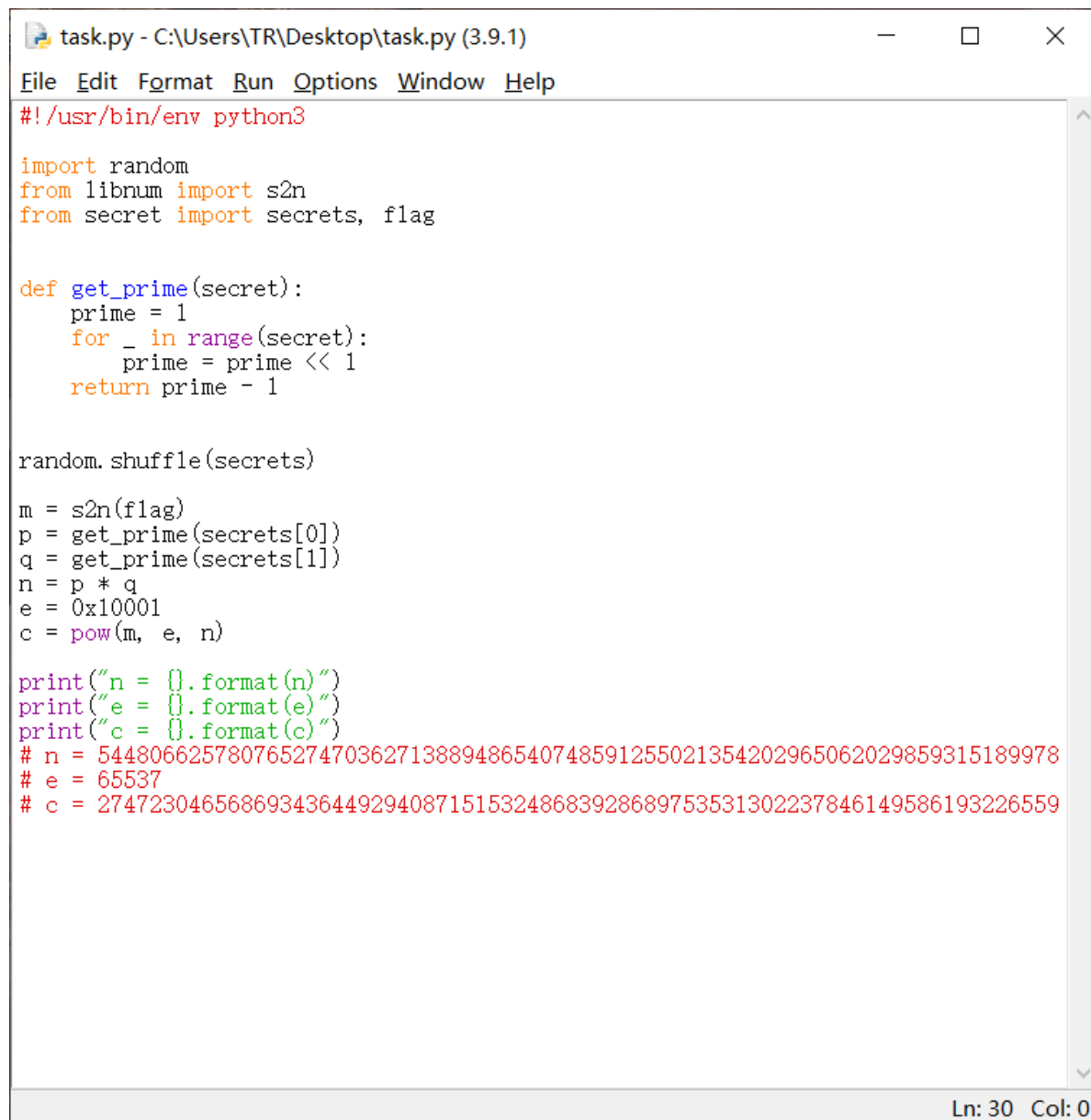


20-Web-Week3-Writeup

Crypto

LikiPrime

1. 打开题目给的文件之后得到了n、e、c



```
task.py - C:\Users\TR\Desktop\task.py (3.9.1)
File Edit Format Run Options Window Help
#!/usr/bin/env python3

import random
from libnum import s2n
from secret import secrets, flag

def get_prime(secret):
    prime = 1
    for _ in range(secret):
        prime = prime << 1
    return prime - 1


random.shuffle(secrets)

m = s2n(flag)
p = get_prime(secrets[0])
q = get_prime(secrets[1])
n = p * q
e = 0x10001
c = pow(m, e, n)

print("n = {}".format(n))
print("e = {}".format(e))
print("c = {}".format(c))
# n = 54480662578076527470362713889486540748591255021354202965062029859315189978
# e = 65537
# c = 27472304656869343644929408715153248683928689753531302237846149586193226559

Ln: 30 Col: 0
```

2. 于是利用[网站](#)分解模

Search	Sequences	Report results	Factor tables	Status	Downloads	Login
<div>544806625780765274703627138894865407485912550213542029650620298593151899783966056690901499</div>						<div>Factorize!</div>
Result:						
status (?)	digits	number				
FF	2612 (show)	5448066257...37 <2612> = 1907970075...91 <1281> · 2855425422...07 <1332>				
More information						
Listed as or is factor of listed special form						
--- none ---						
Algebraic factors						
2^4423-1 <1332>						
2^4253-1 <1281>						
Is factor of:						
(2^9689-1)^2(2^4423-1)^2(2^4253-1) <5529>						
(2^9941-1)^2(2^4423-1)^2(2^4253-1) <5605>						
Others:						
Create time						
Before November 4, 2018, 12:20 am						
ECM 						

3. 把分解得到的n、p、q、e、c代入代码运行得到flag

```
import gmpy2
import binascii

n =
54480662578076527470362713889486540748591255021354202965062029859315189978396605
66909014990944243667121934730267489179347110584556437961995117277785655935072352
10471103749410209708428398554921045548457596717901688849110662006431551066171932
41663991199835668973820397856942107803308420028326687835768515214777667811777301
84118498363548686118560688296238480430905193595405961700128815181081343064720012
09554224511908523620331293629994567633160923078515324787126043639972293248909282
31743561589383273771046819398109277725236276142862880607175974306347753022257321
68627610067560133580625728982295047579308641098722187252431318882430828324755964
14341055382043397510388566153178797955354433740194210175735661605334278427963826
45257612906381492293282290091167114882841499126596308310511811133517609760659106
62007716689271971091481689237703779521406001541275618678202267875909401859492397
91079405614967994735255149734157657559884622856408371334371327945887694841157583
45488712834389335879569948749343202759878091651313990065628228749899051654855861
90933144327823010135060087016313346488684414100985186285654232311076126919516379
32929961651079801802627277235114289178164922207502776459622987767976004949513336
67625344584228921383503930551975304974574547999422721901663712035432220321550490
24090390710036052902064162308087692386918527919568126221866883986473750573671020
66074869115310255610942615484688632815636854194831122945220666215456542098196307
29109439974566550367158285111558035205190151293969910755432001132582172707112666
59255886714938820191456489562716156543164403219053802212312765761900339403703133
72783531071424333120440758646896149373089526503999343561782359884923387763614060
63121639109865757863881574108696140718725257319281225590679935121087856822217023
32733281574520542008914729209077239374901359663637027537059976976910914303558245
35210953592699998553550797879973463843206032130110385893590948147419089764497483
50484578664207026761891970239156797131863132566708791641536507750329437545587756
64802256638829907634379439983164158483265098835872284899637347878548819596838513
24638822085341902685051299305175500860101304573399058668299312927574701741002443
90627674854278375731265389589854644671154749136163469076240220156880259297040239
01048764086196802153415204926780478064298833052055627451253647946021068705815066
81757893277071319158953845175078078660179608196467657260838076736379038608564972
66318537119000663680614845720866964513910465439613131483104774794192030552306409
71182532649432431514480233745395878459081809539376448027105464977578218874490130
7027663871874871746470590469320230424450955567169537

p = gmpy2.mpz(2**4253-1)
q = gmpy2.mpz(2**4423-1)
e = gmpy2.mpz(65537)
phi_n = (p-1)*(q-1)
d = gmpy2.invert(e, phi_n)
```

```
C =
gmpy2.mpz(2747230465686934364492940871515324868392868975353130223784614958619322
65596585247262668570469354159917080211702308916965347523963081589482678834480055
19437899038324466800260210112385442078460903737669349655257050264668557967019304
56816447600337171207404763185935638131422027579737022390476415095604731883855143
6803476580414755539511736976425888570287326879454013818994639479102268406402310
79777127630105054037389923097572091620103615856004008592356652580555453040868030
44578482396308481246784692042201810244836494068180908747532220264804500019436931
05807712298143835223010514248635965907808742142274182694993665388961156462152635
29622610945094596111978549423398090993404775464170357473041015202168599170153336
92830074762521596335176301051252876884312164901291398051157669729305382198068677
07287139630153217683188003090776192007793015105709484640342817567244702177576098
23517195785724018552575622491339480846129060936801610515554076729842184408449667
68906294766977545875699525312078595752518782730038477411233741832248694580584615
50996125400190354002896720959333447900666905073781545087820253667349425331521754
03297643351355016367468130559106542673256774178884259675887894818782003833813500
37822494324414556426263062299310297854297886057151698168811002608704427494270398
81028574011342208187930512146549728462472938448692404226977478314898800442770232
20759805494568265296703349366844815537462967716048687316896774255186618611125542
21348792878903397684855860863804648153355939388600998198808118398319659223657792
97765474981110108781936527757357239555683829958047450274021835014495062065908776
01264648843456420085104988546369727312461256933149887665622038163320029335032434
37889863070276727879267986089985706383225156589922164389585151874961615612334049
40442721057599633807154733219550696077400449101170712492068962996226976825707960
09979231639024949641531925642671892612175509176257755874660881922500484568695515
77722122130117288496516081987452433117295265880011964262689100523055585126630282
17941976930796771184504406989715748422158412810397418240755388517215486585714147
30651174968010005825597947800413565132132920421156725241007781257610670356456198
86377867254247369267789140221428188031625458572695412941813385401518593038753855
90018520548994306598318778978696343040898040428909405086133676918101054783091879
35398788782894721886082746696672680602366985718301192199044918403911149757344481
94634476277255823635122615871665697144074660363358742180644828413968242962580182
82131665158321569398245274315943337003974389707221685509568884412369874529745849
1116059452828366633503448889148896657676191573646532434154)
```

```
m = pow(c, d, n)
m_hex = hex(m)[2:]
print("%s"%(binascii.a2b_hex(m_hex).decode("utf8"),))
```

jupyter happy newyear 最后检查: 1 小时前 (未保存改变)

File Edit View Insert Cell Kernel Widgets Help 可信的 Python 3

```
In [6]: 1 import gmpy2
2 import binascii
3
4 n = 5448066257807652747036271388948654074859125502135420296506202985931518997839660566909014990944243667121934730267489179347110584556
5 p = gmpy2.mpz(2**4253-1)
6 q = gmpy2.mpz(2**4423-1)
7 e = gmpy2.mpz(65537)
8 phi_n = (p-1)*(q-1)
9 d = gmpy2.invert(e, phi_n)
10 c = gmpy2.mpz(274723046568693436449294087151532486839286897535313022378461495861932265596585247262668570469354159917080211702308916965
11
12 m = pow(c, d, n)
13 m_hex = hex(m)[2:]
14 print("%s"%(binascii.a2b_hex(m_hex).decode("utf8"),))
```

hgame{Mers3nne`Pr!Me`re41ly_s0+50~li7tle!}

In []: 1

4. 得到flag为

```
hgame{Mers3nne~Pr!Me^re411y_s0+50-1i7tle!}
```