PwnTheBox(Crypto篇)---Rsa(q相同)

原创

肖萧然 已于 2022-04-11 19:59:20 修改 127 收藏 1

分类专栏： MyCTF # CRYPTO 文章标签： rsa python ctf crypto

于 2022-04-10 14:29:35 首次发布

PwnTheBox(Crypto篇)—Rsa

文章目录

题目描述

```
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
'''
```

程序解析

rsa基础 --> https://blog.csdn.net/qq_52549196/article/details/123491446

```
from Crypto.Util.number import getPrime, bytes_to_long

flag = open("flag", "rb").read()

p = getPrime(1024)  # 得到一个未知素数p
q = getPrime(1024)  # 得到一个未知素数q
assert(e < 100000)  # e是小于100000
n = p*q
m = bytes_to_long(flag) # flag为m转成字节
c = pow(m, e, n)
print c, n
print pow(294, e, n)

p = getPrime(1024)   # 再次得到一个未知素数q
n = p*q    # 此时的p没有再次得到,而是与上一个相同
m = bytes_to_long("BJD"*32)
c = pow(m, e, n)
print c, n
```

拿到flag

将密文 c 解密为明文 m 后再 bytes_to_long

拿到 e

```
assert(e < 100000)
c = pow(m, e, n)
print c, n
print pow(294, e, n)
```

从pow(294, e, n)的值中可以爆破出 e 的值

```
n = 13508774104460209743306714034545670413724762734498113346180195347973601702140172581880846289837599476737562774949483967194454382240305997807381312244140761253065816894298782025678658300694700171174923019354237057095070553016792170283562712240147525103900077501738163390022247472739682370869506313624611565262225976963459130942176126954826098442614882464128501073098321537750925501129873782762161115803297642001166254785451561059795562889807356968415822567833347454392032653289344684980811283747668439003097647205390506985552229785068802696070118654342813984378390762431727479692624882954341346475412720884307033106303 7

result = 38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889859846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501 8

for e in range(1000000):
    if(pow(294, e, n) == result):
        print("e =", e)  # e = 52361
        break
```

拿到 q

比如: p*q=n

```
2 * 3 =6
2 * 7 = 14
```

由于p,q都是素数,当p相同,q不同时,n 值的最大公因数就为p

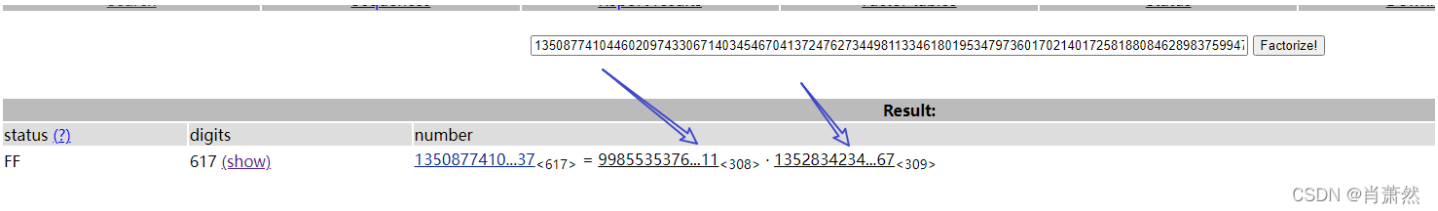
由于两个 n 用的 q 是 同一个 所以可以找到 q 的值

```
import gmpy2
n1 = 135087741044602097433067140345467041372476273449811334618019534797360170214017258188084628983759947673756274949483967194454382240305997807381312244140761253065816894298782025678658300694700171174923019354237057095070553016792170283562712240147525103900077501738163390022247472739682370869506313624611565262225976963459130942176126954826098442614882464128501073098321537750925501129873782762161115803297642001166254785451561059795562889807356968415822567833347454392032653289344684980811283747668439003097647205390506985552229785068802696070118654342813984378390762431727479692624882954341346475412720884307033106303 7
n2 = 128062109030613683690543095751593603740223447745474593452169071281939575929380718158659540732875325459473706718383721448065397538294843560649193572856233052096006805709752246392143968051243508627721592723627787680368446347609176127087217873201593184324560508062277844350911611199826139873032559955431653954266580594621100564313925175487174478980849151676611723629842512016886394696522834523077128213988570164875907949965444688267056003322085352014433222672987471175288829859553752464248126164783271823994617099788934640932451355301354300078422233893602128034398508676151211480500348877675846936087763232522332542610477
print(gmpy2.gcd(n1, n2))  # 9985535376176493930826595149211697679867468128294146251695657771294371785004805127335874509590620708517091579418774995458868585045216216505983174930347310654193094872300088271345367990452565532716866529520742325792266672107774791186015918104142299303061838543650485894361563021945926241971581636178106289891 1
```

偷懒做法

有一个神奇的网站 http://factordb.com/

能够在线 **分解 n** 所以不用努力分析了╭つ ◕_◕╮つ



直接给 n 来一手 !!! 管他怎么绕

拿到 d

公式

```
n=pq

ϕ(n)=(p-1)(q-1)

(de) mod ϕ(n) = 1
```

```
import gmpy2

n = 135087741044602097433067140345467041372476273449811334618019534797360170214017258188084628983759947673756277
494948396719445438224030599780738131224414076125306581689429878202567865830069470017117492301935423705709507055 3
016792170283562712240147525103900077501738163390022247472739682370869506313624611565262225976963459130942176126 9
548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569 6
841582256783334745439203265328934468498081128374766843900309764720539050698555222978506880269607011865434281398 4
378390762431727479692624882954341346475412720884307033106303 7

q = 998553537617649393082659514921169767986746812829414625169565777129437178500480512733587450959062070851709157
941877499545886858504521621650598317493034731065419309487230008827134536799045256553271686652952074232579226667 2
107774791186015918104142299303061838543650485894361563021945926241971581636178106289891 1

p = n//q

e = 52361

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

d = gmpy2.invert(e, phi)

print(d)  # 127654961266723549607306241368455705718189606965043922707732980305444533568678480837769092303745959 9
873467009883319010270651875121757429604270875839653269929942067949999116415472966763781703261391783789416696579 2
300393606552754260925246462375145536208432962469395986428506102815293726955295577318642841093404208833575070203 0
553104648049131790032897958204152659430517790160899574229470216725490471969476757592642238928546623447097549321 3
355894905893794399011239178089769159371839248867254033385259727103135858943287108430964746755639150797845621865 2
168736085280364906514493349967374434638080570271850221237959102333 21
```

拿到 c

将密文 c 解密为明文m： m = c^d mod n

```python
from Crypto.Util.number import *
import gmpy2
c = 126416356178037461503322326463545962927078614802002075371991411836244383037571205700967412480202366669657557
980096565477386163990253001230437662555185961493489304445998206752300464233730530516319325572308490834268594901837323037517440048741830625948568703186142899916759800635483164994869089232096275638715548756127020791005670186989929358182061090875681660973923141057175554829261410305056395717088762131671121879625844840653215457275941351753692339259225077949996073235369768241831629233850056699304034488534651414058468359198429084697875473417523654718924952043076445861613932287760420155341479138883383162441691 20

n = 135087741044602097433067140345467041372476273449811334618019534797360170214017258188084628983759947673756277494948396719445438224030599780738131224414076125306581689429878202567865830069470017117492301935423705709507055301679217028356271224014752510390007750173816339002224747273968237086950631362461156526222597696345913094217612695482609844261488246412850107309832153775092550112987378276216111580329764200116625478545156105979556288980735696841582256783334745439203265328934468498081128374766843900309764720539050698555222978506880269607011865434281398437839076243172747969262488295434134647541272088430703310630 37

e = 52361

d = 127654961266723549607306241368455705718189606965043922707732980305444533568678480837769092303745959987346700988331901027065187512175742960427087583965326992994206794999911641547296676378170326139178378941669657923003936065527542609252464623751455362084329624693959864285061028152937269552955773186428410934042088335750702030553104648049131790032897958204152659430517790160899574229470216725490471969476757592642238928546623447097549321335589490589379439901123917808976915937183924886725403338525972710313585894328710843096474675563915079784562186521687360852803649065144933499673744346380805702718502212379591023332 1

m = pow(c, d, n)  # 162542870525762214163079926812224448168785493481361188108 5


print(long_to_bytes(m))  # b'BJD{p_is_common_divisor}'
```