RSA-GENERATOR

思路为已知p高位攻击。

本题思路和这篇博客的差不多：<https://www.cnblogs.com/WangAoBo/p/7541481.html>

这道题其实是简化版。

首先利用openssl提取私钥 命令：openssl rsa -in public.pem -pubin -text -modulus

提出n和e。

分析已知条件：

n="0x639386F4941D1511D89A9D19DC4731188D3F4D2D04623FB26F5A85BB3A54747BCBADCDBD8E4A75747DB4072A90F62DCA08F11AC276D7588042BEFA504DCD87CD3B0810F1CB28168A53F9196CDAF9FD1D12DCD4C375EB68B67A8EFCCEC605C57C736943170FEF177175F696A0F6123B993E56FFBF1B62435F728A0BAC018D0113"

c="56c5afbc956157241f2d4ea90fd24ad58d788ca1fa2fddb9084197cfc526386d223f88be38ec2e1820c419cb3dad133c158d4b004ae0943b790f0719b40e58007ba730346943884ddc36467e876ca7a3afb0e5a10127d18e3080edc18f9fbe590457352dca398b61eff93eec745c0e49de20bba1dd77df6de86052ffff41247d"

p的前279位：

754000048691689305453579906499719865997162108647179376656384000000000000001232324121

根据wiki上Factoring with High Bits Known¶理论 要p的前288位才可以，所以需要爆破那九位，上sage脚本：

n=0x639386F4941D1511D89A9D19DC4731188D3F4D2D04623FB26F5A85BB3A54747BCBADCDBD8E4A75747DB4072A90F62DCA08F11AC276D7588042BEFA504DCD87CD3B0810F1CB28168A53F9196CDAF9FD1D12DCD4C375EB68B67A8EFCCEC605C57C736943170FEF177175F696A0F6123B993E56FFBF1B62435F728A0BAC018D0113

p = 754000048691689305453579906499719865997162108647179376656384000000000000001232324121

for i in (0,1):

for b in range(0,2\*\*9):

pp=p << 9

pp +=b

pp=hex(pp)+"00000000000000000000000000000000000000000000000000000000"

pp = int(pp, 16)

p\_fake = pp+2\*\*10

pbits = 512

kbits = pbits-288

pbar = p\_fake & (2^pbits-2^kbits)

print "upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)

PR.<x> = PolynomialRing(Zmod(n))

f = x + pbar

try:

x0 = f.small\_roots(X=2^kbits, beta=0.4)[0] # find root < 2^kbits with factor >= n^0.4

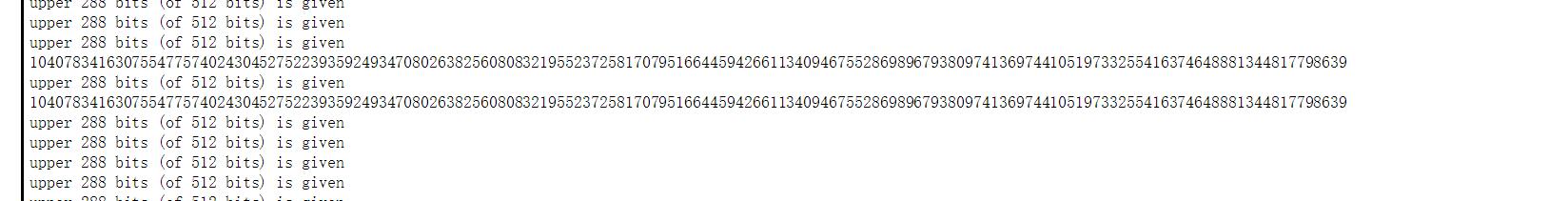
print x0 + pbar

except:

pass

去这个sage在线运行：<http://sagecell.sagemath.org/>

看到跑出了p:



得到p 直接有了 p q，直接计算私钥d 解密 脚本：

import gmpy2

n="0x639386F4941D1511D89A9D19DC4731188D3F4D2D04623FB26F5A85BB3A54747BCBADCDBD8E4A75747DB4072A90F62DCA08F11AC276D7588042BEFA504DCD87CD3B0810F1CB28168A53F9196CDAF9FD1D12DCD4C375EB68B67A8EFCCEC605C57C736943170FEF177175F696A0F6123B993E56FFBF1B62435F728A0BAC018D0113"

n=int(n,16)

c="56c5afbc956157241f2d4ea90fd24ad58d788ca1fa2fddb9084197cfc526386d223f88be38ec2e1820c419cb3dad133c158d4b004ae0943b790f0719b40e58007ba730346943884ddc36467e876ca7a3afb0e5a10127d18e3080edc18f9fbe590457352dca398b61eff93eec745c0e49de20bba1dd77df6de86052ffff41247d"

c=int(c,16)

p=10407834163075547757402430452752239359249347080263825608083219552372581707951664459426611340946755286989679380974136974410519733255416374648881344817798639

q=n/p

e=65537

d=gmpy2.invert(e,(p-1)\*(q-1))

result=pow(c,d,n)

print hex(result)[2:].decode('hex')

flag:

