BUUCTF_Crypto题目：rsa2

原创

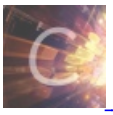
好想变强啊 于 2022-03-18 16:31:59 发布 1121 收藏

分类专栏： BUUCTF刷题记录 文章标签： python 网络安全

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_38798840/article/details/123576840

版权



BUUCTF刷题记录 专栏收录该内容

4 篇文章 0 订阅

订阅专栏

BUUCTF刷题Crypto篇

文章目录

前言

一道很简单的已知n和e，求RSA私钥d的题目。可以借助在线网站分解n得到p和q的网站分解n得到p和q的值，接下来就是各种套用RSA理论知识里的各计算式。想要记录一下这道题的原因在于最后计算结果的过程中有一点小坑，涉及了Python2和Python3在输出方面存在的区别。

一、原题

打开题目文件如下，已知n和e，拿到flag需要求出d再计算md5的结果。可见这是一道求解步骤比较少的题目，甚至都不涉及RSA中的c和m，只需要求出私钥d，然后再按照题目给的式子去计算md5即可。

rsa2.py

```
N =
101991809777553253470276751399264740131157682329252673501792154507006158434432009141995367241962525705950046253400188884658262496534706438791515071885860897552736656899566915731297225817250639873643376310103992170646906557242832893914902053581087502512787303322747780420210884852166586717636559058152544979471
e =
467319195632657213071051804103025186766761355097379929126250929768490752621920925493230823675182643786305433382190257448209164719136960720502919906204865817194103543851217607613742293748476951482305960054099783833697403058160827702839096119563559721818480775199209220592683769588117133651069252352182651730085

import hashlib
flag = "flag{" + hashlib.md5(hex(d)).hexdigest() + "}"
```

CSDN @好想变强啊

二、解题步骤

1.分解n进而求解出d的值

借助在线网站分解n得到p和q的网站，对大数n做质因数分解，得到p和q，然后根据RSA的理论知识，已知e，求解d还需要计算(p-1)*(q-1)，然后调用gmpy2模块的invert函数就可得d。

2.根据原题处理d值并输出

这里就是按照原题给的计算式去处理d值，这里有个小坑，就是由于Python2和Python3有一点区别，在Python3中，hex(d)得到的值在输出形式上相比Python2少了一个末尾的L，再用这个值去做hash得到的md5值也就不同了。经实验发现，正确的结果可以用Python2直接得到，或在Python3中做hash时给参数末尾加上L。
给出Python3的代码如下：

```python
import gmpy2
import hashlib
from Crypto.Util.number import *
n=101991809777553253470276751399264740131157682329252673501792154507006158434432009141995367241962525705950046253400188884658262496534706438791515071885860897552736656899566915731297225817250639873643376310103992170646906557242832893914902053581087502512787303322747780420210884852166586717636559058152544979471
p=90468539152235033517870318889776271069345640432047835931186781819915963165828770575564631525796216990106105695265730319547795207814485506777675652074071 83
q=11273732364123571293429600400343309403733952146912318879993851141423284675797325272321856863528776914709992821287788339848962916204774010644058033316303937
phi=(p-1)*(q-1)
e=46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383369740305816082770283909611956355972181848077519920922059268376958811713365106925235218265173085
d=gmpy2.invert(e,phi)
print(d)
print(hex(d))
#hex(d)='0x13b8f87d588e2aa4a27296cf2898f56ab4c8deb5a1222ec080e23afecaf7f975'

''' 但python3中hex(d)得到的数字最后没有L，导致hash值与python2得到的不同，而正确的flag是用python2得到的结果，所以这里我手动加上了L'''
print(hashlib.md5(b'0x13b8f87d588e2aa4a27296cf2898f56ab4c8deb5a1222ec080e23afecaf7f975L').hexdigest())
```

运行结果如下：

输出的三行分别是d，hex(d)，和最终的flag{}内的字符串

```
892075899541458715282942655858002565735732874583974769373959182028353830744 5
0x13b8f87d588e2aa4a27296cf2898f56ab4c8deb5a1222ec080e23afecaf7f975
47bf28da384590448e0b0d23909a25a4
```

总结

以上就是这题要记录的全部内容了，做题的过程中要多多尝试~