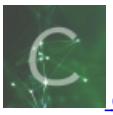
BUUCTF RSA（二）

原创

路由(_)生 于 2021-08-18 15:13:48 发布 196 收藏

分类专栏： crypto

本文链接：https://blog.csdn.net/qq_52193383/article/details/119756514



这里写目录标题

1.[BJDCTF2020]RSA

两个加密的n共用一个q，对n1,n2求最大公约数，进而分解n1,n2，分别求得p，q。

假设c3 = pow(294,e,n1)

,利用其进行枚举求得e,此时的e应该满足三个条件：

1.e<100000

2.(e,n1) = 1

3.pow(294,e,n1) = c3

解是解出来了，不过另一个明文加密的过程好像没有用到，仅仅只是提示了两个模数有共同因数。

```
c1 = 126416356178037461503322326463545962927078614802002075371991411836244383037571205700967412480202366669657557980096565477386163990253001230437662555185961493489304445998206752300464233730530516319325572308490834268594901837323037517440048741830625948568703186142899916759800635483164994869089232096275638715548756127020791005670186989293581820610908756816609739231410571755548292614103050563957170887621316711218796258448406532154572759413517536923392592250779499960732353697682418316292338500566993040344885346514140584683591984290846978754734175236547189249520430764458616139322877604201553414791388833816244169120
n1 = 135087741044602097433067140345467041372476273449811334618019534797360170214017258188084628983759947673756277494948396719445438224030599780738131224414076125306581689429878202567865830069470017117492301935423705709507055301679217028356271224014752510390007750173816339002224747273968237086950631362461156526222597696345913094217612695482609844261488246412850107309832153775092550112987378276216111580329764200116625478545156105979556288980735696841582256783334745439203265328934468498081128374766843900309764720539050698555222978506880269607011865434281398437839076243172747969262488295434134647541272088430703310630 37
q = 998553537617649393082659514921169767986746812829414625169565777129437178500480512733587450959062070851709157941877499545886858504521621650598317493034731065419309487230008827134536799045256553271686652952074232579226667210777479118601591810414229930306183854365048589436156302194592624197158163617810628989 11
p = 135283423427545651023916134156519717109709399113553907832988770259402226695880524199087896377303631866790192008529658716376684328032075836094156150811025163336681163420875451747389868549203081743561907379260240665153166927504059379076555558704275659133135906827306189040804323574468819553401905127999523676067
phi = (q-1)*(p-1)

c3 = 381631268825806469518166370387352035475775677163615730759454343913563615970881967332407709901235637718936184198930226303761876517101208677107311006065728014220477966000620964056616058676999878976943319063836649085085377577273214792371548775204594097887078898598463892440141577974544939268247818937936607013100808169758675042264568547764031628431414727922168580998494695800403043312406643527637667466318473669542326169218665366423043579003388486634167642663495896607282155808331902351188500197960905672207046579647052764579411814305689137519860880916467272056778641442758940135016400808740387144508156358067955215018
e = 2
while e<100000:
    if int(gmpy2.gcd(e,n1)) == 1:
        if pow(294,e,n1) == c3:
            print(e)
            break
    e += 1
d = int(gmpy2.invert(e,phi))
m = pow(c1,d,n1)
print(libnum.n2s(m))

n2 = 128062109030613683690543095751593603740223447745474593452169071281939575929380718158659540732875325459473706718383721448065397538294843560649193572856233052096006805709752246392143968051243508627721592723627787680368446347609176127087217873201593184324560508062277844350911611199826139873032559955431653954266580594621100564313925175487174478980849151676611723629842512016886394696522834523077128213988570164875907949965444688267056003322085352014433222672987471175288829859553752464248126164783271823994617099788934640932451355301354300078422233893602128034398508676151211480500348877675846936087763232522332542610 47
```

2.[BJDCTF2020]rsa_output

可以看出第一二行是（n,e），两个n相同，属于共模攻击。那么只要利用扩展欧几里得算法求出s,t使得s * e1+t * e2 = 1即可。

```
#共模攻击
import libnum
e1 = 2767
c1 = 20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086
88796904072195953319005834276205735943266371782582636544499691546903905642841616617392095824304483140492411344251
26175994268761411842121216775003712369371275718028913217065876103936394468683698717030181301821840888696826388
21230841556074940763302569342851713707585865354151361628611388987289105851383788845308198574786097911269713086241
31845490599291940535575149278911000931313841726512611727371081384392314338127620480251591052746888322427482996241
79636527422350190210717694762908096944600267033351813929448599
e2 = 3659
c2 = 11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081
54502705652379002259823391801126101197319638639568937152677478558232612195918619558606985159246763781936662404411
33661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363481
29587083647269829087983401822887021010233938397814273865372304594365126130473115858750680082108189969414601565891
31413501043836244752242820688494495263982667724781906681270683577310705956708282231230072104982701366041861026511
89288840247186598145741724084351633508492707755206886202876227
n = 21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460
26667269256158268953033778016148293640346244751958599979431463055883159391307774504851962907662496123400543546225
16207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620
73070673410361183317973326409647528649198806399043108538049907500562980770240667670784132466097117325310095636252
83466847529599374738526301458937960566757936464307935782654182559193763237960445885597267038584293117847052450
69845938316802681575653653770883615525735690306674635167111

def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b) # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)
        return x, y, q
r,s,q = ext_euclid(e1,e2)
m = (pow(c1,r,n)*pow(c2,s,n))%n
print(libnum.n2s(m))
```

3.SameMod

可以看出来是共模攻击，不过最后用libnum.n2s等方法解出来的不对。后面尝试了将m的前几个数根据情况分开，进行chr()转换，出现了flag。于是就想着有没有什么函数可以一次性将这串数字转成字符，找了老半天了，后面才知道要自己手动完成！！！

```
import libnum

e1 = 773
c1 = 345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398111315531115373349
e2 = 839
c2 = 567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323830953 5
n = 626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9

def ext_euclid(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = ext_euclid(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, q
r,s,q = ext_euclid(e1,e2)   #r=gmpy2.gcdext(e1,e2)[1],s=gmpy2.gcdext(e1,e2)[2]
m = (pow(c1,r,n)*pow(c2,s,n))%n
print(m)
print(libnum.n2s(m))#b'\x86\x8e~\xd9\x1f\x06\xe6\x9f\xd2\x9b\xd3\xa2j\xa8\xaf4"\xc9\x90\xe2\x81TI>J\xf2\x89\xa9^>\x93\xd4P\xba\x05'

ming = str(m)
i = 0
while i < len(ming):
    if ming[i] == '1' :
        print(chr(int(ming[i:i+3])),end = '')
        i += 3
    else :
        print(chr(int(ming[i:i+2])),end = '')
        i += 2
```

4.[BJDCTF2020]easyrsa

这个n可以用网站进行分解就得到了。不过按照题目的意思来吧。通过求导并化简得$z = p^2 + q^2$

已知$n = p*q$，可以求出$p+q$，再构造方程$x^2-(p+q) * x +p * q = 0$，分别求出p,q。

```
'''
    #分数      #导数     #arctan()                          #反双曲正切arth()
z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
#z = 1/(1/(1+p**2)) - 1/(1/(1-q**2)) = p**2+q**2
'''

import gmpy2,libnum
c = 792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380
351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957 2
389676084392706406084307657000104665696654409155006313203957292885743791715198781974205578654792123191584957665 2
932083904537483691823331528098823124533597061478081989229167627737217266815889771038774541190437448891645293831 8
807749919493290964391869664687690732736475138095318251788313459181080084897171918480871369434298545810300667601 3
451912221080252735948993692674899399826084848622145815461035
z = 321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764
345875193333001424730103446948038851685575488012024959332262154377633292802421135565244984575595628729008116020 5
694442396740377762330696188075761324632872961664303262896407293127208586692804597379937471184682515778105696516 4
178505232524245809179235607571567174228822561697888645968559343608375331988097157145264357626738141646556353500 9
949241158757481983180362968986040970009382721959030567335658801505402753692396377939759233295987160033503082593 2
143675257929100035556043154222969975995514115291470836249448 2
n = 153107451613368954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012
490941618672076865376070474479687087589909501363809247473590525705495940985699706328543518259507297525635022848 4
926373012758638252270395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257 3
405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869 3
439391065292047982859575168607743840018927775259161677432724199585720553322320560959794481550824659777814825983 7
199479887191751476750839473044797477032996768176762549539444 1
#解方程 x*x-(p+q)*x+p*q=0
paq = int(gmpy2.iroot(z+2*n,2)[0])
pd = int(gmpy2.iroot(paq**2-4*n,2)[0])
p = (paq + pd)//2
q = (paq - pd)//2
e = 65537
phi = (p-1)*(q-1)
d = int(gmpy2.invert(e,phi))
m = pow(c,d,n)
print(libnum.n2s(m))
```

5.[NCTF2019]babyRSA

因为phi = (p-1) * (q-1)，又q是p的后一个素数，所以p^2 < phi < q^2,即sqrt(phi)永远在p和q之间，求前后素数即可得到p和q。
已知e * d = 1 mod (phi) ==> ed - 1 = t * phi
因为(e*d-1)是2064bit，p和q是素数，(p-1)和(q-1)是1024bit，乘起来是2048 ~ 2049bit，那么t是15 ~ 16bit。枚举t并验算p,q。

```
'''文件内容
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) #返回下一个素数
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)
'''

import gmpy2,libnum,sympy,binascii
from Crypto.Util.number import long_to_bytes
d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218043
e = 65537
pd = e*d-1
#print(len(bin(pd)[2:])) # 2064bit  p*q至少2048bit  t最多16bit
start = pow(2,15)
end = pow(2,16)
for t in range(start,end):
    if pd%t == 0:
        phi = pd//t
        p = sympy.prevprime(int(gmpy2.iroot(phi,2)[0])) #phi近似于p**2,#prevprime返回上一个素数
        q = sympy.nextprime(p)
        #这里的p,q并无大小之分，可以任取前后素数，但必须是一前一后
        if (p-1)*(q-1) == phi:
            break
n = p*q
m = pow(c,d,n)
print(long_to_bytes(m))#print(binascii.unhexlify(hex(m)[2:]))
```

6.[ACTF新生赛2020]crypto-rsa3

已知n,e,c，利用fatordb网站分解n，得到p,q。

```
'''文件内容
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)
'''
import libnum,gmpy2
n = 17760650483649924697095903022687160888596932177821105108052463408451697333144164499389802957361229009585306926403653045925365287558626794687783105514754691022710056649665814838183468303736613455384801190325125272647404766127422313772768868953582353304677879313190214344440873561082116783871748885990224286368 3
c = 14573903785113823547710005409453611689847750526930736416823750714074908512897030709057495258304830359887371176539714284246123320209259266173955588681603806019124982999228259142295101669579104518417300289198838076344898341288308014072284472217752647113499281562901027823743794067192921160475815605303822100 49
e = 65537
p = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179231
q = 13326909050357447643526585836833969378078147057723054701432842192988717649385731430095055622303549577233495793715580004801634268505725255565021519817179293
d = int(gmpy2.invert(e,(p-1)*(q-1)))
m = pow(c,d,n)
print(libnum.n2s(m))
```

7.[AFCTF2018]你能看出这是什么加密么

解出的结果是一堆'\x'的东西就懒得看后面的了，看到这个我以为转换的方法不对，用了其他方法，还是这样。可恶，到最后网络上一搜发现flag就在最后面。不过当时写的一些题目解出来是'\x'的东西，网上查了一些，不知道为什么会产生这种结果，而且有的貌似还没找到解决方法，曾一度崩溃，所以看到这个的时候真没有心情往下看。

```
import libnum,gmpy2
from Crypto.Util.number import long_to_bytes
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
p = int('0x928fb6aa9d813b6c3270131818a7c54edb18e3806942b88670106c1821e0326364194a8c49392849432b37632f0abe3f3c52e909b939c91c50e41a7b8cd00c67d6743b4f',16)
q = int('0xec301417ccdffa679a8dcc4027dd0d75baf9d441625ed8930472165717f4732884c33f25d4ee6a6c9ae6c44aedad039b0b72cf42cab7f80d32b74061',16)
e = int('0x10001',16)
c = int('0x70c9133e1647e95c3cb99bd998a9028b5bf492929725a9e8e6d2e277fa0f37205580b196e5f121a2e83bc80a8204c99f5036a07c8cf6f96c420369b4161d2654a7eccbdaf583204b645e137b3bd15c5ce865298416fd5831cba0d947113ed5be5426b708b89451934d11f9aed9085b48b729449e461ff0863552149b965e22b6',16)
n = p * q
phi = (p-1)*(q-1)
d = int(gmpy2.invert(e,phi))
m = pow(c,d,n)
print(m)
print(libnum.n2s(m))
```

8.[RoarCTF2019]babyRSA

明知道不行，我还是去算 B！。网上别人用Wilson定理来解。看到Wilson定理，这才恍然大悟。

- Wilson定理
  设p是一个素数，则(p-1)! = -1 (mod p)

因为A是素数，所以(A-1)! = -1(mod A)，又因为B<A，所以有B！ = -1 * $(A-1)^{-1}$ * $(A-2)^{-1}$ * … * $(B+1)^{-1}$ (mod A)，至此(B!)%A 就解决了，从而能求出p和q。
不过这里要注意的是n = p * q * r，所以phi = (p-1) * (q-1) * (r-1)。为什么n = p * q * r ？为什么不是 n = p * q?

```
'''
import sympy
import random

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

'''

import gmpy2,libnum
import sympy
#p=myGetPrime()
A1=218569634524616304373482784341914340000660767504190274938524635134698652620643408366138310666023009597726323977734873175603390566582999544641692644672344078
B1=218569634524616304373482784341914340000660767504190274938524635134698652620643408366138310666023009597726323977734873175603390566582999544641692644671405960
#q=myGetPrime()
A2=164661131158392281197678878993088200257492609338634468882241671698576121786641395457263408674067907545602275160137962699414380768181946170303048518584189270
B2=164661131158392281197678878993088200257492609338634468882241671698576121786641395457263408674067907545602275160137962699414380768181946170303048518583510260

#r=myGetPrime()

#Wilson定理  p为素数,(p-1)! = -1 mod p
def getpq(B,A):
    result = -1
    for i in range(B+1,A):
        pd = int(gmpy2.invert(i,A))
        result = (result*pd)%A
    return result

#n=p*q*r
n = 8549266378627529215983160339108387617514935430932767300871662765071816058563972310079334753464962833041663125566090130753390990043141344752426233223265915304706790869348194712106907045156282241735765643217187095118467313255421369012330804269736196998636037506095470292065636414415414581283855836533417293593144142409627020614069181466231856269692576799193736978262790840823908735803316541002069015206771571111273225203858843289675840589870901034246788226436273 3
#c=pow(flag,e,n)
e = int('0x1001',16)
c = 75700883021669577739329316795450706204502635802310731477156998834710820770245219468703245302009998932067080383977560299708060476222089630209972629755965140317526034680452483360917378812244365884527186056341888615564335560765053550155758362271622330017433403027261127561225585912484777829588501213961110690451987625502701331485141639684356427316905122995759825241133872734362716041819819948645662803292418802204430874521342108413623635150475963121220095236776428
p = sympy.nextprime(getpq(B1,A1))
q = sympy.nextprime(getpq(B2,A2))
r = n//p//q
phi = (p-1)*(q-1)*(r-1)
d = int(gmpy2.invert(e,phi))
m = pow(c,d,n)
print(libnum.n2s(m))
```

9.[RoarCTF2019]RSA

用factordb网站分解n，得到p和q。e通过爆破来求得。因为e模phi不可能都有逆元，所以用try except来处理异常。

```
import gmpy2,libnum,sympy,binascii
from Crypto.Util.number import long_to_bytes,bytes_to_long
#A = (((y%x)**5)%(x%y))**2019+y**316+(y+1)/x
#p = gmpy2.nextprime(z*x*y)
#q = gmpy2.nextprime(z)
# p > q

A = 26833491826787145242474695127934760098610147810049249054841274803081613777681928680615618865770486464323821289608814874634274141761144868858306939594049897432291035169244325127241956544257034536127103105871644170358783083906766125928487502873873181294241952086234402946478173677408782119491475262870912983074805028974622791025725568222316694382793174748284790897190463864119711054487239105947104180939770441799498003732243547291798333932198277893890788692902175695112308689676479630894305942588151463621872508551668975530560737445829461484720683341674454993144715183575352611863187563278900161832284122537243100n = 11793080604350737432598229182302728514880723911798736960958351535388981485608809967145439434081676124297446226843591176504557637776771159310041693201983188905933316694626318486128797572295499221976649308963081087698478111364536245039800923455608533094312556837774106524218307388255883460343086259806678647529991839534101487741690118539290567604379542512696874518564956510632233695442750510490677049315572399538231834671494418457789415022903775843459724256481529917495014775442695025141920491737651736050502454969172368335817082341675797305935478414260143651950081115903679503467636002892830197978052829411493334712 7c = 41971850275428383625653350824107291609587853887037624239544762751558838294718672159979929266922528917912189124713273673948051464226519605803745171340724343705832198554680196798623263806617998072496026019940476324971696928551159371970207365741517064295956376809297272541800647747885170905737868568000101029143923792003486793278197051326716680212726111099439262589341050943913401067673851885114314709706016622157285023272496793595281054074260451116213815934843317894898883215362289599366101018081513215120728297131352439066930452281829446586562062242527329672575620261776042653626411730955819001674118193293313612128

#factordb
p = 84286804568139093453973995920184755228498017995887966793307845395096856615166214726700629357176546313727059415113869577898616511138042880654559358807836533131308423001461871441295958484342158667416268832194288936991239203188262099494424198715307815638947037019551428585073654107862385432795938215675345856 9q = 13991609558311089513359683322750669367930670987317402487689102335586078198117591644632304473291306688078691862908902349931170340848915118188656853562100864499797198218242670659255129108400798338791100626144251963540545707729251508516074416986741097396065208145245537145122226581905155981844125743802107394118 3phi =(p-1)*(q-1)
e = 2
while e < 100000:
    try:
        d = int(gmpy2.invert(e,phi))
        m = pow(c,d,n)
        m = str(long_to_bytes(m))
        if 'CTF' in m:
            print(m)
    except:
        pass
    e = sympy.nextprime(e)
```

10.[HDCTF2019]together

公钥解析求出两个加密的e和n，发现n相同，是共模攻击。另外两个文件里面是base64编码，应该是密文，只通过base64解码只能得到一串'\x'的东西，需要进行bytes_to_long()转换成数字，毕竟写过的题目里面的c都是大数，所以把它转成数字，

```python
import libnum,gmpy2,base64
from Crypto.Util.number import long_to_bytes,bytes_to_long
e1 = 2333
e2 = 23333
n = 14853081277902411240991719582265437298941606850989432655928075747449227799832389574251190347654658701773951599098366248661597113015221566041305501996451638624389417055956926238595947885740084994809382932733556986107653499144588614105694518150594105711438983069306254763078820574239989253573144558449346681620784979079971559976102366527270867527423001083169127402157598183442923364480383742653117285643026319914244072975557200353546060352744263637867557162046429886176035616570590229646013789737629785488326501654202429466891022723268768841320111152381619260637023031430545168618446134188815113100443559425057634959299
c1 = base64.b64decode('R3Noy6r3WLItytAmb4FmHEygoilucEEZbO9ZYXx5JN03HNpBLDx7fXd2fl+UL5+11RCs/y0qlTGURWWDtG66eNLzGwNpAKiVj6I7RtUJl2Pcm3NvFeAFwI9UsVREyh7zIV6sI9ZP8l/2GVDorLAz5ULW+f0OINGhJmZm8FL/aDnlfTElhQ87LPicWpXYoMtyr6WrxjK6Ontn8BqCt0EjQ7TeXZhxIH9VTPWjDmFdmOqaqdVIT+LZemTgLNESwM5nn4g5S3aFDFwj1YiDYl0/+8etvKfOrfoKOwR0CxsRHagwdUUTES8EcHLmMGCxCkDZn3SzmmA6Nb3lgLeSgG8P1A==')
c2 = base64.b64decode('O+rRCXI3aTB6P1rYIOPUdalUp6ujpwEq4I20CoWA+HIL8xxGtqY6N5gpr0guZv9ZgOEAMFnBxOqMdVNnB9GgnhmXtt1ZWydPqIcHvlfwpd/Lyd0XSjXnjaz3P3vOQvR71cD/uXyBA0XPzmnTIMgEhuGJVFm8min0L/2qI7wg/Z7w1+4mOmi655JIXeCiG23ukDv6l9bZuqfGvWCa1KKXWDP31nLbp0ZN2obUs6jEAa1qVTaX6M4My+sks+0VvHATrAUuCrmMwVEivqIJ/nS6ymGVERN6Ohnzyr168knEBKOVj0FAOx3YLfppMM+XbOGHeqdKJRLpMvqFXDMGQInT3w==')
c1,c2 = bytes_to_long(c1),bytes_to_long(c2)
gcd = gmpy2.gcdext(e1,e2)
s,t = int(gcd[1]),int(gcd[2])
m = pow(c1,s,n)*pow(c2,t,n)%n
print(libnum.n2s(m))
```