CTF-CRYPTO-RSA-公约数

原创

大熊何在 于 2020-11-26 21:41:22 发布

分类专栏： CRYPTO

本文链接：https://blog.csdn.net/zippo1234/article/details/110205449

CTF-CRYPTO-RSA-公约数

每天一题，只能多不能少

RSA-公约数

题目分析

1. 公约数
2. 寻找素数结果

开始

1.题目

```
#!/usr/bin/python
#encoding=utf-8
import gmpy2
from libnum import n2s,s2n
from Crypto.Util.number import getPrime

e = 0x10001
flag = 'flag{******}'
m = s2n(flag)
p = []
f = open('output.txt','w')
for i in xrange(5):
    p.append(getPrime(512))
for i in xrange(5):
    for j in xrange(i+1,5):
        n = p[i]*p[j]
        f.write(str(pow(m,e,n))+'\n')
```

2.分析

加密过程一目了然。先生成5个素数，然后两两相乘得到10个n。用这10个n分别去rsa加密m得到10个c。给出了这10个c。

（1）原理

$$\because c1 = m \bmod n1 = m \bmod (p1 * \quad p)$$

（2）尝试求所有p

根据以上原理分析，我们尝试求所有p，因为这里公约数不是100%能与p相同，所以我们用素数这个要素来筛选。

```
def get_prime(k):
    for i in range(0, len(k)):
        for j in range(i + 1, len(k)):
            m = gcd(k[i], k[j])
            if isPrime(m):
                return m
    return 0
```

这里k是

```
r1=[]
r1.append(c1-c2)
r1.append(c1-c3)
r1.append(c1-c4)
r1.append(c2-c3)
r1.append(c2-c4)
r1.append(c3-c4)
k1 = get_prime(r1)
print "k1 = ",k1
```

即列出所有可能，两两求公约数，然后找出其中的素数。

脚本

先求出所有p

```
#!python2
```

```python
# -*- coding: utf-8 -*-
# @Time : 2020/11/25 0:28
# @Author : A.James
# @FileName: exp.py
import gmpy2
import libnum
from libnum import gcd
from Crypto.Util.number import isPrime

#TODO：循环计算公约数并判断是否是素数
def get_prime(k):
    for i in range(0, len(k)):
        for j in range(i + 1, len(k)):
            m = gcd(k[i], k[j])
            if isPrime(m):
                return m
    return 0

c1 = 113598680014922809576865961009282682822329588900929478733255484898360407167359037836875084852085232771180340
319748297307412686673425408790796917814413830979503218234246586144602744474931405799584571569462467391712360256691931198647999657373893180684126430938023813536134159369287097097499905821513067831432381
c2 = 621224823552037005874224033654197530219285444064830803159574668786999102458553927623919781333270208295355878645507327677814593801490846095954997971754791301903398436277296352345367783554265900283634361528287030755857988313200612249066337780274124010564030153911219960769943188451726559441134872274291957069025
c3 = 329955060886135268236264440079973297491096649391552259271201223338872763042257744842820842468302604805369088915495780633666861510958216927485946651302513287046617134006988580989155838366258729183836631951370371020957062292820489234391973594012250571937206763566357893251825065592082549856473787035103753839 6
c4 = 617626238920289619498381182486954481456133648491003026008201553298366450583534747175513048365576887348546295794699491058355766321838651769549697654288322331404129369696756065066144589481638558097056764474290259362682341338386343449788193469670129193522294887572931014982508439432230183496985571633534089667 59
c5 = 957456496596743389087978964678262721266609651166394848151749018821708545153079622805345250121387840592587515454839466706231983894621383770414630775544529394811636336962912197149473525413285164493787210517245875512371055681840310440101014066940328236883600298516675590405911163871415128440489604256865968706776
c6 = 263064583533083822273523816306816699407162393800404020439478677481669800032749006006831951207749828203411370166316858403090145975961336441638527514805945454713551202527679017941343703733943494207547196727040575971702344886098282794871594124238931336203290527639418551818022976859856195199392316860699687841177
c7 = 456631306280119053196685771226378312462354045295532050606973683930619646073675150381593048974851194332641653776305017937766821272149209488132637398230824680863937326559264320896201787465915077955593624693889993079530470844075224985954265946064265762368608024009620964248190952181124947987953440844429244811259
c8 = 900274214828851570167360351215857273069699134790731836648159008723343103833793058098752574456472677870126164059408254778291840452780885547864151572646715496453165537445500741810513010519507759009084938964355583977489476547048329206681089373672095336379069359577394393399516406647509573727569385676656782580657
c9 = 711438830584508809073026945416222870988556886412870510788338244339682568006191163357689147906053596289043592177194256212378849937243821683770866585673166268521375723928571770703381316072453295901454415816925468083152298686857146377189667536479768313355561504000245454892063665030183685462934255630560094973 5
c10 = 656799555913200523988377613749416547338394558139079366748734687054383347481878858375756813509372394458708434043470982141867287685248387847624859416063224300335154371743205396409437885369831877212140894547573362970608674127299818617265365482452867407298159125963017366855351159173297448666941783008931991487635

e = 65537

r1=[]
r1.append(c1-c2)
r1.append(c1-c3)
r1.append(c1-c4)
r1.append(c2-c3)
r1.append(c2-c4)
r1.append(c3-c4)
k1 = get_prime(r1)
print "k1 = ",k1
```

```
r2=[]
r2.append(c1-c5)
r2.append(c1-c6)
r2.append(c1-c7)
r2.append(c5-c6)
r2.append(c5-c6)
r2.append(c6-c7)
k2 = get_prime(r2)
print "k2 = ",k2

r3=[]
r3.append(c2-c5)
r3.append(c2-c8)
r3.append(c2-c9)
r3.append(c5-c8)
r3.append(c2-c9)
r3.append(c8-c9)
k3 = get_prime(r3)
print "k3 = ",k3

r4=[]
r4.append(c3-c6)
r4.append(c3-c8)
r4.append(c3-c10)
r4.append(c6-c8)
r4.append(c6-c10)
r4.append(c8-c10)
k4 = get_prime(r4)
print "k4 = ",k4

r5=[]
r5.append(c4-c7)
r5.append(c4-c9)
r5.append(c4-c10)
r5.append(c7-c7)
r5.append(c7-c10)
r5.append(c9-c10)
k5 = get_prime(r5)
print "k5 = ",k5
```

得到

```
k1 =  87473560548632075515074004799782653571992673783765777489442950981140112724762018078716308755086676032903223374630295747308121720627286318966951890649982 99
k2 =  0
k3 =  0
k4 =  98833398844392665663802928920222910278494418202546534295672136273732284730658726765568020422945681440992692411263382472687619625713602335769760245738441 81
k5 =  0
```

得到p1和p4
由此可以直接解密m

```
#!python2
# -*- coding: utf-8 -*-
# @Time : 2020/11/25 0:28
# @Author : A.James
# @FileName: exp.py
import gmpy2
import libnum

c3 = 32995506088613526823626444007997329749109664939155225927120122333887276304225774484282084246830260480536908889154957806336668615109582169274859466513025132870466171340069885809891558383662587291838366319513703710209570622928204892343919735940122505719372067635663578932518250655920825498564737870351037538396
e=95537

#TODO：K1和k4，相乘为n3，解密c3得到m。
n3 = k1*k4
phi3 = (k1-1)*(k4-1)
d3 = gmpy2.invert(e,phi3)
m = pow(c3,d3,n3)
print libnum.n2s(m)
```

4.get flag

```
flag{c0mm0n_divi20r}
```

结语

这题困在素数这个环节，折腾了很久，因为求出的公约数不是素数又一直被我拿来求n解密c，始终是乱码。后来想到求出的公约数不一定是素数，增加了一个验证素数的环节，就发现来来回回只有p1和p4可以求出来，那接下去就是常规流程了。
希望有大神能有更快速更直接的解法，在评论区指教一二。