGN++
q7jLqDJoXMlPRRczMBAGJIRsz5Dzwtt1ulr0s5yu8RdaufiYeU6sYIKk92b3yygL
FvaYCzjdqBF2EyTWGVE7PL5lh3rPUfxwQFqDR8EhIH5x+Ob8rjlkftIjHTBt1ThJ
JXvDBumXpQKGcBIknRaR9dwR1q8GU58/gIk5ND3eCTAadhrhLByWkHbFArxalx4Q
q8s2ZUe8lDc/N6V93EOFjbKbqqqtDmhniF6jdXQDAIwWTpx6+jmzxlCJoVHd2MBs
ZCcQhvklWtuKz4IYL4+iUpMKGHlhY1vCqFx2EzD4XIljFLP9rk7+9+CoyTuIVL/D
AgMACR0=
-----END PUBLIC KEY-----
```

解 析

详细信息

https://blog.csdn.net/MikeCoke

详细信息

密钥类型	RSA
密钥强度	2047
PN(e)	2333
PN(n)	14853081277902411240991719582265437298941606850989432655928075747449227799832389574251190347654658701773951599098366248661597113015221566041305501996451638624389417055956926238595947885740084994809382932733556986107653499144588614105694518150594105711438983069306254763078820574239989253573144558449346681620784979079971559976102366527270867527423001083169127402157598183442923364480383742653117285643026319914244072975557200353546060352744263637867557162046429886176035616570590229646013789737629785488326501654202429466891022723268768841320111152381619260637023031430545168618446134188815113100443559425057634959299
DER格式	30820120300d06092a864886f70d0101010500038201 0d00308201080282010075a8b8aa2ad2950e9aed4be34618dfbeabb8cba832685cc94f45173330100624846ccf90f3c2db75ba5af4b39caef1175ab9f898794eac6082a4f766f7cb280b16f6980b38dda811761324d513b3cbe65877acf51fc70405a8347c121207e71f8e6fcae39647ed2231d306dd53849257bc306e997a502867012249d1691f5dc11f06539f3f808939343dde09301a761ae12c1c969076c502bc5a971e10abcb366547bc94373f37a57ddc43858db29baaaaad0e68678a3757403008c164e9c7afa39b3c65089a151ddd8c06c64271086f9255adb8acf82182f8fa252930a187961635bc2a85c761330f85c896314b3fdae4efef7e0a8c93b8854bfc30202091d

https://blog.csdn.net/MikeCoke

密钥类型	RSA
密钥强度	2047
PN(e)	23333

PN(n)

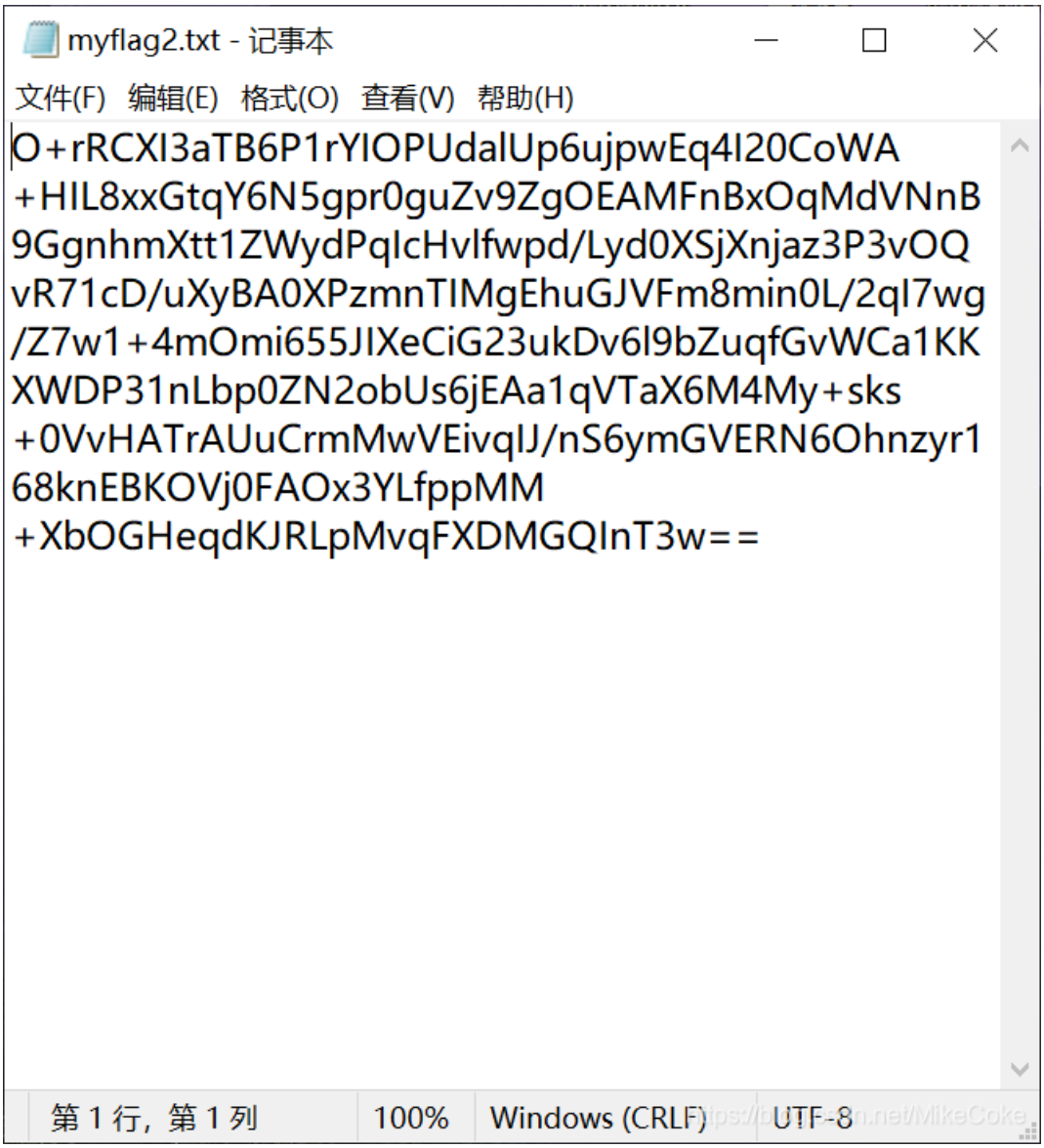
```
14853081277902411240991719582265437298941606850989432655928075747449227799832389574251190347654658701773951599098366248661597113015221566041305501996451638624389417055956926238595947885740084949809382932733556986107653499144588614105694518150594105711438983069306254763078820574239989253573144558449346681620784979079971559976102366527270867527423001083169127402157598183442923364480383742653117285643026319914244072975557200353546060352744266367867557162046429886176035616570590229646013789737629785488326501654202429466891022723268768841320111152381619260637023031430545168618446134188815113100443559425057634959299
```

DER格式

```
30820120300d06092a864886f70d0101010500038201 0d00308201080282010075a8b8aa2ad2950e9aed4be34618dfbeabb8cba832685cc94f45173330100624846ccf90f3c2db75ba5af4b39caef1175ab9f898794eac6082a4f766f7cb280b16f6980b38dda811761324d619513b3cbe65877acf51fc70405a8347c121207e71f8e6fcae39647ed2231d306dd53849257bc306e997a5028670122 49d1691f5dc11d6af06539f3f808939343dde09301a761ae12c1c969076c502bc5a971e10abcb366547bc94373f37a57ddc43858db29baaaaad0e6867805a3757403008c164e9c7afa39b3c65089a151ddd8c06c64271086f9255adb8acf82182f8fa252930a187961635bc2a85c761330f85c314b3fdae4efef7e0a8c93b8854bfc302025b25
```

TOP
https://blog.csdn.net/MikeCoke

对于题目给的这两个flag文件，去直接base64解密，发现解密失败。将base64编码转换为unicode，再转数字。



python之将byte转换为int类型函数 int.from_bytes 详解与原码反码补码的简单介绍

```python
# python3
import gmpy2, libnum, base64

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

n = 14853081277902411240991719582265437298941606850989432655928075747449227799832389574251190347654658701773951599098366248661597113015221566041305501996451638624389417055956926238595947885740084994809382932733556986107653499144588614105694518150594105711438983069306254763078820574239989253573144558449346681620784979079971559976102366527270867527423001083169127402157598183442923364480383742653117285643026319914244072975557200353546060352744263637867557162046429886176035616570590229646013789737629785488326501654202429466891022723268768841320111152381619260637023031430545168618446134188815113100443559425057634959299
e1 = 2333
c1 = int.from_bytes(base64.b64decode('R3Noy6r3WLItytAmb4FmHEygoilucEEZbO9ZYXx5JN03HNpBLDx7fXd2fl+UL5+11RCs/y0qlTGURWWDtG66eNLzGwNpAKiVj6I7RtUJl2Pcm3NvFeAFwI9UsVREyh7zIV6sI9ZP8l/2GVDorLAz5ULW+f0OINGhJmZm8FL/aDnlfTElhQ87LPicWpXYoMtyr6WrxjK6Ontn8BqCt0EjQ7TeXZhxIH9VTPWjDmFdmOqaqdVIT+LZemTgLNESwM5nn4g5S3aFDFwj1YiDYl0/+8etvKfOrfoKOwR0CxsRHagwdUUTES8EcHLmMGCxCkDZn3SzmmA6Nb3lgLeSgG8P1A=='), 'big')
e2 = 23333
c2 = int.from_bytes(base64.b64decode('O+rRCXI3aTB6P1rYIOPUdalUp6ujpwEq4I20CoWA+HIL8xxGtqY6N5gpr0guZv9ZgOEAMFnBxOqMdVNnB9GgnhmXtt1ZWydPqIcHvlfwpd/Lyd0XSjXnjaz3P3vOQvR71cD/uXyBA0XPzmnTIMgEhuGJVFm8min0L/2qI7wg/Z7w1+4mOmi655JIXeCiG23ukDv6l9bZuqfGvWCa1KKXWDP31nLbp0ZN2obUs6jEAa1qVTaX6M4My+sks+0VvHATrAUuCrmMwVEivqIJ/nS6ymGVERN6Ohnzyr168knEBKOVj0FAOx3YLfppMM+XbOGHeqdKJRLpMvqFXDMGQInT3w=='), 'big')

a, b = exgcd(e1, e2)

m = pow(c1, a, n)*pow(c2, b, n)%n
# m = gmpy2.powmod(c1, a, n) * gmpy2.powmod(c2, b, n) % n
print(libnum.n2s(m))
```

flag{23re_SDxF_y78hu_5rFgS}

5.[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=2185696345246163043734827843419143400006607675041902749385246351346986526206434083661383106660230095977263239777348731756033905665829995446416926446723440 7
#B1=2185696345246163043734827843419143400006607675041902749385246351346986526206434083661383106660230095977263239777348731756033905665829995446416926446714059 6

q=myGetPrime()
#A2=1646611311583922811976788789930882002574926093386344688822416716985761217866413954572634086740679075456022751601379626994143807681819461703030485185841892 7
#B2=1646611311583922811976788789930882002574926093386344688822416716985761217866413954572634086740679075456022751601379626994143807681819461703030485185835102 6

r=myGetPrime()

n=p*q*r
#n=8549266378627529215983160339108387617514935430932767300871662765071816058563972310079334753464962833041663125566090130753390990043141344752426233223265915304706790869348194712106907045156282241735765643217187095118467313255421369012330804269736196998636037506095470292065636414415414581283855836533417293593144142409627020614069181466231856269692576799193736978262790840823908735803316541002069015206771571111273225203858843289675840589870901034246788226436273 3
c=pow(flag,e,n)
#e=0x1001
#c=7570088302166957773932931679545070620450263580231073147715699883471082077024521946870324530200999893206708038397756029970806047622208963020997262975596514031752603468045248336091737881224436588452718605634188861556433556076505355015575836227162233001743340302726112756122558591248477782958850121396111069045198762550270133148514163968435642731690512299575982524113387273436271604181981994864566280329241880220443087452134210841362363515047596312122009523677642 8
#so,what is the flag?
```

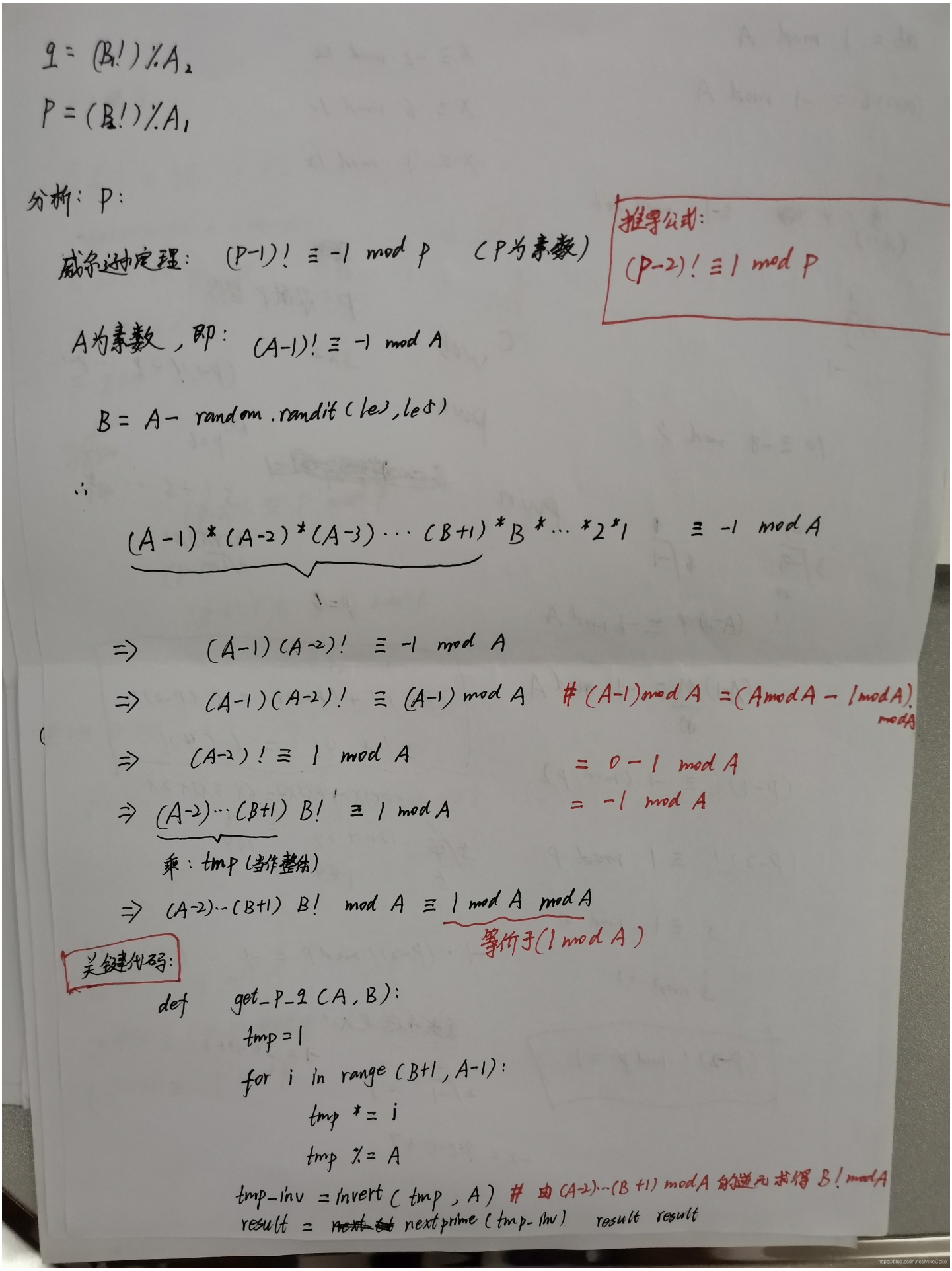
参考wp:

RoarCTF2019 babyRSA

数论考点：

三个重要的同余式——威尔逊定理、费马小定理、欧拉定理 + 求幂大法的证明

关键代码分析：


q = (B!) % A
p = (B!) % A
分析：p：
威尔逊定理：(p-1)! ≡ -1 mod p (p为素数)
推导公式：
(p-2)! ≡ 1 mod p
A为素数，即：(A-1)! ≡ -1 mod A
B = A - random.randit(le,les)
∴
(A-1)*(A-2)*(A-3)…(B+1)*B*…*2*1 ≡ -1 mod A
⇒ (A-1)(A-2)! ≡ -1 mod A
⇒ (A-1)(A-2)! ≡ (A-1) mod A
(A-1)mod A = (AmodA - 1modA) modA
= 0 - 1 mod A
= -1 mod A
⇒ (A-2)!≡1 mod A
⇒ (A-2)…(B+1) B! ≡ 1 mod A
乘：tmp (当作整体)
⇒ (A-2)…(B+1) B! mod A ≡ 1 mod A mod A
等价于(1 mod A)
关键代码：
def get_p_q(A,B):
tmp=1
for i in range(B+1,A-1):
tmp *= i
tmp %= A
tmp_inv = invert(tmp,A) # 由(A-2)…(B+1) modA 的逆元求得 B! modA
result = nextprime(tmp_inv) result result

解密代码：

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



def get_p_q(A,B):
    tmp = 1
    for i in range(B+1,A-1):
        tmp *= i
        tmp %= A
    tmp_inv = invert(tmp,A)
    result = nextprime(tmp_inv)
    return result
A1=2185696345246163043734827843419143400006607675041902749385246351346986526206434083661383106660230095977263239
7773487317560339056658299954464169264467234407
B1=2185696345246163043734827843419143400006607675041902749385246351346986526206434083661383106660230095977263239
777348731756033905665829995446416926446714059 6

A2=1646611311583922811976788789930882002574926093386344688822416716985761217866413954572634086740679075456022751
6013796269941438076818194617030304851858418927
B2=1646611311583922811976788789930882002574926093386344688822416716985761217866413954572634086740679075456022751
6013796269941438076818194617030304851858351026

n=8549266378627529215983160339108387617514935430932767300871662765071816058563972310079334753464962833041663125 5
660901307533909900431413447524262332326591530470679086934819471210690704515628224173576564321718709511846731325
5421369012330804269736196998636037506095470292065636414415414581283855836533417293593144142409627020614069181466
2318562696925767991937369782627908408239087358033165410020690152067715711112732252038588432896758405898709010342
467882264362733
c=7570088302166957773932931679545070620450263580231073147715699883471082077024521946870324530200999893206708038 3
9775602997080604762220896302099726297559651403175260346804524833609173788122443658845271860563418886155643355607
6505355015575836227162233001743340302726112756122558591248477782958850121396111069045198762550270133148514163968
4356427316905122995759825241133872734362716041819819948645662803292418802204430874521342108413623635150475963121
220095236776428

p=get_p_q(A1,B1)
q=get_p_q(A2,B2)

r = n//p//q

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

e=0x1001

d= invert(e,phi)

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

flag{wm-CongrAtu1ation4-1t4-ju4t-A-bAby-R4A}

6.[ACTF新生赛2020]crypto-rsa3

由题目可以得到值 n,c,e，已知p,q接近，所以可以yafu爆破获取到pq

```
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)
```

```
factor(17760650483649924697095903022687160888596932177821105108052463408451697333144164499389802957361229009585306926403
65304592536528755862679468778310551475469102271005664966581483818346830373661345538480119032512527264740476612742231377
76888695358235330467787931319021434444087356108211678387174888599022428636 83)


fac: factoring 17760650483649924697095903022687160888596932177821105108052463408451697333144164499389802957361229009585306
06926403653045925365287558626794687783105514754691022710056649665814838183468303736613455384801190325125272647404766127422
23137727688689535823533046778793131902143444408735610821167838717488859902242863683
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
Total factoring time = 0.7759 seconds


***factors found***

P155 = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715
58000480163426850572525556502151981717929 3
P155 = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715
58000480163426850572525556502151981717923 1

ans = 1
```

https://blog.csdn.net/MikeCoke

Python3脚本

```
from libnum import *
from gmpy2 import *

c = 14573903785113823547710005409453611689847750526930736416823750714074908512897030709057495258304830359887371176539714284246123320209259266173955588681603806019124982999228259142295101669579104518417300289198838076344898341288308014072284472217752647113499281562901027823743794067192921160475815605303822100 49

p = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179293
q = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179231
e=65537
n = p*q

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

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

flag{p_and_q_should_not_be_so_close_in_value}

7.[RoarCTF2019]RSA

题目

```
A=(((y%x)**5)%(x%y))**2019+y**316+(y+1)/x
p=next_prime(z*x*y)
q=next_prime(z)
A =  26833491826787145242474695127934760098610147810049249054841274803081613777681928680615618865770486464323821
28960881487463427414176114486885830693959404989743229103516924432512724195654425703453612710310587164417035878308
39067661259284875028738731812942419520862344029464781736774087821194914752628709129830748050289746227910257255 6
82223166943827931747482847908971904638641197110544872391059471041809397704417994980037322435472917983339321982777
89389078869290217569511230868967647963089430594258815146362187250855166897553056073744582946148472068334167445499
93144715183575352611863187563278900161832284122537 24
n =  11793080604350737432598229182302728514880723911798736960958351535388981485608809967145439434081676124297446
22684359117650455763777677115931004169320198318890593331669462631848612879757229549922197664930896308108769847811
13645362450398009234556085330943125568377741065242183073882558834603430862598066786475299918395341014877416901 1
85392905676043795425126968745185649565106322336954427505104906770493155723995382318346714944184577894150229037755
84345972425648152991749501477544269502514192049173765173605050245496917236833581708234167579730593547841426014366
51950081115903679503467636002892830197978052829411493334712 7
c =  41971850275428383625653350824107291609587853887037624239544762751558838294718672159979929266922528917912189
12471327367394805146422651960580374517134072434370583219855468019679862326380661799807249602601994047632497169 9
28551159371970207365741517064295956376809297272541800647747885170905737868568000101029143923792003486793278197 5
13267166802127261110994392625893410509439134010676738518851143147097060166221572850232724967935952810540742604 1
11621381593484331789489888321536228959936610101808151321512072829713135243906693045228182944658656206224252732 6
72575620261776042653626411730955819001674118193293313612128
```

尝试分解一下n，factordb

顺利获取到p,q的值

发现题目并没有给出e的值，可以采取分析爆破的方式计算出e，但会很麻烦。尝试用一下e的一般值 **65537（0x10001）** 成功解出flag

```
from gmpy2 import*
from libnum import*
n =  11793080604350737432598229182302728514880723911798736960958351535388981485608809967145439434081676124297446
22684359117650455763777677115931004169320198318890593331669462631848612879757229549922197664930896308108769847811
13645362450398009234556085330943125568377741065242183073882558834603430862598066786475299918395341014877416901 1
85392905676043795425126968745185649565106322336954427505104906770493155723995382318346714944184577894150229037755
84345972425648152991749501477544269502514192049173765173605050245496917236833581708234167579730593547841426014366
51950081115903679503467636002892830197978052829411493334712 7
c =  41971850275428383625653350824107291609587853887037624239544762751558838294718672159979929266922528917912189
12471327367394805146422651960580374517134072434370583219855468019679862326380661799807249602601994047632497169699
28551159371970207365741517064295956376809297272541800647747885170905737868568000101029143923792003486793278197051
32671668021272611109943926258934105094391340106767385188511431470970601662215728502327249679359528105407426045 1
11621381593484331789489888321536228959936610101808151321512072829713135243906693045228182944658656206224252732966
72575620261776042653626411730955819001674118193293313612128
p = 84286804568139093453973995920184755228498017995887966793307845395096856615166214726700629357176546313727059 4
15113869577898616511138042880654559358807836533131308423001461871441295958484342158667416268832194288936991239 20
31882620994944241987153078156389470370195514285850736541078623854327959382156753458569
q = 13991609558311089513359683322750669367930670987317402487689102335586078198117591644632304473291306688078691 8
62908902349931170340848915118188656853562100864499797198218242670659255129108400798338791100626144251963540545 70
77292515085160744169867410973960652081452455371451222265819051559818441257438021073941183
e = 65537
phi = (p-1)*(q-1)
d = invert(e,phi)

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

flag{wm-l1l1ll1l1l1l111ll}

8.[AFCTF2018]可怜的RSA

公钥解析，RSA签名加密

```
from gmpy2 import*
from libnum import*
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from base64 import b64decode
e = 65537
n = 79832181757332818552764610761349592984614744432279135328398999801627880283610900361281249973175805069916210179560506497075132524902086881120372213626641879468491936860976686933630869673826972619938321951599146744807653301076026577949579618331502776303983485566046485431039541708467141408260220098592761245010678592347501894176269580510459729633673468068467144199744563731826362102608811033400887813754780282628099443490170016087838606998017490456601315802448567772411623826281747245660954245413781519794295336197555688543537992197142258053220453757666537840276416475602759374950715283890232230741542737319569819793988431443
p = 3133337
q = 25478326064937419292200172136399497719081842914528228316455906211693118321971399936004729134841162974144246271486439695786036588117424611881955950996219646807378822278285638261582099108339438949573034101215141156156408742843820048066830863814362379885720395082318462850002901605689761876319151147352730090957556940842144299887394678743607766937828094478336401159449035878306853716216548374273462386508307367713112073004011383418967894930554067582453248981022011922883374442736848045920676341361871231787163441467533076890081721882179369168787287724769642665399992556052144845878600126283968890273067575342061776244939
phi = (q-1)*(p-1)
d = int(invert(e,phi))

key = RSA.construct((n,e,d,q,p))
cipher = PKCS1_OAEP.new(key)

with open('C:\\Users\\MIKEWYW\\Desktop\\create密码题目\\1212BUUCrypto\\[AFCTF2018]可怜的RSA\\flag.txt','rb') as f
:
    f = f.read()
    c = b64decode(f)
    flag = cipher.decrypt(c)
    print(flag)
```

9.RSA & what

题目分析：

C: > Users > MIKEWYW > Desktop > create密码题目 > 1212BUUCrypto > RSA & what > rsawhat > rsa.py > bytes_to_long

```python
from Crypto.Util.number import bytes_to_long, getPrime
from random import randint
from gmpy2 import powmod

p = getPrime(2048)
q = getPrime(2048)
N = p*q
Phi = (p-1)*(q-1)
def get_enc_key(N,Phi):
    e = getPrime(N)
    if Phi % e == 0:
        return get_enc_key(N, Phi)
    else:
        return e
e1 = get_enc_key(randint(10, 12), Phi)
e2 = get_enc_key(randint(10, 12), Phi)

fr = open(r"./base64", "rb")#flag is in this file
f1 = open(r"./HUB1", "wb")
f2 = open(r"./HUB2", "wb")
base64 = fr.read(255)
f1.write("%d\n%d\n" % (N, e1))
f2.write("%d\n%d\n" % (N, e2))
while len(base64)>0:
    pt = bytes_to_long(base64)
    ct1 = powmod(pt, e1, N)
    ct2 = powmod(pt, e2, N)
    f1.write("\n%d" % ct1)
    f2.write("\n%d" % ct2)
    base64 = fr.read(255)
fr.close()
f1.close()
```


https://blog.csdn.net/MikeCoke

考点：共模攻击+base64隐写

Base64隐写

base64在解码的过程：
1,先去掉'='
2,8位一读
3,末尾不足8位的地方舍去.

EXP:

```
from gmpy2 import*
from libnum import*
from Crypto.Util.number import*
import base64
import binascii

N =785095419718268286866508214304816985447077293766819398728046411166917810820484759314291028976498223661229395009474063173705162627037610993539617751905443039278227583504604808251931083818909467613277587874545761074364427549966555519371913859875313577282243053150056274667798049694695703660313532933165449312949725581708965417273055582162959945876009759701248114962700808969770769460001027010302609905981814664472080547133915263137006813410939222403174281735990316241251551882164894768256061915211820349691203432876911813003996835154148092627004575258766918081802577303517076736603806989738846423068988100006336848787154028231435491398507329828974596980896495611907468 50
```

6981302994580802555823126968731492100282408981378228884925599576650679365733563675897845931190166240724338727445
3743200591166849445573333068938514121465309188801778204904343486262030678343616985656417592987110066991343898089
9219579329897753233450934770193915434791427728636586218049874617231705308003720066269312729135764175698611068808
4040541255815401149564636032402224979193846917187440140025542016023959693129999941595995360263598790602180564963
4574545749391977133760117744989906657985763003635087109045264983077502969548857557498507842856005425318086372536
4147
e2 = 599
c21=5921690793720937273061002160113958578256463239342894809760736290375439229020981209011384544621771599963766541762482389791325287283275903010989661399831579806123205634965461286449677310007166977051040790391562767148721474633508113033932606227070249525435098916922462462779658234144603268112400480605436565886886044523538997790688251209102821670047153397631877347971803269761322133250546971653204791663565625180298059277416566051748097263975657722715620660780761054917459039865978774003702067189549752887210720483336786090550081358090893042290153643484909249740974037346272652976371718188494617665236915952416138787098655064365882689991633429450704953381536005205374985394573965828046929592966127157525731402961357849332061460914366179795997497743306999466375914063562894097160840344510490947152021962034860883687917441076292716473202732598369153127942972465895010086662991657177225077028660334542157832400255043561576644548617552862857777635851777517962526550082063830247078830775137458633120793497907750940807075023928669463257969144506022644625887220522974308276817508273490943239683376703112729337858388506493761156672238216654359115063518914899856275066154920056170986154325225642041528877672441299856810836577833565577566543351865
c22=37394064641683274087873325570756775303371658344840200078920276751192021038283034395555365411148672833398055731979936251496062787901679749139898120077688327309799162306476418727590019068467479776316757043101794488571281603857011858929145230536693665344088637343056352226255909860064204860925504273010869845631264808149870249805946135429783101292476788266914183353005775775279516236964264354978352281670847380077509142702510019213295214790476628486508089899960856001973093614108632385268021278775237672629215151509849985601366471548657911633165030732852239662164410256374522290435100973237243810569763022881368432601639227066929130352224454967160088894658153500454635574421168039073125730994190258730335313995110224486527029541447448879833540463045848970663980518657387481458673674623235884967747753367196834415424296328941556948757989591066099904357873746130040693782892481800265829276988218166878450143925413199684894812078156215886149588382784813942586224957645468913368100954936131446081865899595909822899570220226864963536310554997593239533507652113760428852008204012128661492298655465270005614896651417893595236303696321761987989967138360463841656795042135054620443490211315672000628272088959128885027107607494192771567830605717676
c23=5276309264606229365713856498417582144534168490394124010874434443171018570909047114855381070588230560858405390733459207928713682323554753945710983805968354685099973405056043337305477995609988229897474737803077797177155227877244717247664940907839710305946710131682097176867204485795826183784595679790278222719186531696224281538561989078100402243402703624134324950296721232613754009271598315377607099747787081605832526137843582348585831745447779792428879388275736048377668019983813799990764164446838910780938896860554827098386683561209160403521230190192550845137696038038149477455402871781463895141629127469677151547408635148210795315025361692278726239845037624912699964402638247841308097393317307911130514271613338911139923554527925901742472260184867006155685916394389546655392794641252375016658273400573337832846825056894494591223849587792971710172231467812017222849378796490407258390572107476671173221581556101296039453719575783295926860377511293286210594572085395928518752176355791535642811387689327687977560321771898185211459970669952455197393424204574312274414636159697124503405934591531549523213548346449611477035753657620051149092241320817814986934780298878651345148641140988716451606506208491755612071246507420643583149811613605
c24=878643717869894032287788980700995761677735184497986972696235655324405091128398428096066576164931089523045507297743141510205398773596932655397899485316248305154465687329455511600999559204318307020870625816484054054599577072097104139505857517663273929851202628854185356185647194933800084230503413037858933077130371493074778305367582836810935176178201691814207961053386815822307883181084281320517937610149528373304562622728286273557014647405781979663326131273070372556472868234963559176423533279124400196218388703880918247486296374257591252146398851301631837523789087297735170532592125254945558809210526795125820515166042970982043635250810393823584839267270086793277190831388659692919118636303820971602309607380435755593302640182127744245277191532485638767600679314990293842289932538625019393377585143774720119332792731811448303811698493878937993907550520930691796055794857103436555700285925958824366324265276544528954317587151265801649024102864226372150984763160423679167794310522675457694959947237211299436162948796423055458949129146329804550317558790874015753106997654084736061667271379342245159984166251222130562088000950779331031506992726501161516747024384630627344727140049261036683785068040027400455479647166935363494476608505805
c25=2053149622045115003528583722541325331675499608254989496185148415707031992648674315807546742759905544781406370414278421113917468832574471200359476214568638909340620440107954430592817363469761757724150348383346827266352634326555378529421773348880252837486115761715342514618473495665056282905872241508696403864376233712497431652603966752206833021428056463689069305751406286100039191319992958550121511139329481821879998270328930459698907047500008158

```
17551008543229026450202373689910474631683074222694639502702982082579183187085738264722132273460502621007309391833
3124749430755560033555094234052653628137203661213871388109886630316942550199897840000882987308096559200937117620
8668290074288903681417933657472279670688597862835627506340169978450918788539270346340385928840299573889292189531
7380821664087340463814235164676943289713854219073148142834893226193865700461835565723839807772771733492093306834
2434365817978101507225937857613044222298496307116620764258558982206159728246785086805073795772642371376169423187
9497037175627546427449730638216214828463003483408928375620315193290871300316930139260521382533279767663839278693
7504094194932807533684515088026582722207676247663906392853084336072552532827023837621497559355189220755846375124
94819
c26=2714536347325026133789481612564709912600527787991287898396245158091435273632068132195800981969575102916484936981444975673920652512448440749927346694902962939973861983592803166559046916393674822032100518091259044104315069
2523837484385634324327650828064105969093893095747443451830864661895900421683113009987353271437240211779666656067
7624822509159287675432413016478948594640872091688482149004426363946048517480052906306290126242866034249478040406
3519400882310814561091957994429967996416471675526895646133464152479068520555884983056659284508287561521030966292
7476060152873763941536146794134998221364145496796272387503263826731193504233458491389733855395396187743938958879
3074211502597238465542889335363559052368180212013206172712561221352833891640659020253527584706465205486408990762
7592308421920283810485634377245284091747900227525557512795782713125166158329880702730769957185428522011430144840 2
3225641911363167934317168063163077526648873817370735712313936882508704378584216904994323753718812936727573098478
9479909103397937113837824575137021012333461552176687570010445744268373840742899299977372834041925102853718964831
2252504072795784650085375426596736856862427733791319048908651106991904515344454345339191276589768747210295861681
06207

e1 = 1697
c11=4126295261631507486193280913067422676757405780118000624771741897821512739707835312275797585403649704853501579443215791082322210723971359340340644814978870796411318088382427438115114513550244369830505720209250656443555664
3462561813320302421594153492611389293798852091893906144160691555651624605734958992149435138316003628082602460535
1878408056180907759973804117263002554923041750587548819746346813966673034182913325507826219961923932100526305289
8949652166082542521883985801395451896818758240894561950449845858249383845219053342899064224541529768348673046932
9246667635576017323240775325625631754619017199527625892461353317989846768335893475199965519679016843834319822918
3747091108262988777659858609744709324571850262293294975336628234767258858873839342596887193772615000676401522431
5183106483039755935829650211891822469869573492531567365260716399738440390689964042905484746406688518560782010933
3542541284229560491906548730134090157380961754918510607279879915972637523512526050915883299670192787871308475333
4549129580912412168594170659605421750204835970231909591063407612779337478065175988365401590396247576709343727196
1060584771669456701178689890259030239981428503389569858161318053495490593770474771312708475790956283845696456368
21650
c12=4946443479437105452246788319415890865727007924654595587707822135500697094585683496869986605418101668720340415847674871501401111517882214600278971932482734616074110278159848839693962206263586250417815582778049302126542967
0405589068379694132771275879777082000662328914699000011491529353963976684691027403424560774623074085193815839056
2286057002223177609606376329007676845450142537930798148258428701466415483232670659815791064681384406494388237742
3307862255573039880254680368200829597120507330958605468604685758570846160691320510948829192537452347620297591247
7634804758775589712357512350697614090056523884075284185671361336825007192617187321389791479411546689071912329946
9964019450899291410760762179836946570945555295288184698184555018368687708432612286248476073758067175481771199066
5815728701754600160171004144793464370342917848371322408913219316014944149089277132084489272210957458023800144418
4113988239137841043876488459793877386877189625232951744006867353246837284083051021858525543200069026522601657331
3570977945083879214961394087065558376158826938257664840570952233832852869328785568175434516247720356520242602299
5103743174881827387327000788796657459096037664821001380014170236806477178243231433888578175957661721528834842747
18248
c13=1529422835997283071681441373701272126726118940720387321260410981026288310530009867592602712106719220705559480236885965754158229840261590105744043594746704286785182621750338805139843729097489927278283816944167767409810217
3054537400297403789653494456712454327273761838064677107180487879658598378336055376182832581782026020482000442197
9881871027255562690952334900616675606524933557440263648233514757200263521499508373975003431306847453046714027687
1083969457198034444449540793084049471262163955265512921047220478781783732078860330718572778579979322552513159828
3789216442129820207394591918777985678589271725174670453731500377136973785489659517048515259101367694241813427853
4037654467840633528916812275267230155352077736583130992587670941654695382287023971261529987384520843829695778029
3117864312274091890192058183519115727571455569936066434643361968023502046160562864972460168001050031430461206086
7349619675872055277677279660967053705633199689432277926763528147248155981983904242401717171830321405972056848493
9239370144038161541354254182769979771948759413102933987773401644506930205164891773826513161783736386604783484446
3457449571194697992317963683249275706944966794533139275623456566902404146244313046462485992260465247023641310959
64335
c14=7971798893624795126548915758369795603189347785885418699105152916187947848828174406231860047090612096000228288651147729455560650308316944933517486442418070108020399332999622656620383469386952579769596961006599194139672395
```

```
90326800190825068164430415983004776257934330806643464705864163858546921244263485872110265686676948058495547807940337647140165217114675572848467372363749901213168098338199968215928326390240264115204073302062812653901307639481656945745121405187756030401820298187718667495487619388706055901743308879498474208778292401314909024326020056810851808072941768376460625680948757669458903829717900154901633850881446735490850796350832629751542062696791424128974382317197049332586607793107373026802654454377719777499591107449593685862930820160679275485649674008459923800761075227555665317606288233745197187637403782955855355917528873392229473971841163267067999215154311856367408257077827423737834757810526742572929102138439861329874668100272750524167746933634461845189018992025028286703094526223475329326788749908099306825757386538762893841514968071941463086143688210066606268709897846970451602310694284589611077512077710937773946168563052933356038921783275207565543333659751142359811734513681316804048508327731473330137169201c15=123111353650401158556639983459870663057297871992927053886971224773529636525110628183715748795987525113177540092814119928708272290370336537110381023134637759740716140969662183269370676630325583385284994943164692397459103195434968057377474610500216801375394703781249039351368816958227409657934091741509357152328382960684515093945552479461382281913961956745154260686029997827565075768703774895750561575155143606297116391666385705899138085693913246313778033627210312268959737394553510894720099165193981333775907531107232556909478156441457899797515694348816961762796703443502856101079430585547997496001098926600499728389113862894833789669213630332988693669889340482430613291490613803204484751470676686041002772556117213612152322606737150858116122936539131795111263513114569794532805886643087299918196635113037777138666914296986040549274559835214505300618256105508764026461518876579387159881983544667258537064954616097750399839661065797883103731694314852301848272092388637114950059216922969842082648527035538090054093890365647676119748995243416337805666557501345234056968476142608491830438065401219751688687373709390057521910942736632126729711606256158399963682990881473178216060827021373776598901281958527655543318413664277921492723185984c16=368698068159360469118481958174058173502598908714830631843737283979689094584326250460253762902147299140383875347317622379783390117248588188601811788116394689962062947114958538073112400137862268842651181195463772721545556153631052361928782927033314735476230217443170348194166245628962261945236397935730280066662362718123907590362358674958032559058436364472522254138710387626578013456475844939175762634715873472026643919085701403891269032046023910939908271886750901997506173037735748219263871944788751918288149712966745305193215308053026679259987118350198067611330784032814048893746638750773391689012978194364999209582684836843359983010560683802288735248003839114024908071392689640951650696104546775588087564443815421737828152279209062249310284570736524537774243878735332804559446465929969206179566757862867114475403538834002824025511581699583894501680795684596565269118578353757480158148605067079218529970961562758049559899642150777336217699380754130078042232170916046131322530463994567475953004045641722243339364055459218196544354370721333875235335684724435322000691330229791956856835082973379617011693947949662564151122465877061038196204282582459995390407219293171300888741615770939625794874283587364016871231742071982514498514295295
```

```
c1 = [c11,c12,c13,c14,c15,c16]
c2 = [c21,c22,c23,c24,c25,c26]

s = ''
a = gcdext(e1,e2)
for i in range(6):
    m = pow(c1[i],a[1],N)*pow(c2[i],a[2],N)%N
    s +=n2s(m)                                          #16进制转字符串
    #s += binascii.unhexlify(hex(m)[2:]).decode()      16进制转字符串（另一种写法）
ans=s.split('\n')   # 通过分隔符\n对 s 内的字符串进行切片操作

# 关键代码
libase64=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z
','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','0','1
','2','3','4','5','6','7','8','9','+','/']
def myb64_bytes(str):
    s=''
    temp=str.replace('=','')         #去掉=号
    for i in temp:
        s+=bin(libase64.index(i))[2:].zfill(6)
    return s

def bytes_to_string(str):
    s=''
```

```
    for i in range(0,len(str),8):
        s+=chr(eval('0b'+str[i:i+8]))
    return s

flag = ''
for i in ans:
    b=myb64_bytes(i)
    temp=len(b)%8
    if temp !=0:
        flag += b[-temp:]
    else:
        flag=flag

print(bytes_to_string(flag))
```

10.[网鼎杯 2020 青龙组]you_raise_me_up（离散对数）

涉及知识点：**离散对数**，可以通过sage来解决
也可以通过 python 第三方sympy库来解决。

这个考点以后可以写一篇专门的文章来解析离散对数及解决代码。

可以参考一下的文章：
1.[网鼎杯2020青龙组] you_raise_me_up
2.离散对数概念
3.Sympy常用函数总结

```
from sympy import*
求离散对数（如下15 = 7**3 mod 41）：
discrete_log(41, 15, 7)
```

```
来分析一下这道题
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Util.number import *
import random

n = 2 ** 512
m = random.randint(2, n-1) | 1
c = pow(m, bytes_to_long(flag), n)
print 'm = ' + str(m)
print 'c = ' + str(c)

# m = 391190709124527428959489662565274039318305952172936859403855079581402770986890308469084735451207885386318986881041563704825943945069343345307381099559075
# c = 666585139420321424585678945072365863252081679162179677590976689523300023402364287878602564495379799537321130848560539702412318008592411761080248597258449 9
```

从题目可以看出来，要解出flag，只需要一个公式即可。

c = m ** flag mod n (这里的等号是同余号)。
意思为 c同余m的flag次方 模 n

我们要解flag的值，用函数
discrete_log(n,c,m) #(模数，同余数，同余数)

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

n = 2 ** 512
# n = 13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006084096
m = 39119070912452742895948966256527403931830595217293685940385507958140277098689030846908473545120788538631898688104156370482594394506934334530738109955907 5
c = 66658513942032142458567894507236586325208167916217967759097668952330002340236428787860256449537979953732113084856053970241231800859241176108024859725844 99

flag = discrete_log(n,c,m)
print(n2s(flag))
```

flag{5f95ca93-1594-762d-ed0b-a9139692cb4a}