BUU-RSA入门题合集 第二弹

原创

拔草能手晓寒 于 2021-07-15 09:53:31 发布 201 收藏 1

分类专栏： BUUCTF RSA专栏 文章标签： rsa 密码学 python

BUUCTF-RSA签到题第二弹，有意思或者有难度的RSA题目单独放在专栏里了BUUCTF RSA专栏_晓寒的博客-CSDN博客

文章目录

Dangerous RSA(低加密指数攻击)

题目

> n=0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793L
> e=0x3
> c=0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365
> so,how to get the message?

解题思路

题目只给出了n，e，c，且模数n很大，难以分解

但是我们注意到公钥e = 3，这就给加密带来了很大的不安全性

利用这一点可以针对小加密指数进行攻击：

- 若$m^e < n$，则m可以由c直接开3次方得到
- 若$m^e > n$，令$m^e = c + kn$，则m可以由(c+kn)开3次方得到，这里对k进行遍历

附上代码：

```
import gmpy2
import binascii

n=0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793
e=0x3
c=0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365

k = 0
while(True):
    #gmpy2.iroot(x,n) x开n次方根
    #返回值m为开方计算结果，f为bool变量，标识x能否被开方
    m, f = gmpy2.iroot(c+k*n, e)
    if f:
        m = hex(m)[2:]
        print("明文数据为：0x" + m)
        flag = binascii.unhexlify(m)
        print(flag) #flag{25df8caf006ee5db94d48144c33b2c3b}
        break
    k += 1
```

此题中，直接开方就得到了结果

flag

> flag{25df8caf006ee5db94d48144c33b2c3b}

[HDCTF2019]basic rsa(基础题)

题目

> 给我们的是一个代码文件attachment.py，内容如下

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

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

p = 26224880018227704065019205543990658047 9
q = 26285499423932282854792559548751991555 1

e = 65533
n = p*q


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

print c

# 27565231154623519221597938803435789010285480123476977081867877272451638645710
```

解题思路

先运行一下试试

这里有点坑，因为我用的python3，需要修改一些地方才能解决报错问题，修改如下：

```
c = pow(int(b2a_hex(bytes(flag, encoding = "utf8")),16),e,n)
print(c)
```

1. b2a_hex()接受的参数必须为bytes类型，这里进行了强制类型转换
2. print函数在python3中要加括号

分析一下代码：

b2a_hex()是将字节类型数据转换为十六进制数据，然后用int()又转换成十进制

最后用pow()函数进行模幂运算，c = $flag^{(e)}$ % n，也就是说把flag用RSA加密了

加密输出结果如下：

> 954455212242600299696284381044184839703678406319148778406581776490899851981 9

嗯，完全不知道这波操作在干什么，可能在教大家如何使用RSA加密文本？（斜眼笑.jpg）

那么真正的flag在哪呢

注意到代码最后一行注释有一串数据，尝试对其解密

附上代码：

```python
import gmpy2
import binascii

p = 26224880018227704065019205543990658047 9
q = 26285499423932282854792559548751991555 1
e = 65533
n = p*q
c = 27565231154623519221597938803435789010285480123476977081867877272451638645710

d = gmpy2.invert(e,(p-1)*(q-1)) # 求逆元，de = 1 mod fai(n)
m = gmpy2.powmod(c,d,n)# 幂取模，求明文

flag = binascii.unhexlify(hex(m)[2:])
print(flag) #flag{B4by_Rs4}
```

get flag! 这行注释才是密文本文

flag

> flag{B4by_Rs4}

BabyRsa(基础题)

题目

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

解题思路

这道题直接给出了私钥d，但是没给模数n，因此要从p+q和(p+1)(q+1)中求出n = pq

非常简单就能得到：

> (p+1)(q+1) = pq + p + q + 1
> pq = (p+1)(q+1) - (p+q) - 1

得到n之后正常解密就能拿到flag

附上代码：

```
import gmpy2
import  binascii

#p+q = 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea
#(p+1)(q+1) = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740
e = 0xe6b1bee47bd63f615c7d0a43c529d219
d = 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5
enc_flag = 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a
a = 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea
b = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740
n = b - a - 1   #计算模数n

m = gmpy2.powmod(enc_flag, d, n)    #解密

print("明文数据为：" + hex(m))
flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```

flag

> flag{cc7490e-78ab-11e9-b422-8ba97e5da1fd}

RSA5(多N分解)

题目

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-C6X6udkC-1626313465179)(C:\Users\xiao_han\AppData\Roaming\Typora\typora-user-images\image-20210709092753224.png)]

给出了加密指数e，以及不同模数n对应的加密结果c，一共给出20组nc

解题思路

由于这道题中e不是小指数，因此不能像RSA4一样用中国剩余定理求解

因为我们手上一共有20个模数n，尝试一下是否会有某两个n的大素数因子相同

因为p，q都为素数，所以n的因子只有{1, p, q, n}

直接计算两个n的公约数，若大于1则必定是p或者q中的一个

有了p或者q中的一个，就可以分解出n = p * q，从而计算出私钥d进行解密

附上代码：

```
import gmpy2
import  binascii
```

```python
def solve(nList, cList):
    for i in range(len(nList)):
        for j in range(i+1, len(nList)):
            p = gmpy2.gcd(nList[i], nList[j])   #计算最大公约数
            if p != 1:  #若存在最大公约数，则该数为n[i]和n[j]共有的大素数因子，即p或者q
                print("n[%d]和n[%d]存在公约数为\n%d" % (i,j,p))
                q = nList[i] // p
                print("分解n[%d]得到:\n%d =\n%d * \n%d" % (i,nList[i],p,q))
                d = gmpy2.invert(e,(p-1)*(q-1)) #计算e的逆元，即私钥d
                m = gmpy2.powmod(cList[i], d, nList[i]) #模幂运算，解密
                return (binascii.unhexlify(hex(m)[2:]))

if __name__ == "__main__":
    e = 65537
    #========== n c ==========
    n1 = 20474918894051778533305262345601880928088284471121823754049725354072477155873778848055073843345820697886641086842612486541250183965966001591342031562953561793332341641334302847996108417466360688139866505179689516589305636902137210185624650854906780037204412206309949199080005576922775773722438863762117750429327585792093447423980002401200613302943834212820909269713876683465817369158585822294675056978970612202885426436071950214538262921077409076160417436699836138801162621314845608796870206834704116707763169847387223307828908570944984416973019427529790029089766264949078038669523465243837675263858062854739083634207
    c1 = 97446390824333086572897876921359540078205339859689774131627572259641501891292950863739385091922496927176638871002519503989696195606289557006214694773634034292797499261667889337274426195417287349087880548324119634588172116407865115606711995781642276852444202568807946265675560598210417400163534587402213304540234401004596111172015199041203447775585180276906930906901873854185413018369220475876142712127998200299393974534369567190001529679063746488033737551153642479689099652668120063308684103632039584772593574475799301335280465057506813612929559130656921330015633365091079594680082006749414336488584289629112613732 0
    n2 = 20918819960648891349438263046954902210959146407860980742165930253781318759285692492511475263234242002509419079545644051755251311392635763412553499744506421566074721268822337321637265942226790343839856182100575539845358874937183342375858212633881811265451897234292621496306512894465534021905311355208361042171602683496885251683752134625702136128458989896943242694102024968716886499783702846610173990569039318406567573308596261837733965740564130173676064465401999731556304662394536372329369040637065511606502950312733856194707405935102672859579058015663625022627577506291629373737212917895276595314994352352616203097590 1
    c2 = 15819636201971185538694880505120469332582151856714070824521803121848292387556864177196229718923770810072104155432038682511434979353089791861087415144087855679134383396897817458726543883093567600325204596156649305930352575274039425470836355002691145864435755333821133969266951545158052745938252574301327696822347115053614052423028835532509220641378760800693351542633860702225772638930501021571415907348128269681224178300248272689705308911282208685459668200507057183420662959113956077584781737983254788703048275698921427029884282557468334399677849962342196140864403989162117738206246183665814938783122909930082802031855 2
    n3 = 25033254625906757272369609119214202033162128625171246436639570615263949157363273213121556825878737923265290579551873824374870957467163989542063489416636713654642486717219231225074115269684119428086352535471683359486248203644461465935500517901513233739152882943010177276545128308412934555830087776128355125932914846459470221102007666912211992310538890654396487111705385730502843589727289829692152177134753098649781412247065660637826282055169991824099110916576856188876975621376606634258927784025787142263367152947108720757222446686415627479703666031871635656314282727051189190889008763055811680040315277078928068816491 3
    c3 = 41853085294168740058312307810140924071984513859556773996685018339026234783956692794048839907251843327091524433725837010761987866352917393567708572867021071567300200043589556225110614106610589826220551997368208082038414467963052843946517144309186903894869205608346723161581464531837894121409390290293247560353580817544266451600332629243302486752161082709801570497054886202634851294809528147640028652800191851276624493183242793832777664162581422751439235321687984130110282715430852490290489974522125031117423023020650410514580665853953604684474606586729528516435471938227752183878536234536380254923891222045075555908862 4
    n4 = 21206968097314131007183427944486801953583151151443627943113736996776787181111063957960698092696800555044199156765677935373149598221184792286812213294617749834607696302116136745662816658117055427803315230042700695125718401646810484873064775005221089174056824724922160855810527236751389605017579545235876864998419873065217294820244730785120525126565815560229001887622837549118168081685183371092395128598125004730268910276024806808565802081366898904032509920453785997056150497645234925528883879419642189109649009132381586673390027614766605038951015853086721168018787523459264932165046816881682774229243688581614306480751 5
    c4 = 45210380110447584418911284684672330884938857508505889857085199111547780905971361261502890418934541266744681413934726623373503617122126948673116229704407077279411132638323571731417758552279737425710889745934763020841117706257642228383662775595608870429488598921385514726806545178149166092797483655806107122598566777405184770865 5
```

```
15922331071754700682919036075057994329319896637074770179046114262137702383970057437303860800319556941584665584755997519402450391676291265767840244823484528683134174715429567782855677794359402671406799066865318624676272384010034591016371912972094224703881218025365697614144576182583435506 13
    n5 = 22822039733049388110936778173014765663663303811791283234361230649775805923902173438553927805407463106104699773994158375704033093471761387799852168337898526980521753614307899669015931387819927421875316304591521901592823814417756447695701045846773508629371397013053684553042185725059996791532391626429712416994990889693732805181947970071429309599614973772736556299404246424791660679253884940021728846906344198854779191951739719342908761330661910477119933428550774242910420952496929605686154799487839923424336353747442153571678064520763149793294360787821751703543288696726923909670396821551053048035619499706391118145067 
    c5 = 15406498580761780108625891878008526815145372096234083936681442225155097299264808624358826686906535594853622687379268969468433072388149786607395396424104318820879443743112358706546753935215756078345959375299650718555759698887852318017597503074317356745122514481807843745626429797861463012940172797612589031686718185390345389295851075279278516147076602270178540690147808314172798987497259330037810328523464851895621851859027823681655934104713689539848047163088666896473665500158179046196538210778897730209572708430067658411755959866033531700460551556380993982706171848970460224304996455600503982223448904878212849412357 
    n6 = 21574139855341432908474064784318462018475296809327285532337706940126942575349507668289214078026102682252713757703081553093108823214063791518482289846780197329821139507974763780260290309600884920811959842925540583967085670848765317877441480914852329276375776405689784571404635852204097622600656222714808541872252335877037561388406257181715278766652824786376262249274960467193961956690974853679795249158751078422296580367506219719738762159965958877806187461070689071290948181949561254144310776943334859775121650186245846031720507944987838489723127897223416802436021278671237227993686791944711422345000479751187704426369 
    c6 = 20366856150710305124583065375297661819795242238376485264951185336996083744604593418983336285185491197426018595031444652123288461491879021096028203694136683203441692987069563513026001861435722117985559909692670907347563594578265880806540396777223906955491026286843168637367593400342814725694366078337030937104035993569672959361347287894143027186846856772983058328919716702982222142848848117768499996617588305301483085428547267337070998767412540225911508196842253134355901263861121500650240296746702967594224401650220168780537141654489215019142122284308116284129004257364769474080721001708734051264841350424152506027932 
    n7 = 25360227412666612490102161131174584819240931803196448481224305250583841439581008528535930814167338381983764991296575637231916547647970573758269411168219302370541684789125112505021148506809643081950237623703181025696585998044695691322012183660424636496897073045557400768745943787342548267386564625462143150176113656264450210023925571945961405709276631990731602198104287528528055650050486159837612279600415259486306154947514005408907590083747758953115486124865486720633820559135063440942528031402951958557630833503775112010715604278114325528993771081233535247118481765852273252404963430792898948219539473312462979849137 
    c7 = 19892772524651452341027595619482734356243435671592398172680379981502759695784087900669089919987705675899945658648623800090272599154590123082189645021800958076861518397325439521139995652026377132368232502108620033400051346127757698623886142621793423225749240286511666556091787851683978017506983310073524398287279737680091787333547538239920607761080988243639547570818363788673249582783015475682109984715293163137324439862838574460108793714172603672477766831356411304446881998674779501188163600664488032943639694828698984739492200699684462748922883550002652913518229322945040819064133350314536378694523704793396169065179 
    n8 = 22726855244632356029159691753451822163331519237547639938779517751496498713174588935566576167329576494790219360727877166074136496129927296296996970048082870488804456564986667129388136556137013346228118981936899510687589585286517151323048293150257036847475424044378109168179412287889340596394755257704938006162677656581509375471102546261355748251869048003600520034656264521931808651038524134185732929570384705918563982065684145766427962502261522481994191989820110575981906998431553107525542001187655703534683231777988419268338249547641335718393312295800044734534761692799403469497954062897856299031257454735945867491191 
    c8 = 60401197951758564075410823600235322046147238586886367248227127175727597939602463418003081497398098712343130496297329347975697810530006861856663748339784032905250725987740017313502447445907727957010651295618981165764999841859206612711236653561327191936654742355968842391080306058827778688561223782226811405705191803212869769471540422726224113039810113025862256308598927317246405746581254782871151984062538473679798837680008126053954829526986896044777194789475954421859214806526378683356732332006621006210250615008957296053056658646931229525573618715231653002060703256603530955927780377673953603292313313228236100600 06
    n9 = 23297333791443053297363000786835336095252290818461950054542658327484507406594632785712767459958917943095522594228205423428207345128899745800927319147257669773812669542782839237744305180098276578841929496345963997512244219376701787616046235397139381894837435562662591060768476997333538748065294033141610502252325292801816812268934171361934399951548627267791401089703937389012586581080223313060159456238857080740699528666411303029934807011214953984169785844714159627792016926490955282697877141614638806397689306795328344778478692084754216753425842557818899467945102646776342655167655384224860504086083147841252232760941 
    c9 = 54181203012083787131158894655799642578718141145150460960909601597378590768292585169203615778539039259541984068437573036875578483023022002292959169024302057378436018067007382347566985757086124249284804408687391200758
```

```
886816720622065291565664212766111078029174189936250296906271968138303263698742497776192396033006058768659675157190797971159105786535627878990193101399459049580248824178337363048947654334894762345753567552751472565773870228733489069001496349404747104513850154118106991137072643308620284663108283052245750945228995387803432128842152251549292698947407663643895853432650029352092018372834457054271102816934
    n10 = 288736679047156827229872342934932003069769478987112550641251159336669686787425988587224314262189144629035215963417711316956193822661942335616778243573798053038859938042664368106062630220979002669752504315756546869150496930914678648205127670707132677089938998990111561067661789067003361117128033621130396135486729370533978756631447940180870177319490877948949037376823839161732674214034081409677130710260018747334872950075010688710446491706157098914518567922323155266962201618427426647785812873213187482024314665089489027453143722997995616251869552346730120982109197458798822685126569317143267823352110895768973105914910
    c10 = 991988046378683668498795797909152747747144499639237524407552784186550916018166654301631763496351243751032419870241632284137748941702957238847445007580146299682524465753028610742818635417283671650281760907059092976926193232427535328993930253644031062869834924487206400570064452022372767095078792429600429688303297894120088336265399335163854586020717902247249267125663042722846185266811803531702142867595487494701519774591691819772512112223636938274153398302346225591392469280624938744901662986582331640236601765784416691984668349785184238805828385621990053556742710360386995506619342550138525532209790153140210388386910
    n11 = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947873124455281596355278339044758018903814556538072655012173287578713527312930003034382053158167926639175790666748423077438452617710323639285688446698957680925156583287562292458370252617442606148607469979315035487885099838680383497202253057309855762936752690737090223507008365100540676417537132129999543070225244958855833617073785137421625663390101343549078637332059218450389182244639037898418814008140745872617202838797601220709014665171182654228634203769215367348455021002514608724991222366868321895496980207371766830 19
    c11 = 149152705020329498988282924856039518480497727774712614310395721916462418752844104783735126358044068647476738046400554026462791012648312993066834409581454759211506105784347013149807506042039511100861902719903701992570123666016656306824568397578776280435952016470169169091648259102613858270555824686949616275978087843713796082300004398822730300387641050312137016330371160335943076453933759786686250845152815828510325181005874187968787521838416028250617270661335947765721542073481604939333959375548921858879660706026189790523345326867141161063104734045948793747951193345036946221379573893301900147180315760779173853846 7
    n12 = 276467464237590201110078286532640279992578476456661299077890260545943936488002361170467691127626417886562089244342310018961932758581138488351542491875274955962755363778503735963980112521325616300843194259372793193189819972755276862677561847983302910124969257371603070669570251098228355574085104702267248574343246464777288231421517611473225749724028416401691401868904455721892030026223465284063240606727337526930100840986019318082236673587728820578331432610226375650378673612232134832003195001214490586955620401743059365605286793949363316349958024222476340433880702251013621718777908491799617160273706564991036724299
    c12 = 219915241289572605360437712848549203931058081267001282221258567755068857219711931093613159611291908146746471364648870878939906608949616128382050864010188854576674889118986542702355619801111746033237212809111974882865852693568495792630434563163194764958886962193442198665168611876541805092478812512512789193462671299047392773862892403943845751243311356559435138310099340233974570821846997377343888237633068053264303958499357702138175333872354863070088924109206116699326930181655694174458858108257496093886272312358409126446546858196209316633462975963348344986617890164503717692036501099947718724041857702301729340139711
    n13 = 205454874058169287317389883744750126868279337097897843918557068351362702709334012030193291369376508783861171877765306393425721232718805397862269728252147391797828283043216115322121619416987966954199884069138302548722085087207543606430849992495851797972795440296561219608140434165151732636404151925012503642482263435426877389546569892088343922299658122635859587399397660469983061393232072055413001167129794443351504718056548449519100388759989128903798201021635783107832815902895322205691818936584071158867109333301311745403431362285508279581312233856244622304121119227708922507832468210803384302390355017289195967355 1
    c13 = 142274391881910294612504766927905396546191998884873194291144145579753763086889080281408171572055798040597838076413055773857247585301385149729622090622305761074061424026034843756260773451908830940976360197713778663395315119651366505674123638891831596161884492637524753286632453110599883379960473592632888374363055888480445729377594244665868702805124243368070647298945158405524047568795906987970463333364454651204450875876217439066242796217796347723788029591097144005161837183232672738247365401685459464444375862992141104247381599573883507859993485351715535693730882515527123912883652952676656913577196160116136287721751
    n14 = 273597277115842772348971577240558527940192168452297989386558142694600463843535681385985677553925596534609494445578791200407967981422189392518447624612702516723995467740672753482910039625519646487420532154246202569993454483988052785927770496682815583128717739799313430978068787011140560300415066904769542540065925552753425795296252311943213579046685121215395148807040469699748984120956750825853154582675910167349246462943576669242939084183455089021127110752320479987753036031753639640550485897693185621048836597549749555617256947797542796067263585888624791988159992768392349521420172105938873719506454184173559125679871
    c14 = 378852978424825502708167454087701637280784822277688792045348887824713793057829679743764792249451048376765115049293335609328896594374157026894386198702427661071271740913994640951396304311446393314608843000423774716 3
```

```
4228029592502966025706493630161515813640067958942265995847080725826969967405188876067854607758510298142803593857
6309107890230195722648462042851360463058513151116701576319059122588420277284045656364315950780571100411390141750
3751181050823638207803533111429510911616160851391754754434764819568054850823810901159821297849790005646102129354
0357353501244768387866615420890455096569103486767428449570088574577
    n15 = 2754593760375173724878522089173579646897332973807620914407992144996729257234942453901050228756403011 68
3126126819738465051104306873891142916973064013594780088598717153926721461190768757058700193382920865510082804565
1391618089603288456570334500533178695238407684702251252671579371018651675054368606282524673369983034682330578308
7698864563358187338272372945704768536735526853616891442615528957582665223930041160178493973462591192210638216632
8093582044067182560145241748733010528088952000791797911556806716159005827741837149322863123245797249428501476746
989364789288868143396585749691611070494475807026862689704501478 2837
    c15 = 1406911297060889573241703997754273266579660189376240150087878687168064579875478331569351126174005972 51
7134240418657106697254633281366771113566117665942461993610103890343914429488637932259163576668264517988805861757
7572409307484708171144488708410543462972008179994594087473935638026612679389759756811490524127195628741262871304
4279084812149924711828593088287781190057509289357649279672123435265034105157937172013603604379813225767980562766
5714036333270071473222484834680896399230240903770609458896417023952119358947007083979040459725299081858371786914
022981171229500571054047635674337890664226704572363387401164925 9842
    n16 = 2574616207569791156026318179121643306257417857242460033685627817611273305443146325390343312823270905 41
4160710089117780428581378324773506375340652467803056128449148122168195456480414145466692865754967026677565986281
4924386584148785453647316864935942772919140563506305666207816897601862713092809234429096584753263707828899780979
2231181810092936555631465267923889134625573064336642969663314699064286651274388293997030028678002699478558692620
3671425655007552019312598701194519227353173227664172800840685587159867893658532478243866874681051666015201824425
300809247006655568727713893729874795192957623103625131627060251 3451
    c16 = 1734428486027548947749152581992285532679227512871970940129254560812285982982746208839004461223496755 16
8287995430145842584283199551383241035532806556209876366032616326203320034733877343909570994420225249455217258950
3915965931524326523663289777583152664722241920800537867331030623906674081852296232306336271542832728410803631170
2296427175249423323908424670351436315044011407270832707324642374439152638658805803087761112197189617463788429246
4414212724357382497253381947907938102310358586209906338212975756012407467615062228870609411007556770640344292069
64726277976076979628730261122405274983085359032326639390285870 36724
    n17 = 2328848693411712031503691941858813622702848549413793019632371533620884932783396569389467056721797172 79
2124383912996912878385301576015544677059069603758268484593713279004736321636208727786133696476089021405973277938
3020349204803205725870225429985939570141508220041286857810048164696707018663758416807708910671477407366098883430
8118619330149734093901799485777125797493522994403105436890356514653998679084288855412377761434043763334429493970
6324922370235505157179055515120386682186790853173378878497866747870767298453951243154955867246775271200451930031
899920810207673250141258910490473498378989535875366407748689452 9499
    c17 = 1073825441811407654807144884496404646814162174060321438498635418910523697707100142927156063642807597 04
5989095827494176252811644517116104004083335787613468974984694005261939275039468350481608119343235066945244611328
5638982551762586656329109007214019944975816434827768882704630460001209452239162896576191876324662333153835533956
6002952551583770251984269509440406432354302110110635860324677243297357859473720517590421381710541658548424729905
8380089998489323254909276640051030008358551301417122042310345229289149614180695630039654068238166836756456942781
309206405399310353763599431114301070881485186723970649257720389 9024
    n18 = 1959144138395852943559872911393634665700135257835790934765725723977754042481174981778306123323581791 65
6068913834404149773274901151973630303898627739403671879097137465683274105454705641777150123449476850978036907544
3550907847298246275717420562375114406055733620258777905222169702036494045086017381084272496162770259955811174440
4901265147478766613177506494887749923480050443890811016860164462192640699713706463195464297829048100630203247041
3849560876153256331069975332244487106038369304448193226580150581964699853519208303687255168340576612396848790764
898090071211805234617453351397800913175716754759585755237058635 3973
    c18 = 3834917098887202931981968704659119341624432294759361919553937551053499607440333234018189141970246302 29
9385742548278589896033282894981200353270637127213483172182529890495903425649116755901631101665876301799865612717
7503600890851791427506646034541936420530163847145158558683687235089222717671902855211377856880756228329248292483
6277447645623282688580104696938451954938542825959156671689084460469625878363939085415303932948072620514719924718
3621535172450825979047132495439603840806501254997167051142427157381799890725323765558803808030109468048682252028
720241357478614704610089120810367192414352034177484688502364022 887
    n19 = 1925424257158843017130819175787126107535852115862474570274405755605465233249596119679536963048478293 02
9200323873026739646249173355771537995696969423826790898525169983470773440077531145286892433086650242957695193427
9223234676654749272932769107390976321208605516299532560054081301829440688796904635446986081691156842271268059970
7620042592190367531749099423432044327950763774321076302036217545528041244087923582200718623694432015841557118933
8887735013802323862456661655124680405472049281622665146701780250409407061489255644442591592026948586179953247338
330462206449322362755255834408883986017829458948189920631886331 0603
    c19 = 6790553533991297205804561991225493105312398825187682250780197510784765226429663284220400480563039341 93
```

```
859978334672405107621126566346864382643010901324501403581117829508193995868708747731286772028996450609781976209524447912935999886767181181973819668788469668046345866137431099461076000947426411575020492087552743448643753662358968451941151910017029142336742493856682031548650744420202240800387911846576127391675529089811299152554611419106402299132972437006463256990385618923617789400776669078263024744389535889398373582282424348718185109878727127025678089109440512194763108872991739831765232049776510179013267917188 9
    n20 = 268097002511712791029749629491844111364593722676205351984214498332984480925804974853019537966191853393160643877980922202986304282075564828057398034202790561911943600496517674125726091876805080730746532913509982539387932692142304571171944348538887653034033858247862318594503512124494048707763202974197124865748047943256027603473064329272817161603688301879449401289079710278385100795194668461761065651647309639888924002400630893977204149213989363999279482351950852021712647288161845326511382218622409696551855966282858140570824483217495679439462737761846576981044650627492443270925882379279964196201702544238378768066599 c20 = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752426501104314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980064667130174890152813215371291330117925487987744132228591454497451972730731100233035053485786751646661247476975357785866007583059289140355186724605739783968832917253017718704222902868586203614077906577106193352813742301940731147358183240589908970925174700278803200209449537961468654467296907324930970348255638602462281473101576781004296981375254861746497491571442559535194026607702167240985864542734 6
    nList = [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20]
    cList = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15, c16, c17, c18, c19, c20]

    print(solve(nList, cList))
```

运行结果

> n[4]和n[17]存在公约数为
> 132585806383798600305426957307612567604223562626764190211333136246643723811046149337852966828729052476725552361132437370521548707664977123165279305052971868012755509160408641100548744046621516877981864180076497524093201404558036301820216274968638825245150755772559259575544101918590311068466601618472464832499
> 分解n[4]得到:
> 228220397330493881109367781730147656636633038117912832343612306497758059239021734385539278054074631061046997739941583757040330934717613877998521683378985269805217536143078996690159313878199274218753163045915219015928238144177564476957010458467735086293713970130536845530421857250599967915323916264297124169949908896937328051819479700714293095996149737727365562994042464247916606792538849400217288469063441988547791919517397193429087613306619104771199334285507742429104209524969296056861547994878399234243363537474421535716780645207631497932943607878217517035432886967269239096703968215510530480356194997063911181450677 = 132585806383798600305426957307612567604223562626764190211333136246643723811046149337852966828729052476725552361132437370521548707664977123165279305052971868012755509160408641100548744046621516877981864180076497524093201404558036301820216274968638825245150755772559259575544101918590311068466601618472464832499 * 172130338499326278748088659642118539903263306644625489813269854049704514120598134934786316771912260248369075948864036229605563950070491992643125838594149381631362120542615545158696925360916086470107987771246645459433841320759048661246016875180635458357799131806734777129141845728102816378815607663660131827433
> b'flag{abdcbe5fd94e23b3de429223ab9c2fdf}'

flag

> flag{abdcbe5fd94e23b3de429223ab9c2fdf}

[HDCTF2019]bbbbbbrsa(基础题)

题目

enc文件：

```
p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
c =
==gMzYDNzljMxUTNylzNzljMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwljN2QzM5ADMwlDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM
```

encode (1).py：

```
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
```

解题思路

已知{p, n, c}，q = n / p

现在要计算解密私钥d的话，还需要知道e的值

从加密代码中看到e的范围是(50000,70000)，直接爆破就行了

当然，c是经过base64和倒序的，首先要还原成数字

附上代码：

```python
import base64
import gmpy2
import binascii

c = "==gMzYDNzIjMxUTNyIzNzIjMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwIjN2QzM5ADMwIDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM"
c = int(base64.b64decode(c[::-1]))  #base64解码+倒序，还原出c
n = 374218295098877962748971622493673294009886471456133253673379680633413727260 61
p = 17707738967525769504250799816500646084 9
q = n // p
phi = (p-1)*(q-1)
for e in range(50000, 70000):
    if gmpy2.gcd(e,phi) == 1:
        d = gmpy2.invert(e,(p-1)*(q-1)) #计算私钥d
        m = hex(gmpy2.powmod(c, d, n))    #解密
        if len(m) % 2 == 1: #不是整字节，不能转成文本的直接跳过
            continue
        flag = str(binascii.unhexlify(m[2:]))
        print(flag)
        if "flag" in flag or "FLAG" in flag:
            print(e)
            break
```

爆破速度很快，e = 51527就出结果了，基本上是秒出

flag

> flag{rs4_1s_s1mpl3!#}

[BJDCTF2020]RSA(基础题)

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:
12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755798009656547738616399025300123043766255518596149348930444599820675230046423373053051631932557230849083426859490183732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018698992935818206109087568166097392314105717555482926141030505639571708876213167112187962584484065321545727594135175369233925922507794999607323536976824183162923385005669930403448853465141405846835919842908469787547341752365471892495204307644586161393228776042015534147913888338316244169120  13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037381631268825806469518166370387352035475775677163615730759454343913563615970881967332407709901235637718936184198930226303761876517101208677107311006065728014220477966000620964056616058676999878976943319063836649085085377577273214792371548775204594097887078898598463892440141577974544939268247818937936607013100808169758675042264568547764031628431414727922168580998494695800403043312406643527637667466318473669542326169218665366423043579003388486634167642663495896607282155808331902351188500197960905672207046579647052764579411814305689137519860880916467272056778641442758940135016400808740387144508156358067955215018979153370552535153498477459720877329811204688208387543826122582132404214848454954722487086658061408795223805022202997613522014736983452121073860054851302343517756732701026667062765906277626879215457936330799698812755973057557620930172778859116538571207100424990838508255127616637334499680058645411786925302368790414768248611809358160197554369255458675450109457987698749584630551177577492043403656419968285163536823819817573531356497236154342689914525321673807925458651854768512396355389740863270148775362744448115581639629326362342160548500035000156097215446881251055505465713854173913142040976382500435185442521721  12806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
'''
```

解题思路

这里进行了两次加密，分别是：

1. 对flag加密，已知{n1, c1}
2. 对"BJD"*32加密，已知{n2, c2}

同时还已知pow(294, e, n1)，根据这个可以爆破求解e，e的范围是(e<100000)

然后在这两次加密中，复用了同一个大素数因子q

即n1 = p1 * q；n2 = p2 * q

所以很简单，计算n1和n2的公约数即可得出q，p1 = n1 // q

至此，{n1, p1, q, e, c}俱全

附上代码：

```
import base64
import gmpy2
import binascii

c1 = 12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755
79800965654773861639902530012304376625551859614934893044459982067523004642337305305163193255723084908342685949018373230375174400487418306259485687031861428999167598006354831649948690892320962756387155487561270207910056701869
89929358182061090875681660973923141057175554829261410305056395717088762131671121879625844840653215457275941351753692339259225077949996073235369768241831629233850056699304034488534651414058468359198429084697875473417523654718924952043076445861613932287760420155341479138883383162441691200
n1 = 13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037
n2 = 12806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
c0 = 38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889859846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501 8
#c0 = pow(294,e,n1)
for e in range(100000):
    if gmpy2.powmod(294,e,n1) == c0:
        print("e = ", e)
        break
q = gmpy2.gcd(n1, n2)
p = n1 // q
d = gmpy2.invert(e,(p-1)*(q-1))
m = gmpy2.powmod(c1, d, n1)
flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```

运行结果

```
e = 52361
b'BJD{p_is_common_divisor}'
```

flag

```
flag{p_is_common_divisor}
```

[BJDCTF2020]rsa_output(共模攻击)

题目

```
{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,2767}
{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,3659}
message1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599
message2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227
```

解题思路

观察一下给出的内容，应该是已知{n1, e1, c1}, {n2, e2, c2}，求解明文

注意到这里n1和n2是相同的，所以是共模攻击

共模攻击是指生成秘钥的过程中使用了相同的模数n，此时用不同的秘钥e1，e2加密同一信息m，得到不同的密文c1，c2，即

$$m^{e1} \% n = c1$$
$$m^{e2} \% n = c2$$

因为e1和e2都是素数，所以由扩展欧几里得算法

可以计算出s1和s2，使得

$$e1 * s1 + e2 * s2 = 1$$

根据这些信息，可以直接计算m

公式推导如下：

> $m = m \% n$
> $m = m^1 \% n$
> $m = m^{(e1*s1+e2*s2)} \% n$
> $m = (m^{e1*s1} \% n * m^{e2*s1} \% n) \% n$
> $m = (c1^{s1} \% n * c2^{s2} \% n) \% n$

附上代码：

```
import gmpy2
import binascii

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

#扩展欧几里得算法
#return (r,x,y) 其中，r为a和b的最大公约数，xy满足ax + by = 1
r, s1, s2 = gmpy2.gcdext(e1, e2) #计算s1，s2
m = (gmpy2.powmod(c1,s1,n)*gmpy2.powmod(c2,s2,n)) % n #计算明文m

flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```

运行结果

> b'BJD{r3a_C0mmoN_moD@_4ttack}'

flag

> flag{r3a_C0mmoN_moD@_4ttack}

[ACTF新生赛2020]crypto-rsa0(基础RSA+伪加密)

题目

给出challenge.zip打开发现提示需要输入密码

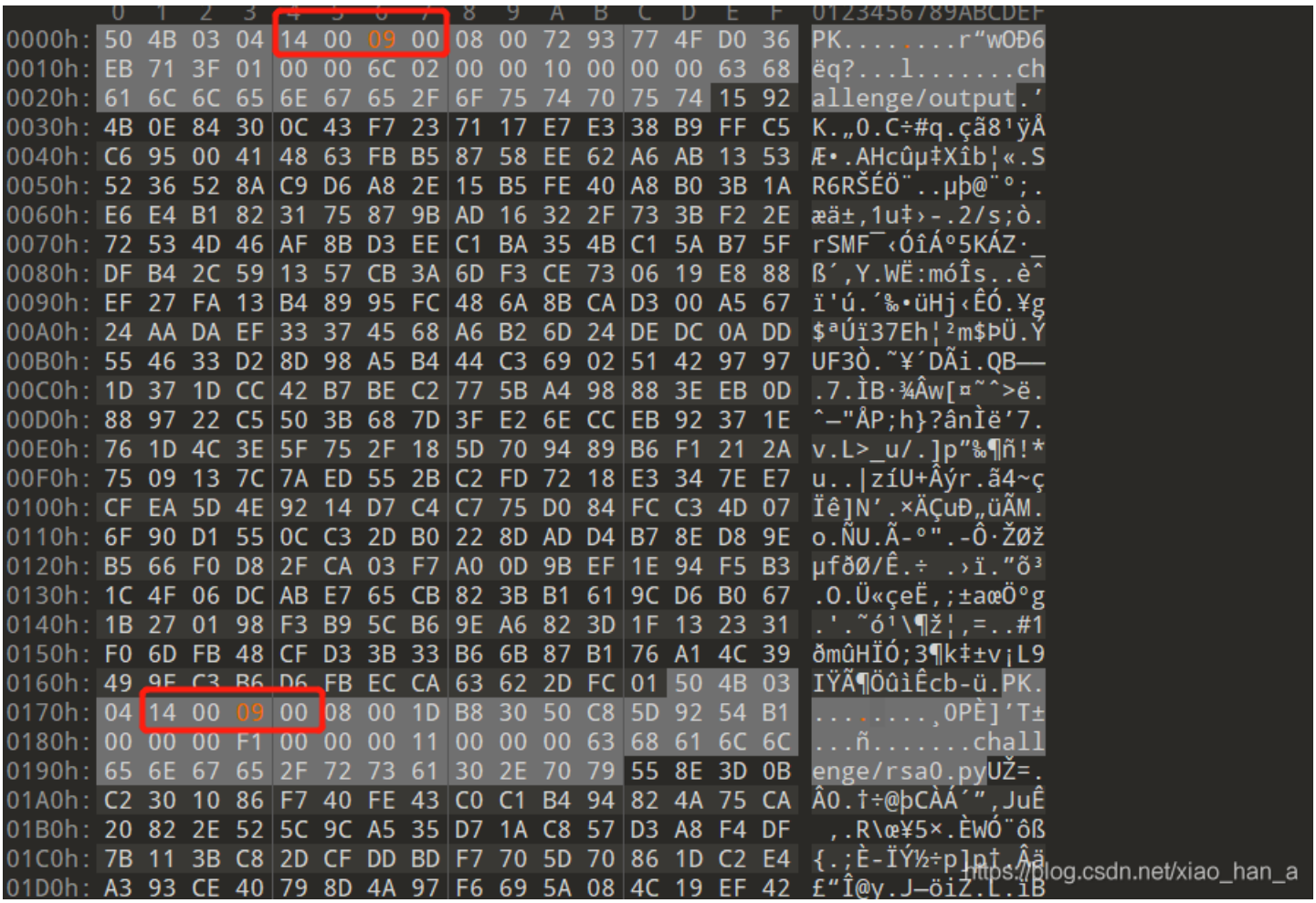
再看一下hint.txt：

> 怎么办呢，出题人也太坏了，竟然把压缩包给伪加密了！

伪加密？？？我开始还以为是空密码，但是尝试之后发现不能输入空密码

没办法，又是知识盲区，上网学习一波

伪加密原理



上面是用010-editor打开该压缩文件的屏幕截图
一个zip文件由三部分组成：压缩源文件数据区+压缩源文件目录区+压缩源文件目录结束标志
压缩源文件数据区：
50 4B 03 04：这是头文件标记（0x04034b50）
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密）
压缩源文件目录区：
50 4B 01 02：目录中文件文件头标记(0x02014b50)
3F 00：压缩使用的 pkware 版本
14 00：解压文件所需 pkware 版本
00 00：全局方式位标记（有无加密，伪加密的关键）
关注到文件头的加密位标记，伪加密就是把未加密的文件，通过修改加密位，让其转换成了伪加密文件

因此，查找所有的文件头中的14 00，把14 00后面的加密位都改成00 00，伪加密文件就变成了无加密文件

得到里面的output文件

```
90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083
99237153883494964743436193853978459947060210411
75470056738777382578357297600377652133400366963507663242291436131799321451221306857785040624101370436359582
08805698698169847293520149572605026492751740223
50996206925961019415256003394743594106061473865032792073035954925875056079762626648452348856255575840166640
51933486269006394931651575025654593749821347628663745580345289078126444603073236987104487035983856861817658
62060410550002979817332728160898060144008463923077420655593318749722748449920478494722033903 50
```

里面的rsa0.py

```
from Cryptodome.Util.number import *
import random

FLAG=#hidden, please solve it
flag=int.from_bytes(FLAG,byteorder = 'big')

p=getPrime(512)
q=getPrime(512)

print(p)
print(q)
N=p*q
e=65537
enc = pow(flag,e,N)
print (enc)
```

已知{p, q, e, c}，是道基础题，直接解题

附上代码：

```
import gmpy2
import binascii

e = 65537
p = 90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992371538834949647434361938539784599470602104 11
q = 75470056738777382578357297600377652133400366963507663242291436131799321451221306857785040624101370436359582088056986981698472935201495726050264927517402 23
c = 50996206925961019415256003394743594106061473865032792073035954925875056079762626648452348856255575840166640519334862690063949316515750256545937498213476286637455803452890781264446030732369871044870359838568618176586206041055000297981733272816089806014400846392307742065559331874972274844992047849472203390350
n = p * q
d = gmpy2.invert(e,(p-1)*(q-1))
m = gmpy2.powmod(c, d, n)
flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```

运行结果

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

flag

flag{n0w_y0u_see_RSA}

[GWCTF 2019]BabyRSA(基础RSA+解方程)

题目

secret：

```
N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061631m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532391m2=48744398575740517342662818837565711760423550793696752299325797210887228369830523845446572321422687141427678891205818619703982124291273674282408062768097180251120691439467215924020691073585065199931610001469106729570813863936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891367546618155100552706530951536495061013720639325714835765966668709166274984856022545382636227170429269284759633953322908803882053208610942115857584107760126871317509787408353624900601894878941323878392284563349402360886525607196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954 6
```

encrypt.py：

```
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()
```

解题思路

已知{n, e, m1, m2}(这里m指输出的密文)，求解明文

这里先看加密代码，flag被平分成两部分F1和F2。

然后令c1 = F1 + F2(这里F均为长整型，c1是算术相加)，c2 = pow(F1, 3) + pow(F2, 3)

再用RSA加密出m1和m2

RSA部分，模数n可以直接放到网站上分解，就不费功夫了

factordb.com

得到p和q之后，自然可以解密算出c1和c2

后面的问题就是求解一个简单的二元三次函数：

> $F1 + F2 = c1$
> $F1^3 + F2^3 = c2$

这里推导后得出的求解公式为：

> 令$t = (c1^3 - c2)/(3*c1)$
> $F1 = (c1+sqrt(c1^2-4t)) / 2$ #二次方程的求根公式
> $F2 = c1 - F1$

附上代码：

```
import math
import gmpy2
import binascii
n = 6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061631
m1 = 900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532391
m2 = 48744398575740517342662818837565711760423550793696752299325797210887228369830523845446572321422687141427678891205818619703982124291273674282408062768097180251120691439467215924020691073585065199931610001469106729570813863936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891367546618155100552706530951536495061013720639325714835765966668709166274984856022545382636227170429269284759633953322908803882053208610942115857584107760126871317509787408353624900601894878941323878392284563349402360886525607196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954 6
p = 7978628639024219849512313504303122605177732696849584563428609832361841296023909190260484961197571877020764995513107941779179201376468358888627061269240884115709971412571595639527258822141811855312091869723514699462695085113128637791232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777476997
q = 7978628639024219849512313504303122605177732696849584563428609832361841296023909190260484961197571877020764995513107941779179201376468358888627061269240884115709971412571595639527258822141811855312091869723514699462695085113128637791232053223784521942612170165525277545132155203294999671081969688331633297246202510960803777487377
e = 65537
d = gmpy2.invert(e,(p-1)*(q-1))
c1 = gmpy2.powmod(m1, d, n) #c1 = F1 + F2
c2 = gmpy2.powmod(m2, d, n) #c2 = pow(F1, 3) + pow(F2, 3)
t = ((c1**3)-c2)//(3*c1)
F1 = (c1+gmpy2.iroot(c1**2-4*t,2)[0])//2
F2 = c1 - F1
f1 = binascii.unhexlify(hex(F1)[2:])
f2 = binascii.unhexlify(hex(F2)[2:])
print(f1+f2)
```

求解的时候要注意，开方不能使用math.sqrt，而要用gmpy2.iroot，前者在计算大数的时候会有明显误差

运行结果

> b'GWHT{f709e0e2cfe7e530ca8972959a1033b2}'

flag

> flag{f709e0e2cfe7e530ca8972959a1033b2}

SameMod(共模攻击)

题目

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

message1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311 1537349
message2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323 8309535
```

解题思路

提示非常明显，SameMod，相同的模数，即共模攻击

观察一下给出的内容，已知{n1, e1, c1}, {n2, e2, c2}，这里有n1 = n2，求解明文

共模攻击原理参见文章上面那道题: BJDCTF2020rsa_output

附上代码：

```
import gmpy2
import binascii

e1 = 773
e2 = 839
n = 626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9
c1 = 345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311 1537349
c2 = 567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323 8309535
#扩展欧几里得算法
#return (r,x,y) 其中，r为a和b的最大公约数，xy满足ax + by = 1
r, s1, s2 = gmpy2.gcdext(e1, e2) #计算s1，s2
m = (gmpy2.powmod(c1,s1,n)*gmpy2.powmod(c2,s2,n)) % n #计算明文m
print(m)
mStr = str(m)
i, flag = 0, ''
while i < len(mStr):
    if mStr[i] == '1':
        flag += chr(int(mStr[i:i+3]))
        i += 3
    else:
        flag += chr(int(mStr[i:i+2]))
        i += 2
print(flag)
```

运行结果

```
102108971031231191041011101191011161041051101071051161051151121111151151059810810112 5
flag{whenwethinkitispossible}
```

还有一个点要注意，一般计算出明文之后都是十六进制转Ascll文本。

但是这道题比较特殊，是十进制转Ascll码，注意区分ascll值是两位数还是三位数

flag

> flag{whenwethinkitispossible}

[WUSTCTF2020]babyrsa(基础题)

题目

```
c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 73069886771625642807435783661014062604264768481735145873508846925735521695159
e = 65537
```

解题思路

一个签到题，已知{n, e, c}，求解明文

拿n去分解再计算私钥d解密就行了，没什么好说的

因数分解网站factordb.com

运行结果

> b'wctf2020{just_@_piece_0f_cak3}'

flag

> flag{just_@_piece_0f_cak3}