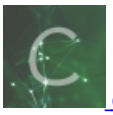
BUUCTF 打卡2

原创

路由(_)生 于 2021-08-20 22:01:55 发布 39 收藏

分类专栏： crypto 文章标签： 动态规划

本文链接：https://blog.csdn.net/qq_52193383/article/details/119815926


crypto 专栏收录该内容

35 篇文章 3 订阅

订阅专栏

1.RSA4

题目给出了三组c和n，猜测是广播攻击。猜测e = 3。自己写的脚本有点不靠谱，就借用书上的。值得注意的是题目给出的c和n都是5进制数！！！（仔细点的话可以看出数据中没有出现>=5的数字）

```
#广播攻击？？？
import gmpy2,libnum
from Crypto.Util.number import long_to_bytes,bytes_to_long


def broadcast_attack(data):
    def extended_gcd(a,b):
        x,y = 0,1
        lastx,lasty = 1,0
        while b:
            a,(q,b) = b,divmod(a,b)
            x,lastx = lastx-q*x,x
            y,lasty = lasty-q*y,y
        return (lastx,lasty,a)
    def chinese_remaindor_theorem(items):
        N = 1
        for a,n in items:
            N *= n
        result = 0
        for a,n in items:
            m = N//n
            r,s,d = extended_gcd(n,m)
            if d != 1:
                N = N//n
                continue
            result += a*s*m
        return result%N ,N
    x,n = chinese_remaindor_theorem(data)
    m = int(gmpy2.iroot(x,3)[0])
    return m

N1 = int('331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400220
031202142322434104143104244241214204444443323000244130122022422310201104411044030113302323014101331214303223312
024304024044130332431321010104222401331222114004340232222142314024034032000122210233413333400423431223021134102
```

```
011022123324130302443133000130340402010444244312013000033411004243201020340144040401000344200122304221144200141300
4',5)
c1 = int('3100200042340333042442004214144133203413010021230303112023402224103014234403124124402402441102001121411402012240324022321312042130123032044220033000040114341021413212233112432420100141404224113423043222012411124021322031011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343332113021024131111142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230104224324300324401233400340443142234004012241113230002422344204412404110210231002220031232143430301220323010422432',5)

N2 = int('3022400000404214101444221333341431400110110443222231444120022202430011411411141232233313313043211302123120432223312012144443421004123221414441324443442430231122214322440230243210224213224403201002011322401112104323214322120342424313404431402221202434310004234200243233114430021421241403341412000434421133022402030122303333432424403120424012230124223201130321122004422241113440301213242031111030244234402112210122441123000220334414014304411431',5)
c2 = int('1122002034040134303302141240044044232100413210430003032331414233441442223434010422003340332031240300114400142101121032344403121340321234004443441442330201301101340421022203020024133211020224141304430411442403101210201003101043342042344124114244203212111122320311213303103334144234333433220244001212003333304322234214333441220230124400130414014232022101240244310400134143131211234334241131134144220433304220023141441111341420443334041122403443',5)

N3 = int('3322003244100411114342221230431213314421032333324223410413404120342300033144203113331013442312121302003120410443244311410330043331100210130201400200112220123000200413420400040022202102231221113141121243332111322303321240224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230301210341133344044404212422401131032030133412313300043320403024400113240041303240343234301431024014401302423214240203233',5)
c3 = int('1001344412014113032243320412400224222433233401112421001244024140234210041033113144130324201100210132304040331112042130442222220032440224424332242244441404334213011111133002221320303032442210113303221204204224310143434220320412104211321210421242333033113431131111414320001124000211131212223434000340331204040104302143311203133432432212330411234001403013202143210113021124113442241344231201304214121200310221130032140404301212433201324043124242',5)
data = [(c1,N1),(c2,N2),(c3,N3)]
print(long_to_bytes(broadcast_attack(data)))
```

2.[AFCTF2018]可怜的RSA

对public.key文件的内容进行公钥解析得e和n。对flag.enc文件的内容进行base64解码得密文。不过按照往常得写法是写不出的。网上大佬说c和n的长度一样，这样就要拆解c，使c的长度小于n。而RSA的库自带PKCS1_OAEP 可以padding，所以可以解出。

RSA库的使用

RSA密钥长度、明文长度和密文长度

网上这方面的资料有点少而且重复，就RSA库的使用来说我觉得没有解释那些函数返回的是什么东西，有点难懂。

代码中的注解不严谨也不详细：)。

```
import gmpy2,base64
from Crypto.Util.number import long_to_bytes,bytes_to_long
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

txt = 'GVd1d3viIXFfcHapEYuo5fAvIiUS83adrtMW/MgPwxVBSl46joFCQ1plcnlDGfL19K/3PvChV6n5QGohzfVyz2Z5GdTlaknxvHDUGf5HC
ukokyPwK/1EYU7NzrhGE7J5jPdi0Aj7xi/Odxy0hGMgpaBLd/nL3N8O6i9pc4Gg3O8soOlciBG/6/xdfN3SzSStMYIN8nfZZMSq3xDDvz4YB7TcT
Bh4ik4wYhuC77gmT+HWOv5gLTNQ3EkZs5N3EAopy11zHNYU80yv1jtFGcluNPyXYttU5qU33jcp0Wuznac+t+AZHeSQy5vk8DyWorSGMiS+J4KNq
SVlDs12EqXEqqJ0uA=='
c= base64.b64decode(txt)

e = 65537
n = 79832181757332818552764610761349592984614744432279135328398999801627880283610900361281249973175805069916210179560506497075132524902086881120372213626641879468491936860976686933630869673826972619938321951599146744807653301076026577949579618331502776303983485566046485431039541708467141408260220098592761245010678592347501894176269580510459729633673468068467144199744563731826362102608811033400887813754780282628099443490170016087838606998017490456601315802448567772411623826281747245660954245413781519794295336197555688543537992197142258053220453757666537840276416475602759374950715283890232230741542737319569819793988431443
p = 3133337
q = 25478326064937419292200172136399497719081842914528228316455906211693118321971399936004729134841162974144246271486439695786036588117424611881955950996219646807378822278285638261582099108339438949573034101215141156156408742843820048066830863814362379885720395082318462850002901605689761876319151147352730090957556940842144299887394678743607766937828094478336401159449035878306853716216548374273462386508307367713112073004011383418967894930554067582453248981022011922883374442736848045920676341361871231787163441467533076890081721882179369168787287724769642665399992556052144845878600126283968890273067575342061776244939
phi = (p-1)*(q-1)
d = int(gmpy2.invert(e,phi))
#print(len(str(bytes_to_long(c))) == len(str(n)))
rsa_components = (n,e,d,p,q)
arsa = RSA.construct(rsa_components)#构建RSA密钥
rsakey = RSA.importKey(arsa.exportKey())# .exportKey() 私钥e的生成
rsakey = PKCS1_OAEP.new(rsakey)#返回密码对象，用于执行 PKCS1_OAEP 加密或解密
decrypted = rsakey.decrypt(c)#对密文解密,c为bite串
print(decrypted)

'''
c = bytes_to_long(c)
m = pow(c,d,n)
print(long_to_bytes(m))
'''
```

3.[ACTF新生赛2020]crypto-classic0

小z同学的生日？？？：）。原来是ziperello。那好，给它解密吧，既然是生日，那应该是纯数字并且应该是8位数。得到加密脚本后就可以写出解密脚本了。

```
#include<stdio.h>

char flag[25] = "Ygvd mq[lYate[elghqvakl}";

int main()
{
// int i;
// for(i=0;i<25;i++)
// {
//  flag[i] -= 3;
//  flag[i] ^= 0x7;
//  printf("%c",flag[i]);
// }
 int i;
 for(i=0;i<25;i++){
  flag[i]^=0x7;
  flag[i]+=3;
  printf("%c",flag[i]);
 }
 return 0;
}
```

用python写的时候连着写，不知道+的运算优先级大于^的运算优先级。真魂淡：）。

```
txt = 'Ygvd mq[lYate[elghqvakl}'
for i in txt:
    print(chr((ord(i)^7)+3 ),end='')
```

4.RSA & what

从所给出的代码可以看出是共模攻击，明文是比特类型。根据代码中的加密过程写出解密过程。一开始输出的txt为多段base64编码。

```
b'VEhJUz==\nRkxBR3==\nSVN=\nSElEREVOLo==\nQ0FO\nWU9V\nRklORM==\nSVT=\nT1VUP4==\nRE8=\nWU9V\nS05PV9==\nQkFTRTY0P5
==\nWW91bmdD\nVEhJTku=\nWU9V\nQVJF\nTk9U\nVEhBVE==\nRkFNSUxJQVI=\nV0lUSO==\nQkFTRTY0Lh==\nQmFzZTY0\naXO=\nYW==\n
Z3JvdXA=\nb2b=\nc2ltaWxhcn==\nYmluYXJ5LXRvLXRleHR=\nZW5jb2Rpbme=\nc2NoZW1lc0==\ndGhhdD==\ncmVwcmVzZW50\nYmluYXJ5
\nZGF0YW==\naW5=\nYW6=\nQVNDSUl=\nc3RyaW5n\nZm9ybWF0\nYnk=\ndHJhbnNsYXRpbmd=\naXS=\naW50b1==\nYT==\ncmFkaXgtNjQ=
\ncmVwcmVzZW50YXRpb24u\nVGhl\ndGVybc==\nQmFzZTY0\nb3JpZ2luYXRlc8==\nZnJvbd==\nYY==\nc3BlY2lmaWN=\nTUlNRT==\nY29u
dGVudI==\ndHJhbnNmZXI=\nZW5jb2Rpbmcu\nVGhl\ncGFydGljdWxhct==\nc2V0\nb2b=\nNjR=\nY2hhcmFjdGVyc5==\nY2hvc2Vu\ndG+=
\ncmVwcmVzZW50\ndGhl\nNjQ=\ncGxhY2UtdmFsdWVz\nZm9y\ndGhl\nYmFzZd==\ndmFyaWVz\nYmV0d2Vlbt==\naW1wbGVtZW50YXRpb25z
Lp==\nVGhl\nZ2VuZXJhbI==\nc3RyYXRlZ3n=\naXO=\ndG9=\nY2hvb3Nl\nNjR=\nY2hhcmFjdGVyc5==\ndGhhdA==\nYXJl\nYm90aN==\n
bWVtYmVyc5==\nb2a=\nYS==\nc3Vic2V0\nY29tbW9u\ndG8=\nbW9zdM==\nZW5jb2RpbmdzLA==\nYW5k\nYWxzb8==\ncHJpbnRhYmxlLg==
\nVGhpc9==\nY29tYmluYXRpb25=\nbGVhdmVz\ndGhl\nZGF0YW==\ndW5saWtlbHk=\ndG/=\nYmV=\nbW9kaWZpZWS=\naW5=\ndHJhbnNpdE
==\ndGhyb3VnaN==\naW5mb3JtYXRpb26=\nc3lzdGVtcyw=\nc3VjaN==\nYXM=\nRS1tYWlsLD==\ndGhhdA==\nd2VyZQ==\ndHJhZGl0aW9u
YWxseQ==\nbm90\nOC1iaXQ=\nY2xlYW4uWzFd\nRm9y\nZXhhbXBsZSw=\nTUlNRSdz\nQmFzZTY0\naW1wbGVtZW50YXRpb24=\ndXNlcw==\n
QahDWiw=\nYahDeiw=\nYW5k\nMKhDOQ==\nZm9y\ndGhl\nZmlyc3Q=\nNjI=\ndmFsdWVzLg==\nT3RoZXI=\ndmFyaWF0aW9ucw==\nc2hhcm
U=\ndGhpcw==\ncHJvcGVydHk=\nYnV0\nZGlmZmVy\naW4=\ndGhl\nc3ltYm9scw==\nY2hvc2Vu\nZm9y\ndGhl\nbGFzdA==\ndHdv\ndmFs
dWVzOw==\nYW4=\nZXhhbXBsZQ==\naXM=\nVVRGLTcu'
```

将它们分开，依次进行base64解码。单纯的输出的话是比特类型，每段之间有 “b’”，看着有点碍眼。想着把它弄掉。想出了两种方法，但都有缺陷。

1.如果进行bytes.decode()的话，b’A\xa8CZ,’ 和b’a\xa8Cz,'会报错。

2.发现取比特串中的某一位输出，输出的是对应的ASCII码，再挨个转成字符。但当遇到上述比特串中的具有特殊含义的字符形成错误的字符。

进行上述转换后，发现是一段简介（修正了一下）：

```
THIS FLAG IS HIDDEN. CAN YOU FIND IT OUT? DO YOU KNOW BASE64? YoungC THINK YOU ARE NOT THAT FAMILIAR WITH BASE64
. Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string for
mat by translating it into a radix-64 representation. The term Base64 originates from a specific MIME content tr
ansfer encoding. The particular set of 64 characters chosen to represent the 64 place-values for the base varies
 between implementations. The general strategy is to choose 64 characters that are both members of a subset comm
on to most encodings, and also printable. This combination leaves the data unlikely to be modified in transit th
rough information systems, such as E-mail, that were traditionally not 8-bit clean.[1] For example, MIME's Base6
4 implementation uses A\xa8CZ, a\xa8Cz, and 0\xa8C9  for the first 62 values. Other variations share this proper
ty but differ in the symbols chosen for the last two values; an example is UTF-7.
```

```
import gmpy2,libnum,base64
from Crypto.Util.number import long_to_bytes

N = 785095419718268286865082143048169854470772937668193987280464111669178108204847593142910289764982236612293950094740631737051626270376109935396177519054430392782275835046048082519310838189094676132775878745457610743644275499665555193719138598753135772822430531500562746677980496946957036603135329331654493129497255817089654172730555822162959945876009759701248114962700808969770769460001027010302609905981814664472080547133915263137006813410939222403174281735990316241251551882164894768256061915211820349691203432876911813003996835154148092627004575258766918081802577303517076736603806989738846423068988100006336848787154028231435491398507329828974596980896495611907468506981302994580802555823126968731492100282408981378228884925599576650679365733563675897845931190166240724338727445374320059116684944557333306893851412146530918880177820490434348626203067834361698565641759298711006699134389808992195793298977532334509347701939154347914277286365862180498746172317053080037200662693127291357641756986110688084040541255815401149564636032402224979193846917187440140025542016023959693129999941595995360263598790602180564963457454574939197713376011774498990665798576300363508710904526498307750296954885755749850784285600542531808637253641470
e1 = 1697
e2 = 599
gcd = gmpy2.gcdext(e1,e2)#返回(最大公因数,s(e1的),t(e2的))
s,t = int(gcd[1]),int(gcd[2])
txt = b''
c1 = [412629526163150748619328091306742267675740578011800062477174189782151273970783531227579758540364970485350157944321579108232221072397135934034064481497887079641131808838242743811511451355024436983050572020925065644355566434625618133203024215941534926113892937988520918939061441606915556516246057349589921494351383160036280826024605351878408056180907759973804117263002554923041750587548819746346813966673034182913325507826219961923932100526305289894965216608254252188398580139545189681875824089456195044984585824938384521905334289906422454152976834867304693292466676355760173232407753256256317546190171995276258924613533179898467683358934751999655196790168438343198229183747091108262988777659858609744709324571850262293294975336628234767258858873839342596887193772615000676401522431518310648303975593582965021189182246986957349253156736526071639973844039068996404290548474640668851856078201093335425412842295604919065487301340901573809617549185106072798799159726375235125260509158832996701927878713084753334549129580912412168594170659605421750204835970231909591063407612779337478065175988365401590396247575670934372719610605847716694567011786898902590302399814285033895698581613180534954905937704747713127084757909562838456964563682165
      ,494644347943710545224678831941589086572700792465459558770782213550069709458568349686998660541810166872034041584767487150140111151788221460027897193248273461607411027815984883969396220626358625041781558277804930212654296704055890683796941327712758797770820006623289146990000114915293539639766846910274034245607746230740851938158390562286057002223177609606376329007676845450142537930798148258428701466415483232670659815791064681384406494388237742330786225557303988025468036820082959712050733095860546860468575857084616069132051094882919253745234762029759124776348047587755897123575123506976140900565238840752841856713613368250071926171873213897914794115466890719123299469964019450899291410760762179836946570945555295288184698184555018368687708432612286248476073758067175481771199066581572870175460016017100414479346437034291784837132240891321931601494414908927713208448927221095745802380014441841139882391378410438764884597938773868771896252329517440068673532468372840830510218585255432000690265226016573313570977945083879214961394087065558376158826938257664840570952233832852869328755681754345162477203565202426022995103743174881827387327000788796657459096037664821001380014170236806477178243231433888578175957661721528834842747182480
      ,152942283599728307168144137370127212672611894072038732126041098102628831053000986759260271210671922070555948023688596575415822984026159010574404359474670428678518262175033880513984372909748992727828381694416776740981021730545374002974037896534944567124543272737618380646771071804878796585983783360553761828325817820260204820004421979881871027255562690952334900616675606524933557440263648233514757200263521499508373975003431306847453046714027687108396945719803444444954079308404947126216395526551292104722047878178373207886033071857277857997932255251315974718248
```

```
82837892164421298202073945919187779856785892717251746704537315003771369737854896595170485152591013676942418134278534037654467840633528916812275267230155352077736583130992587670941654695382287023971261529987384520843829695778029311786431227409189019205818351911572757145556993606643464336196802350204616056286497246016800105003143046120608673496196758720552776772796609670537056331996894322779267635281472481559819839042424017171718303214059720568484939239370144038161541354254182769979771948759413102933987773401644506930205164891773826513161783736386604783484446345744957119469799231796368324927570694496679453313927562345656690240414624431304646248599226046524702364131095964335,797179889362479512654891575836979560318934778588541869910515291618794784882817440623186004709061209600022828865114772945556065030831694493351748644241807010802039933299962265662038346938695257976959696100659919413967239590326800190825068164430415983004776257934330806643464705864163858546921244263485872110265686676948058495547807940337647140165217114675572848467372363749901213168098338199968215928326390240264115204073302062812653901307639481656945745121405187756030401820298187718667495487619388706055901743308879498474208778292401314909024326020056810851808072941768376460625680948757669458903829717900154901633850881446735490850796350832629751542062696791424128974382317197049332586607793107373026802654454377719777499591107449593685862930820160679275485649674008459923800761075227555665317606288233745197187637403782955855355917528873392229473971841163267067999215154311856367408257077827423737834757810526742572929102138439861329874668100272750524167746933634461845189018992025028286703094526223475329326788749908099306825757386538762893841514968071941463086143688210066606268709897846970451602310694284589611077512077710937773946168563052933356038921783275207565543333659751142359811734513681316804048508327731473330137169203716920,123111353650401158556639983459870663057297871992927053886971224773529636525110628183715748795987525113177540092814119928708272290370336537110381023134637759740716140969662183269370676630325583385284994943164692397459103195434968057377474610500216801375394703781249039351368816958227409657934091741509357152328382960684515093945552479461382281913961956745154260686029997827565075768703774895750561575155143606297116391666385705899138085693913246313778033627210312268959737394553510894720099165193981333775907531107232556909478156441457899797515694348816961762796703443502856101079430585547997496001098926600499728389113862894833789669213630332988693669889340482430613291490613803204484751470676686041002772556117213612152322606737150858116122936539131795111263513114569794532805886643087299918196635113037777138666914296986040549274559835214505300618256105508764026461518876579387159881983544667258537064954616097750399839661065797883103731694314852301848272092388637114950059216922969842082648527035538090054093890365647676119748995243416337805666557501345234056968476142608491830438065401219751688687373709390057521910942736632126729711606256158399963682990881473178216060827021373776598901281958527655543318413664277921492723185984,368698068159360469118481958174058173502598908714830631843737283979689094584326250460253762902147299140383875347317622379783390117248588188601811788116394689962062947114958538073112400137862268842651181195463772721545556153631052361928782927033314735476230217443170348194166245628962261945236397935730280066662362718123907590362358674958032559058436364472522254138710387626578013456475844939175762634715873472026643919085701403891269032046023910939908271886750901997506173037735748219263871944788751918288149712966745305193215308053026679259987118350198067611330784032814048893746638750773391689012978194364999209582684836843359983010560683802288735248003839114024908071392689640951650696104546775588087564443815421737828152279209062249310284570736524537774243878735332804559446465929969206179566757862867114475403538834002824025511581699583894501680795684596562691185783537574801581486050670792185299709615627580495598996421507773362176993807541300780422321709160461313225304639945674759530040456417222433393640554592181965443543707213338752353356847244353220006913302297919568568350829733796170116939479496625641511224658770610381962042825824599953904072192931713008887416157709396257948742835873640168712317420719825144985142929 5]
c2 = [592169079372093727306100216011395857825646323934289480976073629037543922902098120901138454462177159996376654176248238979132528728327590301098966139983157980612320563496546128644967731000716697705104079039156276714872147463350811303393260622707024952543509891692246246277965823414460326811240048060543656588688604452353899779068825120910282167004715339763187734797180326976132213325054697165320479166356562518029805927741656605174809726397565772271562066078076105491745903986597877400370206718954975288721072048333678609055008135809089304229015364348490924974097403734627265297637171818849461766523691595241613878709865506436588268999163342945070495338153600520537498539457396582804692959296612715752573140296135784933206146091436617979599749774330699946637591406356289409716084034451049094715202196203486088368791744107629271647320273259836915312794297246589501008666299165717722507702866033454215783240025504356157664454861755286285777763585177751796252655008206383024707883077513745863312079349790275094080707502392866946325796914450602264462588722052297430827681750827349094323968337670311272933785838850649376115667223821665435911506351891489985627506615492005617098615432522564204152887767244129985681083657783356557756654335186,373940646416832740878733255707567753033716583448402000789202767511920210382830343955553654111486728333980557319799362514960627879016797491389812007768832730979916230647641872759001906846747977631675043101794488571281603857011858929145230536693665344088637343056352226255909860064204860925504273010869845631264808149870249805946135429783101292476788266914183353005775775279516236964264354978352281670847380077509142702510019213295214790476628486508089899960856001973093614108632385268021278775237672629215151509849985601366471548657911633165030732852239
```

```
662164410256374522290435100973237243810569763022881368432601639227066929130352224454967160088889465815350045463557442116803907312573099419025873033531399511022448652702954144744887983354046304584897066398051865738748145867367462323588496774775336719683441542429632894155694875798959106609990435787374613004069378289248180026582927698821816687845014392541319968489481207815621588614958838278481394258622495764546891336810095493613144608186589959590982289957022022686496353631055499759323953350765211376042885200820401212866149229865546527000561489665141789359523630369632176198798996713836046384165679504213505462044349021131567200062827208895912888502710760749419277156783060571760,527630926460622936571385649841758214453416849039412401087443444317101857090904711485538107058823056085840539073345920792871368232355473945710983805968354685099973405056043337305477995609988229897474737803077797177155227877244717247664940907839710305946710131682097176867204485795826183784595679790278222719186531696224281538561989078100402243402703624134324950296721232613754009271598315377607099747787081605832526137843582348585831745447779792428879388275736048377668019983813799990764164446838910780938896860554827098386683561209160403521230190192550845137696038038149477745540287178146389514162912746967715154740863514821079531502536169227872623984503762491269996440263824784130809739331730791113051427161333891113992355452792590174247226018486700615568591639438954665539279464125237501665827340057333783284682505689449459122384958779297171017223146781201722284937879649040725839057210747667117322158155610129603945371957578329592686037751129328621059457208539592851875217635579153564281138768932768797756032177189818521145997066995245519739342420457431227441463615969712450340593459153154952321354834646961147703575365762005114909224132081781498693478029887865134514864114098871645160650620849175561207124650742064358314981136058,878643717869894032287788980700995761677735184497986972696235655324405091128398428096066576164931089523045072977431415102053987735969326553978994853162483051544656873294555116009995592043183070208706258164840540599577072097104139505857517663273929851202628854185356185647194933800084230503413037858893307713037149307477830536758283681093517617820169181420796105338681582230788318108428132051793761014952837330456262272828627355701464740578197966332613127307037255647286823496355917642353327912440019621838870388091824748629637425759125214639885130163183752378908729773517053259212525494555880921052679512582051516604297098204363525081039382358483926727008679327719083138865969291911863630382097160230960738043575559330264018212774424527719153248563876760067931499029384228993253862501939337758514377472011933279273181144830381169849387893799390755052093069179605579485710343655570028592595882436632426527654452895431758715126580164902410286422637215098476316042367916779431052267545769495994723721129943616294879642305545894912914632980455031755879087401575310699765408473606166727137934224515998416625122213056208800095077933103150699272650116151674702438463062734472714004926103668378506804002740045547964716693536349447660850580,205314962204511500352858372254132533167549960825498949618514841570703199264867431580754674275990554478140637041427842111391746883257447120035947621456863890934062044010795443059281736346976175772415034838334682726635263432655537852942177334888025283748611576171534251461847349566505628290587224150869640386437623371249743165260396675220683302142805646368906930575140628610003919131999295855501215111393294818218799982703289304596989070475000081175510085432290264502023736899104746316830742226946395027029820825791831870857382647221322734605026210073093918331247494307555600335550942340526536281372036612138713881098866303169425501998978400008829873080965592009371176208668290074288903681417933657472279670688597862835627506340169978450918788539270346340385928840299573889292189531738082166408734046381423516467694328971385421907314814283489322619386570046183556572383980777277173349209330683424343658179781015072259378576130442222984963071166207642585589822061597282467850868050737957726423713761694231879497037175627546427449730638216214828463003483408928375620315193290871300316930139260521382533279767663839278693750409419493280753368451508802658272220767624766390639285308433607255253282702383762149755935518922075584637512494819,271453634732502613378948161256470991260052778799128789839624515809143527363206813219580098196957510291648493698144497567392065251244844074992734669490296293997386198359280316655904691639367482203210051809125904410431506925238374843856343243276508280641059690938930957474434518308646618959004216831130099873532714372402117796666560677624822509159287675432413016478948594640872091688482149004426363946048517480052906306290126242866034249478040406351940088231081456109195799442996799641647167552689564613346415247906852055588498305665928450828756152103096629274760601528737639415361467941349982213641454967962723875032638267311935042334584913897338553953961877439389588793074211502597238465542889335363559052368180212013206172712561221352833891640659020253527584706465205486408990762759230842192028381048563437724528409174790022752557512795782713125166158329880702730769957185428522011430144840232256419113631679343171680631630775266488738173707357123139368825087043785842169049943237537188129367275730984789479909103397937113837824575137021012333461552176687570010445744268373840742899299977372834041925102853718964831225250407279578465008537542659673685686242773379131904890865110699190451534445434533919127658976874721029586168106207]
for i in range(6):
    m = (pow(c1[i],s,N)*pow(c2[i],t,N))%N
    txt += long_to_bytes(m)
print(txt)
```

```
txt = bytes.decode(txt)
txt = txt.split()
for i in txt:
    s = base64.b64decode(i)
    for j in s:
        print(chr(j),end = '')
    print(end = ' ')
```

明文被隐藏！原来是信息隐写，还真没接触过这方面的知识，那就多学学吧：）。这里的是Base64隐写。

Base64隐写

借用一下他人代码：）。（这里先把一段段base64写入文件中）

```
def Base64Stego_Decrypt(LineList):
    Base64Char = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"     #Base64字符集 已按照规范排列
    BinaryText = ""
    for line in LineList:
        if line.find("==") > 0:      #如果文本中有2个=符号
            temp = bin(Base64Char.find(line[-3]) & 15)[2:]      #通过按位与&15运算取出二进制数后4位 [2:]的作用是将0b过滤掉
            BinaryText = BinaryText+"0"*(4-len(temp))+temp      #高位补0
        elif line.find("=") > 0:         #如果文本中有1个=符号
            temp = bin(Base64Char.find(line[-2]) & 3)[2:]       #通过按位与&3运算取出二进制数后2位
            BinaryText = BinaryText+"0"*(2-len(temp))+temp      #高位补0
    Text = ""
    if(len(BinaryText) % 8 != 0):        #最终得到的隐写数据二进制位数不一定都是8的倍数，为了避免数组越界，加上一个判断
        print("警告:二进制文本位数有误，将进行不完整解析。")
        for i in range(0, len(BinaryText), 8):
            if(i+8 > len(BinaryText)):
                Text = Text+"-"+BinaryText[i:]
                return Text
            else:
                Text = Text+chr(int(BinaryText[i:i+8], 2))
    else:
        for i in range(0, len(BinaryText), 8):
            Text = Text+chr(int(BinaryText[i:i+8], 2))      #将得到的二进制数每8位一组对照ASCII码转化字符
        return Text

file = open(r'bas.txt', "r")
LineList = file.read().splitlines()
print("隐写内容为:")
print(Base64Stego_Decrypt(LineList))
```