[De1CTF2019]babyrsa writeup

原创

mortal15 于 2021-05-27 00:39:07 发布
分类专栏： Crypto

本文链接：https://blog.csdn.net/a5555678744/article/details/117308377

总述

对于学习RSA来说，这道题无疑是很好的教材，这道题集合了许多常考的RSA的出题点，前三步解密较为简单，但是第四步有一定难度，要想做出来得费不少功夫。总而言之是一道很不错的Crypto题。

```
import binascii
import gmpy2
from data import e1,e2,p,q1p,q1q,hint,flag

n =  [201296153524917654993401129431883171805487615978613008473058271415104656196705368446345582464392303
c =  [191314326612179084702623384212996919985261577905835441567419812388221585639885202259869152345700375
f=lambda m,e,n,c:pow(m,e,n)==c
assert(sum(map(f,[p]*4,[4]*4,n,c))==4)

ee1 = 42
ee2 = 3
ce1 =  457226517863401239469608150030593225288104818413782472806428685536076921495091269628725830371424613
ce2 =  139084683323335671584691364399323259923496988912910393540076023931945440953972538974705921383523833
tmp =  864078778078609835167779565982540757684070450697854309005171742813414963447462554999012718960925081
n  =  159115815557967986147116252885083097047918375162321224104409588307260788210690504040128208962600717
assert(pow(e1,ee1,n)==ce1)
assert(pow(e2+tmp,ee2,n)==ce2)

e = 46531
n = 162785240342783648429643860624761135170679118916997899913559821210849739517383240633051906308655115548
c = 149921321409961603309673075585031172556269257774266119785183390506710130414907246168926349110309183608
hint=int(binascii.hexlify(hint),16)
assert(q1p*q1q==n)
assert(q1p<q1q)
assert(c==pow(hint,e,n))

flag=int(binascii.hexlify(flag),16)
q1=   127587319253436643569312142058559706815497211661083866592534217079310497260365307426095661281103710
q2 =  114401188227479584680884046151299704656920536168767132916589182357583461053336386996123783294932566
c1 =  262739975753930281690942784321252339035906196846340713237510382364557685379543498765074448825799342
c2 =  739559112922887664903081961668582189920483268499575772492445081297747078782226638712233472213276047
assert(c1==pow(flag,e1,p*q1))
assert(c2==pow(flag,e2,p*q2))
```


题解

第一步

最上面一层给了4组n和c，中国剩余定理估计是八九不离十了，而且这里用的lambda函数希望对python不熟悉的可以去查一下了解一下，算是一个很方便的小函数创建，也经常在Crypto的py脚本中出现。

中国剩余定理的python脚本如下：

```
import gmpy2
import math
def  merge(a1,n1,a2,n2):
    d = math.gcd(n1,n2)
    c = a2-a1
    if c%d!=0:
        return 0
    c = (c%n2+n2)%n2
    c = c//d
    n1 = n1//d
    n2 = n2//d
    c *= gmpy2.invert(n1,n2)
    c %= n2
    c *= n1*d
    c += a1
    global n3
    global a3
    n3 = n1*n2*d
    a3 = (c%n3+n3)%n3
    return 1
def exCRT(a,n):
    a1=a[0]
    n1=n[0]
    le= len(a)
    for i in range(1,le):
        a2 = a[i]
        n2=n[i]
        if not merge(a1,n1,a2,n2):
            return -1
        a1 = a3
        n1 = n3
    global mod
    mod=n1
    return (a1%n1+n1)%n1
def exCRT_getequation(a,n):
    a1=a[0]
    n1=n[0]
    le= len(a)
    for i in range(1,le):
        a2 = a[i]
        n2=n[i]
        if not merge(a1,n1,a2,n2):
            return -1
        a1 = a3
        n1 = n3
    return (a1,n1)
#a为余数列表
#n为模数列表
n =  [20129615352491765499340112943188317180548761597861300847305827141510465619670536844634558246439230371
c =  [19131432661217908470262338421299691998526157790583544156741981238822158563988520225986915234570037383
p_4=exCRT(c,n)
p=gmpy2.iroot(p_4,4)[0]
print(p)
```

注意中国剩余定理直接算出来的结果其实是**p**的**4**次方，请使用**gmpy2.iroot()**函数进行**4**次开方

第二步

首先要对ee2=3非常敏感，低加密指数一般是个突破口，再观察tmp和n的相对大小，发现tmp的3次方与n大小相当

已知

$$k*n+ce2=(e2+tmp)**3$$

那么就可以爆破k的大小来获取e2

脚本如下：

```
import gmpy2
# gcd(e1,e2)=1
# 存在整数r，s使得re1 + se2 = 1(贝祖定理)
ee1 = 42
ee2 = 3
ce1 =  45722651786340123946960815003059322528810481841378247280642868553607692149509126962872583037142461390
ce2 =  13908468332333567158469136439932325992349696889129103935400760239319454409539725389747059213835238370
tmp =  86407877807860983516777956598254075768407045069785430900517174281341496344746255499901271896092508160
n  =  15911581555796798614711625288508309704791837516232122410440958830726078821069050404012820896260071751
# assert(pow(e1,ee1,n)==ce1)
# assert(pow(e2+tmp,ee2,n)==ce2)
# c1 = m**e1 mod n
# c2 = m**e2 mod n
# (c1**r*c2**s) mod n = (m**(re1+se2)) mod n =m
# re1 + se2 = 1
for k in range(40000,45000):
    if(gmpy2.iroot(k*n+ce2,3)[0]==int(gmpy2.iroot(k*n+ce2,3)[0])):
        print(gmpy2.iroot(k*n+ce2,3)[0]-tmp)
```

根据经验来说e2一般不是一个太大的数，输出有很多，从正到负都有，我们想要的就是输出刚刚从负变正的时候对应的e2

就像这样：

```
-142076599221195476784246000569054144637180507129381420367908395519651410142661970607146017893917383752954461969149064446120630254758980664774131860231618377681092326614820163194373807022468418109013605300
-710377155803417284288371728165312625740113583757012068022620393440656531586712104026470323874470209128035199424041530265452446215913141920330068693846381112496328868733142602180565668633770276151626403500
381791429275130
710365475678432624978541123521884753400584815232352909309533217307286540987275957076293243978044460787998576881659305388802675884025070372037617470604575484849743283122809929087286460659667213715985262214207192717119845535471344728525730452934645597526177755611285396954128928131367511796988290992857907582036882790724380406556567178083132452774028891070508186311409982611472788137149892776791530695960999213106138858072817440508904245743029718133254789936719655981504442421656499887875454149575078860144284446885553507835426939532712190124605308652560101102740887893880639936353828271611881633990024761939742841391826764704304906240647709160195988216120074110013182142983753976766522712723444289968549399566402273074226808764332854828978371410956887063261185176724779737696668541712363860727241472011775918124835517105867439221939137240063218386186619584883037968679402078342416828
```

易得

```
e2=381791429275130
```

随后是解e1：

仔细观察可以发现ce1的值与n的值相差了数十个数量级，从概率统计的意义上讲，如果比n小的每个数作为结果的可能相同，那么这种事情发生的概率非常小，但是还有一种可能就是，由于e1很小，e1的42次方都比n小，从而导致模n没起效果。

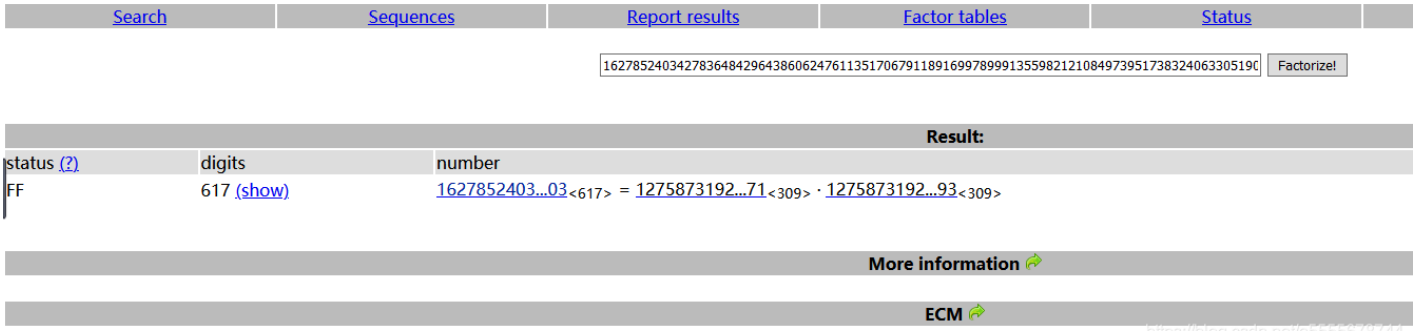
我们试着给ce1开42次方，果然直接得到一个差不多的整数，证实了猜想。

```
e1=15218928658178
```

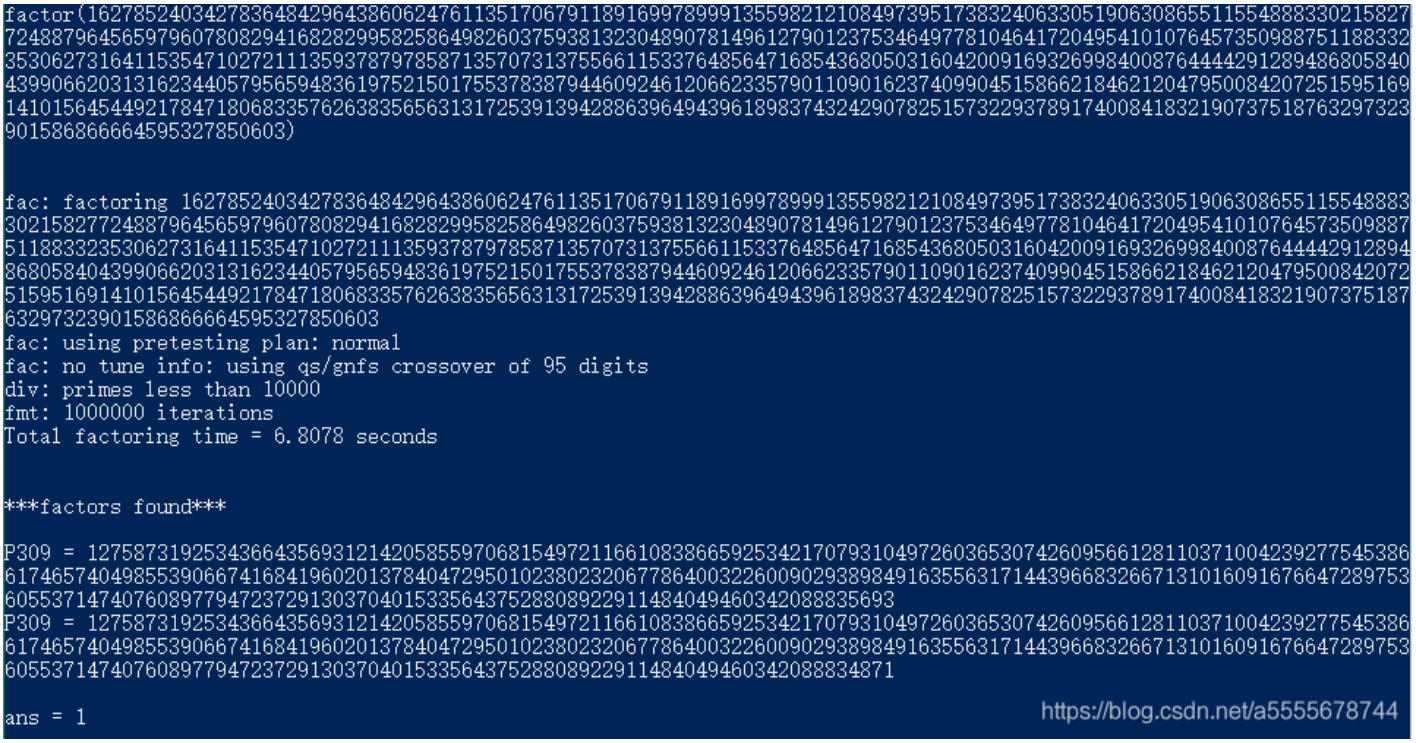
第三步

只给了e，n，c，这样一般是没办法直接得到明文的，这种情况下只能看看有没有历史分解记录，或者是暴力分解行不行

历史分解记录：



yafu暴力分解：



两个质因数差距非常小，所以能够快速解出来。

那么此题的hint也就轻松得到了

hint：

```
b'orz...you.found.me.but.sorry.no.hint...keep.on.and.enjoy.it!'
```

额，挺。。可爱的。。

第四步：

前面三关其实都好过，但是这第四关是真的鬼门关，搞了半天最后还是只能依靠其他师傅的wp，不过总算是整明白了，以人话跟大家大概说说是怎么回事吧。

先上代码，大家对着代码看解释：

```
import gmpy2
from Crypto.Util.number import *
q1=   12758731925343664356931214205855970681549721166108386659253421707931049726036530742609566128110371004
q2 =  11440118822747958468088404615129970465692053616876713291658918235758346105333638699612378329493256656
c1 =  26273997575393028169094278432125233903590619684634071323751038236455768537954349876507444882579934219
c2 =  73955911292288766490308196166858218992048326849957577249244508129774707878222663871223347221327604709
e1=15218928658178
e2=381791429275130
p=10993585793386782972898539856323545548112030085931142176254085876272195503831011760945676333808223790700 5
n =  [ q1, q2]
a=gmpy2.gcd(e1,(p-1)*(q1-1))
b=gmpy2.gcd(e2,(p-1)*(q2-1))
#a,b相同
c =  [ gmpy2.powmod(c1,gmpy2.invert(e1//a,(p-1)*(q1-1)),q1), gmpy2.powmod(c2,gmpy2.invert(e2//b,(p-1)*(q2-1
M=n[0]*n[1]
m=[0]*2
t=[0]*2
x=0
for i in range(2):
    m[i]=M//n[i]
    t[i]=gmpy2.invert(m[i],n[i])#t[i]是m[i]的模n[i]逆元
    x+=(c[i]*t[i]*m[i])
x=x%M
print(x)
#x=358048329943239557365931775343557832763798535764670058924479535376357371533613925776320773576418002758 36
e=7
d=gmpy2.invert(e,(q1-1)*(q2-1))
flag=gmpy2.iroot(gmpy2.powmod(x,d,q1*q2),2)[0]
print(long_to_bytes(flag))
```

本来这个问题p，q1，q2，e1，e2都知道，解flag出来是分分钟的事，不过一旦e1，e2和phi1和phi2不互质了问题就复杂起来了，因为这样就没法正常地求私钥d了。

一般来说遇到这种情况都是让e除去其与欧拉函数的最大公约数，让这两个数重新互质，然后求m**gcd（e，phi）的值。

这个问题碰巧a=b=14,那么最后就求出来了m**14相关的两个式子，这个幂次还是有点高，不好处理。

这里就有很巧妙的一招可以把复杂度降下来：

两个式子其实可以用中国剩余定理求m**14，为了使求出来的数尽可能小我们采用：

m14 ☰a1 mod p
m14 ☰ a1 mod q1
m14 ☰ a2 mod p
m 14 ☰ a2 mod q2

然后用求出来的(m**2)**7%（q1*q2）组成一个新的RSA解密，e=7，n=q1*q2

但是用这个实际操作的时候就发现

$$gcd((p-1),7)! = 0$$

那么可以直接用第二和第四两个式子求出x(m**2)

随后常规RSA解密后开方，即可得到flag

```
FLAG:de1ctf{9b10a98b-71bb-4bdf-a6ff-f319943de21f}
```