BUU_crypto刷题记录2

原创

te_mgl 于 2020-10-27 21:40:27 发布 161 收藏

分类专栏： 密码学 文章标签： 密码学

本文链接：https://blog.csdn.net/weixin_44617902/article/details/109203920

这里写目录标题

1.[NCTF2019]babyRSA

文件打开分析代码：

```
import ...

def nextPrime(n):
    n += 2 if n & 1 else 1
    while not isPrime(n):
        n += 2
    return n

p = getPrime(1024)
q = nextPrime(p)
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)

# d = 192757789460378997180354554381755091757239114661274621545069165641015199236033089003314276019834768862558492
# c = 538272316807382811069616855829420668175799114902277782112756330141348322387452723330072118083929861707670568
```


已知e,d,c,求m。看来必须求q,p了

进一步分析p,q是1024位的,因此两者相乘大概是2048位,通过运算可知ed-1为2064位,因此k一定小于16位,我们只需在0到2**16遍历即可条件为（e*d-1*）%*k*==0(*因为*ed-1=k*phi)

代码：

```
import sympy.crypto
import gmpy2
import binascii
e=0x10001
d = 192757789460378997180354554381755091757239114661274621545069165641015199236033089003314276019834768862558492003323740819964429763070585973088116815586223853301862194473329920810818581417946684450446816320036999656426592102288867006255450475851245321743477820468049493138182917270504007525517165504036471481971488844082646868466938421183872177535169634497538098603540476192567878694002978585681397003965675194698253985751038854876244634244299130177295856208771681716034441114646928413796611120751233993432706102722878652008803981935732608482686334619834350150312270702178527282408473980844146871463973031107092149130
c = 538272316807382811069616855829420668175799114902277782112756330141348322387452723330072118083929861707670568504117424741582615709658305506933739398789226276421122522703588075441745705672390913552524495793590690266567977710113011139278023750292865622570526243143195300352009393292437590211128007725520511821743674411206406942967863292325989862799714580389275398925561527314030002104065450590144278781065362652430570631666316934179720575293875559005656898673822780348746727411439825718796214079655113622053280968760686738563936774370552751168071995538074637763115646868984415087838146056099075565289944934004531352180 4
e_d_1=e*d-1
p=0
q=0
for k in range(pow(2,15),pow(2,16)):
    if e_d_1%k==0:
        p=sympy.prevprime(gmpy2.iroot(e_d_1//k,2)[0])
        q=sympy.nextprime(p)
        if (p-1)*(q-1)*k==e_d_1:
            break
n=p*q
print(n)
m=gmpy2.powmod(c,d,n)
print(m)
print(binascii.unhexlify(hex(m)[2:]))
```

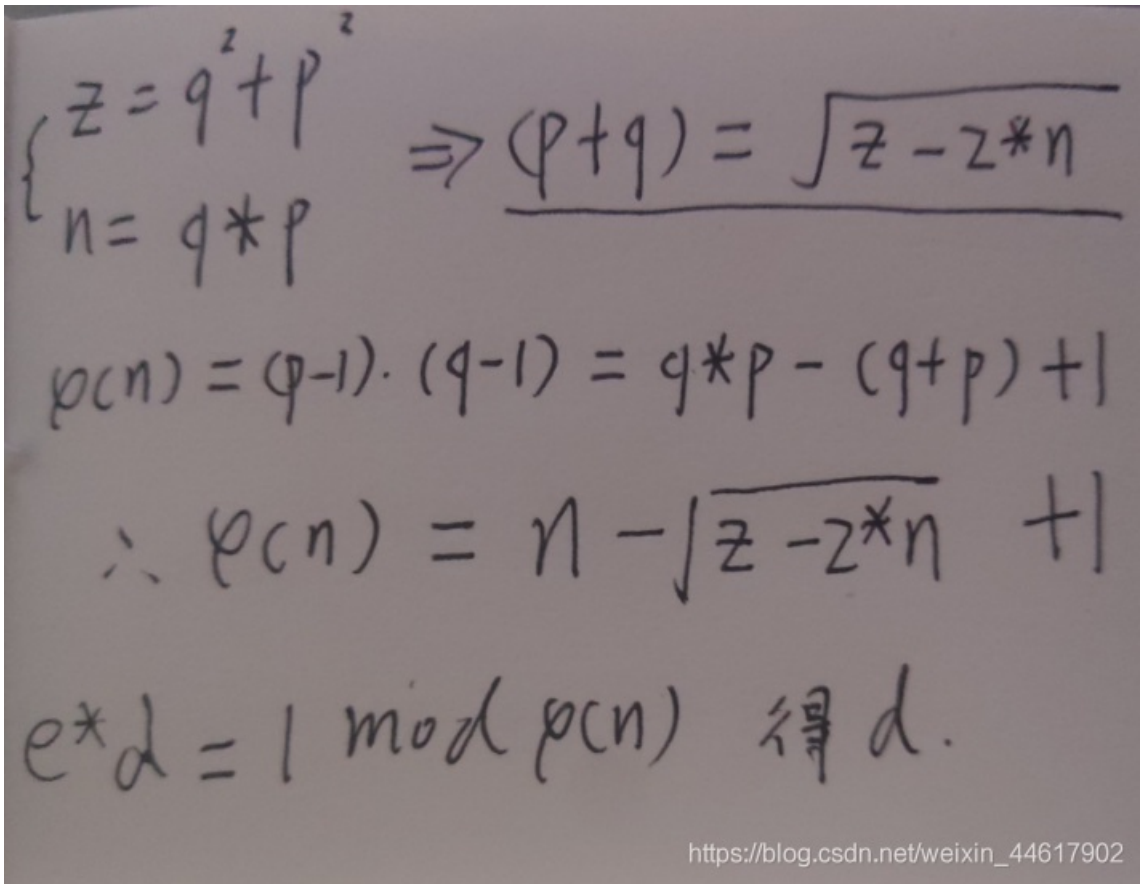
2.[BJDCTF2020]easyrsa

记录哈这个题的函数，Fraction(a,b)相当于a/b,Derivative(f(x),x))为f(x)的导函数，arctan()反正切函数，arth()反双曲函数所以最终z就等于q**2+p**2

```
from Crypto.Util.number import getPrime,bytes_to_long
from sympy import Derivative
from fractions import Fraction
from secret import flag

p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
z=Fraction(1,Derivative(arctan(p),p))-Fraction(1,Derivative(arth(q),q))
m=bytes_to_long(flag)
c=pow(m,e,n)
print(c,z,n)
'''
output:
792254786685776145980749150265421628301277617778951154935067295810181028134840228
321157486776232096674716228721852750702579247660150200728052673598390593932843165
153107451613368954134066900093247662007891792488969519420472354489016123511284593
'''
```


解题



代码：

```
import gmpy2
import binascii
e=65537
c = 7922547866857761459807491502654216283012776177789511549350672958101810281348402284098310147796549430689253803510994877420135537268549410652654479620858691324110367182025648788407041599943091386227543182157746202947099572389676084392706406084307657000104665696654409155006313203957292885743791715198781974205578654792123191584957665293208390453748369182333152809882312453359706147808198922916762773721726681588977103877454119043744889164529383188077499194932909643918696646876907327364751380953182517883134591810800848971719184808713694342985458103006676013451912221080252735948993692674899399826084848622145815461035
z = 3211574867762320966747162287218527507025792476601502007280526735983905939328431659588293337228973212727407643458751933330014247301034469480388516855754880120249593322621543776332928024211355652449845755956287290081160205694442396740377762330696188075761324632872961664303262896407293127208586692804597379937471184682515778105696516417850523252424580917923560757156717422882256169788864596855934360837533198809715714526435762673814164655635350099492411587574819831803629689860409700093827219590305673356588015054027536923963779397592332959871600335030825932143675257929100035556043154222969975995514115291470836249448 2
n = 1531074516133689541340669000932476620078917924889695194204723544890161235112845930914582554756929847982110124909416867207686537607047447968708758990950136380924747359052570549594098569970632854351825950729752563502284849263730127586382522703959893392329333760927637353052250274195821469023401443841395096410231843592101426591882573405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869343939106529204798285957516860774384001892777525916167743272419958572055332232056095979448155082465977781482598371994798871917514767508394730447974770329967681767625495394441
qaddp = gmpy2.iroot((z+2*n),2)
print(qaddp)
qaddp= 2504740285943774261118212188840619334679075975745782550661462603670945953997411968275329238367617335949769333666361492014926287084133199293610976465266521402045615425736632234690098359253099355158924584996769031491725344945805030888684306251448081890837088273633350450287029932822315378937995416851699112324422
phi = n-qaddp+1
phi=gmpy2.mpz(phi)
d = gmpy2.invert(e,phi)
print('d:',d)
m = gmpy2.powmod(c,d,n)
flag = binascii.unhexlify(hex(m)[2:])
print('flag:',flag)
```

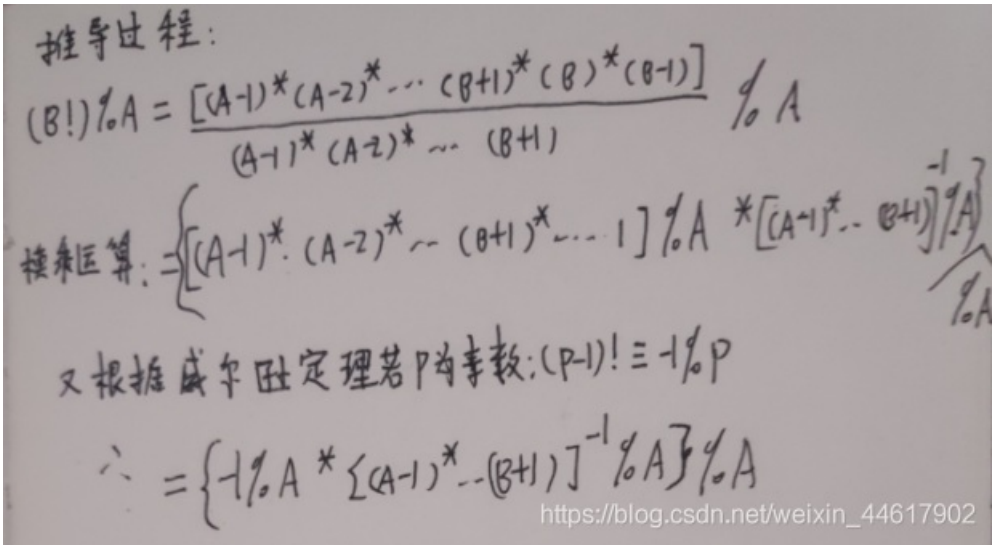
3.[ACTF新生赛2020]crypto-rsa3

直接用yafu分解n，后面就是常规的rsa解题过程了。

```
import binascii
import gmpy2
e=65537
q = 133269090503574476435265858368339693780781470577230547014328421929887176493857314300950556223035495772334957937155800048016342685057252555650215198171792 93
p = 133269090503574476435265858368339693780781470577230547014328421929887176493857314300950556223035495772334957937155800048016342685057252555650215198171792 31
n = 177606504836499246970959030226871608885969321778211051080524634084516973331441644993898029573612290095853069264036530459253652875586267946877831055147546910227100566496658148381834683037366134553848011903251252726474047661274223137727688689535823533046778793131902143444408735610821167838717488859902242863683
c = 145739037851138235477100054094536116898477505269307364168237507140749085128970307090574952583048303598873711765397142842461233202092592661739555886816038060191249829992282591422951016695791045184173002891988380763448983412883080140722844722177526471134992815629010278237437940671929211604758156053038221004 9
phi = (q-1)*(p-1)
n = p*q
d = gmpy2.invert(e,phi)
m = pow(c,d,n)
flag = binascii.unhexlify(hex(m)[2:])
print(flag)
```

4.[RoarCTF2019]babyRSA

打开文件有阶乘运算想到威尔逊定理，解题过程推到如下：



代码：

```
import gmpy2
import binascii
import sympy
A1=218569634524616304373482784341914340000660767504190274938524635134698652620643408366138310666023009597726323977734873175603390566582999544641692644672344 07
B1=218569634524616304373482784341914340000660767504190274938524635134698652620643408366138310666023009597726323977734873175603390566582999544641692644671405 96
A2=164661131158392281197678878993088200257492609338634468882241671698576121786641395457263408674067907545602275160137962699414380768181946170303048518584189 27
B2=164661131158392281197678878993088200257492609338634468882241671698576121786641395457263408674067907545602275160137962699414380768181946170303048518583510 26
n=854926637862752921598316033910838761751493543093276730087166276507181605856397231007933475346496283304166312556609013075339099004314134475242623322326591530470679086934819471210690704515628224173576564321718709511846731325542136901233080426973619699863603750609547029206563641441541458128385583653341729359314414240962702061406918146623185626969257679919373697826279084082390873580331654100206901520677157111127322520385884328967584058987090103424678822643627 33
e=0x1001
c=757008830216695777393293167954507062045026358023107314771569988347108207702452194687032453020099989320670803839775602997080604762220896302099726297559651403175260346804524833609173788122443658845271860563418886155643355607650535501557583622716223300174334030272611275612255859124847778295885012139611106904519876255027013314851416396843564273169051229957598252411338727343627160418198199486456628032924188022044308745213421084136236351504759631212200952367764 28
def jie(A,B):
    ans=1
    temp=gmpy2.powmod(-1,1,A)
    for i in range(B+1,A):
        ans=(ans * gmpy2.invert(i,A) )%A
    return (ans*temp)%A
p = sympy.nextprime(jie(A1,B1))
q = sympy.nextprime(jie(A2,B2))
r = n//p//q
phi = (p-1)*(q-1)*(r-1)
d = gmpy2.invert(e,phi)
flag = gmpy2.powmod(c,d,n)
print(binascii.unhexlify(hex(flag)[2:]))
```

5.[MRCTF2020]Easy_RSA

先读懂代码，关键在于求P，Q。

```
def gen_p():
    p = getPrime(1024)
    q = getPrime(1024)
    assert (p < q)
    n = p * q
    print("P_n = ", n)
    F_n = (p - 1) * (q - 1)
    print("P_F_n = ", F_n)
    factor2 = 2021 * p + 2020 * q
    if factor2 < 0:
        factor2 = (-1) * factor2
    return sympy.nextprime(factor2)


def gen_q():
    p = getPrime(1024)
    q = getPrime(1024)
    assert (p < q)
    n = p * q
    print("Q_n = ", n)
    e = getRandomNBitInteger(53)
    F_n = (p - 1) * (q - 1)
    while gcd(e, F_n) != 1:
        e = getRandomNBitInteger(53)
    d = invert(e, F_n)
    print("Q_E_D = ", e * d)
    factor2 = 2021 * p - 2020 * q
    if factor2 < 0:
        factor2 = (-1) * factor2
    return sympy.nextprime(factor2)
```

首先看gen_p()函数要求P必须知道p，q
由两个等式可求p，q。用sage解方程：

```
sage: P_n =  140573321395373957012384636499444468125752110876930399967995532195041112385722698230721990398122084922508482686658959347725753762078210306431342748775024099427363967328185280882426903064598376668106616623710994930151635857933293394780142635709194350912242693822450297748437956252964976682327991427626735740
....: var("q p")
....: eq1 = P_n==p*q
....: eq2 = P_F_n==P_n-p-q+1
....: solve([eq1,eq2],p,q)
```


再求P：

```
p1 = 11815357834556225055076705773138578296306373458632111257986974765000144847363386030514228150486252192824652087630070740551514144472755083906683519590592728190388030786094263032249910616419173617420150645715727222080251560793961847671659388842883296237449414772357798099266162925471311692369006782715566888957 1
q1 = 11897508595485866064256258415213926142249334853259340030796012731724951176154203045191256136268736105319137530718041393172135525189535093637678165767489680138880637975075726437739660817423507502185461432800989740882423580016736920420368093829880375296498335829829969927342559638226886923713972475421444355638 3
f1 = 2021*p1 + 2020*q1
if f1<0:
    f = (-1)*f1
P = sympy.nextprime(f1)
print('P:',P)
```

接着来求Q，已知e*d，n求p，q参考RSA已知e，d和n，分解N
脚本：

```
from random import randint
import gmpy2

def oddr(r):
    while r % 2 == 0:
        r = r // 2
    return r
def funp(e_d, N):
    k = e_d - 1
    r = oddr(k)
    while True:
        b = randint(2, N - 1)
        a = gmpy2.powmod(b, r, N)
        if a == 1:
            continue
        y = gmpy2.gcd(a - 1, N)
        if a > 1 and y > 1:
            q = N // y
            return q
        else:
            r = r * 2
n = 207142983381604497495453607436880188428772740545408520964594852839368023412713637661579761125250340043199380
540349348808609569665850516844836625357806216733167748426147017264458706301091960166767251834128798704634322776 2
991666913049404040373329559365530610417636790235248436752026291794310046769754059392570716216261663553355026271 8
808746254599456286578409187895171015796991910123804529825519519278388910483133813330902530160448972926096083990 2
082432745485612382530027894749207307600011040480932956805930333278188212553008934234121922658144185461340155575 7
923621946178034446912798766956513893030852518994489742175394 7
e_d = 100772079222298134586116156850742817855408127716962891929259868746672572602333918958075582671752493618259 5
182863361227727033301830372211050582986534907943378850984990735838218325327983095135383831752334295334673483903 8
932322519880529495048480206814859090290722115096853906798043283131037636820277321226632011267069973750105483164 6
286585142281419237572222713975646843555024731855688573834108711874406149540078253774349708158063055754932812675 7
861237007682880484453261998809837175045388254981037893048736821910530503668068258026026586742684408445779554993 6
840401911491393447716042842866284701228951665531068011963860031522828429893520 1
p = funp(e_d, n)
q = n // p
phi = n - (p + q) + 1
if p * q == n:
    print('p:',p)
    print('q:',q)
```

再求Q：

```
q2 = 171847486694659608706336923173786708071603689972942289760669690002615525263534483261477699540482615520223300
780778172120221008417518590133753701145591943840552802072474293556608389677806415392384924913911677288126066245
025731416399656855625839288752326267741979436855441260177305707529456715625062080892327017
p2 = 120538849514661970159855851547577637711900368732462953774738483480759950867244867240401273864984981385806453
735655967797329769252143125966966236767391995563418243748302685348336642872306042286401427581501609713577329945
760930395130411743322595026287853073310150103535873078436896035943385067893062698858976291
f2 = 2021 * p2 - 2020 * q2
if f2 < 0:
    factor2 = (-1) * f2
Q = sympy.nextprime(f2)
print('Q: ',Q)
```

后面就常规的rsa解密了：

```
c =  40855937355228438525361161524441274634175356845950884889338630813182607485910094677909779126550263304194796
00090438477549500094342407039633443581012653616533256541733679703661177338272834468717525308104758660283868502742829262155791451462902432479427577252201312646492699062014040641299948572875038587686811509173542557755502739403341664303264477433964465401168671663976051235335571906579522220116721983178096130822578047848246729441082854348841225876444649481523876618572845441669189885946253208343721379310482375914731761363788141978758192074515143039452671279060844296010653753912188051426983069634173750771744894696202l
e = 65537
d = gmpy2.invert(e,(Q-1)*(P-1))
m = pow(c,d,P*Q)
flag = long_to_bytes(m)
print(flag)
```

6.[MRCTF2020]babyRSA

题目代码：

```
import sympy
import random
from gmpy2 import gcd, invert
from Crypto.Util.number import getPrime, isPrime, getRandomNBitInteger, bytes_to_long, long_to_bytes
from z3 import *
flag = b"MRCTF{xxxx}"
base = 65537

def GCD(A):
    B = 1
    for i in range(1, len(A)):
        B = gcd(A[i-1], A[i])
    return B

def gen_p():
    P = [0 for i in range(17)]
    P[0] = getPrime(128)
    for i in range(1, 17):
        P[i] = sympy.nextprime(P[i-1])
    print("P_p :", P[9])
    n = 1
    for i in range(17):
        n *= P[i]
    p = getPrime(1024)
    factor = pow(p, base, n)
    print("P_factor :", factor)
    return sympy.nextprime(p)

def gen_q():
    sub_Q = getPrime(1024)
    Q_1 = getPrime(1024)
    Q_2 = getPrime(1024)
    Q = sub_Q ** Q_2 % Q_1
    print("Q_1: ", Q_1)
    print("Q_2: ", Q_2)
    print("sub_Q: ", sub_Q)
    return sympy.nextprime(Q)

if __name__ == "__main__":
    _E = base
    _P = gen_p()
    _Q = gen_q()
```

```
    assert (gcd(_E, (_P - 1) * (_Q - 1)) == 1)
    _M = bytes_to_long(flag)
    _C = pow(_M, _E, _P * _Q)
    print("Ciphertext = ", _C)
'''
P_p = 20602792684730861271967757255499114342 1
P_factor = 213671742765908980787116579976289600595864704574134469173111790965233629909513884704158446946409910475727584342641848597858942209151114627306286393390259700239698869487469080881267182803062488043469138252786381822646126962323295676431679988602406971858136496624861228526070581338082202663895710929460596143281673761666804565161435963957655012011051936180536581488499059517946308650135300428672486819645279969693519039407892941672784362868653243632727928279698588177694171797254644864554162848696210763681197279758130811723700154618280764123396312330032986093579531909363210692564988076206283296967165522152288770019720928264542910922693728918198338839
Q_1 = 10376643984946558808462504949579385763455651706456348843314822452463810597116105176312771843806286254818481474760129949405281366285145974012749955778539871448190946163199602004831579016796769993296797448448120987966417300958523146978514162898202184788394587120143015507125780316352361286311396749596957860552 1
Q_2 = 15101073427691693979059146127898148644254803503235079730649610513635872358695312348408786017643862984368846267168177751365294755532560741485851456605351324308362781068608489026112064116198761443511488756549186612050784456621056162050396120585140938604119432672843707399537232243303515351975017396063066469743
sub_Q = 16899252979359331575789599510143024199495363833091931480013053680980182497111203957256238944958435064392439198480097819370779590995647299263100429047927352511695946185622726223260008917695081072947505826033217762696128600987663034094509362995930280318966890412389099106911382624149778366699575139136102894965 1
Ciphertext =  17091872405163671414608621877494510476440948857917616735746743308408427921897950499683941222168544917579226476564309085870599970704886742203308478718118367245419076669830423762164115618266400607343070134587949250256840628045894398430272902820349996179151242318385245936070803773009851521798281995694742416786515597717633955966971402060725376881297901264720539873915382800070822030063480291257296502076613623719361967895626584587783125335059388589596445412335786543409259019639579800476391141700339365700602504389061305913779041821116222365675070227111764573014765434616005249930453007284328156720773998796682764718321
```

分析完代码主要求q，p。
求q容易就幂的模（开始我用的python自带函数pow()算不出来还是要用gmpy2库的powmod()）

```
#求q
Q = gmpy2.powmod(sub_Q,Q_2,Q_1)
q = sympy.nextprime(Q)
```

求p也简单就把P[9]左右的素数找到就好，用到sympy库的nextprime()，prevprime()。

```
#求p
P = [0 for i in range(17)]
P[9] = P_p
for i in range(10,17):
    P[i] = sympy.nextprime(P[i-1])
for i in range(8,0,-1):
    P[i] = sympy.prevprime(P[i + 1])
P[0] = sympy.prevprime(P[1])
n = 1
phi = 1
for i in P:
    n *=i
    phi *= (i-1)
d = gmpy2.invert(e,phi)
m = gmpy2.powmod(P_factor,d,n)
p = sympy.nextprime(m)
```

综合代码：

```
import binascii
import sympy
import gmpy2

P_p = 20602792684730861271967757255499114342 1
P_factor = 213671742765908980787116579976289600595864704574134469173111790965233629909513884704158446946409910475727584342641848597858942209151114627306286393390259700239698869487469080881267182803062488043469138252786381822646126962323295676431679988602406971858136496624861228526070581338082202663895710929460596143281673761666804565161435963957655012011051936180536581488499059517946308650135300428672486819645279969693519039407892941672784362868653243632727928279698588177694171797254644864554162848696210763681197279758130811723700154618280764123396312330032986093579531909363210692564988076206283296967165522152288770019720928264542910922693728918198338839
Q_1 = 10376643984946558808462504949579385763455651706456348843314822452463810597116105176312771843806286254818481474760129949405281366285145974012749955778539871448190946163199602004831579016796769993296797448448120987966417300958523146978514162898202184788394587120143015507125780316352361286311396749596957860552 1
Q_2 = 15101073427691693979059146127898148644254803503235079730649610513635872358695312348408786017643862984368846267168177751365294755532560741485851456605351324308362781068608489026112064116198761443511488756549186612050784456621056162050396120585140938604119432672843707399537232243303515351975701739606306646974 3
sub_Q = 16899252979359331575789599510143024199495363833091931480013053680980182497111203957256238944958435064392439198480097819370779590995647299263100429047927352511695946185622726223260008917695081072947505826033217762696128600987663034094509362995930280318966890412389099106911382624149778366699575139136102894965 1
Ciphertext =  17091872405163671414608621877494510476440948857917616735746743308408427921897950499683941222168544917579226476564309085870599970704886742203308478718118367245419076669830423762164115618266400607343070134587949250256840628045894398430272902820349996179151242318385245936070803773009851521798281995694742416786515597717633955966971402060725376881297901264720539873915382800070822030063480291257296502076613623719361967895626584587783125335059388589596445412335786543409259019639579800476391141700339365700602504389061305913779041821116222365675070227111764573014765434616005249930453007284328156720773998796682764718 32
e = 65537
#求p
P = [0 for i in range(17)]
P[9] = P_p
for i in range(10,17):
    P[i] = sympy.nextprime(P[i-1])
for i in range(8,0,-1):
    P[i] = sympy.prevprime(P[i + 1])
P[0] = sympy.prevprime(P[1])
n = 1
phi = 1
for i in P:
    n *=i
    phi *= (i-1)
d = gmpy2.invert(e,phi)
m = gmpy2.powmod(P_factor,d,n)
p = sympy.nextprime(m)
#求q
Q = gmpy2.powmod(sub_Q,Q_2,Q_1)
q = sympy.nextprime(Q)

d=gmpy2.invert(e,(p-1)*(q-1))
plaintext=gmpy2.powmod(Ciphertext,d,p*q)
print(binascii.unhexlify(hex(plaintext)[2:]))
```

```
b'MRCTF{sti11_@_b@by_qu3st10n}'
```

7.[BJDCTF2020]Polybius

打开文件题目很明确给了密文，和hint

```
密文：ouauuuoooeeaaiaeauieuooeeiea
hint：VGhlIGx1bmd0aCBvZiB0aGlzIHBsYWludGV4dDogMTQ=
flag:解出明文后，请加上BJD{}
```

hint用base64解密得到

```
The length of this plaintext: 14
```

与密文刚好2：1.搜索一下题目Polybius发现是一种加密方法。

信息安全加密技术–Polybius密码

密文出现aeiou五个字符，但是不知道排列顺序，选择暴力破解。

先把五个字符的所有排列 列出来按照5*5矩阵。再替换密文。

直接上脚本吧：

```
import itertools
s="aeoiu"
sumresult=[]
for i in itertools.permutations(s,5):
    sumresult.append("".join(i))
print(sumresult)
zimu = "abcdefghjklmnopqrstuvwxyz"
for h in sumresult:
    c = "ou au uu oo oe ea ai ae au ie uo oe ei ea"
    n = 0
    for i in h:
        for j in h:
            c = c.replace(i+j,zimu[n])
            n +=1
    cc = c.replace(' ', '')
    if 'flag'in cc:
        print(cc)
```

找到flag我是把i换成了j所以需要替换j为i。

```
flagjspolybjus
flagkxoplubkyx
```