CTF-CRYPTO-RSA-diffwiener

原创

大熊何在 于 2020-12-11 00:10:00 发布 135 收藏 1

分类专栏： CRYPTO

本文链接：https://blog.csdn.net/zippo1234/article/details/111026309

CTF-CRYPTO-RSA-diffwiener

停更好几天了。。其实没闲着，只是题目都太难了，实在是没脸上来丢人。

diffwiener

题目分析

1. variant of Wiener’s Attack
2. Wiener’s Attack

开始

1.题目

diffwiener.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Util.number import *

flag = "flag{******}"
m = bytes_to_long(flag)

p = getPrime(1024)
q = getPrime(1024)
r = getPrime(1024)
n = p * q * r
phi = (p-1) * (q-1) * (r-1)

d = getPrime(512)
e = inverse(d, phi)
c = pow(m, e, n)

print "n = %s" % str(n)
print "e = %s" % str(e)
print "c = %s" % str(c)

# n = 24200322442558234307996132315038234291566474519323322540031775165952710648103551437950063774775381153977868
# 19513233233891992714238478285183494458404967155706588646444836564539258000370429434888887831091362276690532389 51
# 83517166684050749514166506919963464578828710127415058556721750312679968361404291172479016184720986271746746194 59
# 49107387775430069246118314177717610531093535852860996248191605933159234361306205833438466966784573208607058388 83
# 25609113265414717089737013875577060097649176234431736731854096695220295890371264786203975647918297017058442163 01
# 09314805541595107467920368677464138395140713150840889483059672487008731260933948853067331225426140389200258623 12
# 50710804852777662734888713129347872342984947294421926566652570406674229942752756313091331710461527286965509797 92
# 71784837069973776297037765897347226852354921556918673813185503066023386090831686691775666120647160826368008908 82
# 62866591158838489966111390085518 19
# e = 32963234673870704100445517346726497971982993497766509899224202662312397669756300349164499114559175218253229
# 18669331161532542723271163876138350025858491883018527791261271089898703902816740374648288153415059615836352676 14
# 47467237298059458062763728934113200075225201456826996881146381342779410067107410327501722012036389269324184712 12
# 89563622646395762668425076722455637817120950630633722390280553804748481650506569764104758495533066897148648068 43
# 34393141728131358672441794422066934521972147691809225779464650697583752658520190362020539191384025827390998024 07
# 44857368033412917998560674588220531279826739195334643103706381118877150714082841558545758189124566868829356678 70
# 80896071754120378329235576974724409288797172524231737127529149489036118180171573115049098326837778732182598825 75
# 51602078107176576376818206741673654262292499539955431517752626603691426882652618324898471143048885404047416764 57
# 14482331733098641587043972 1198187
# c = 77946857064750780999513551023189412264862767880123382013538836593457179168709582111830183257444952761894680
# 94296588676758733896666291065368985138754818925993166888217931023379741388469219763374665147931844366735284578 40
# 23804467579655634728482805128504881806679171647745950343416691608521141456599299657881476306864288949256982779 96
# 96358101731189151371675026116225708067971048934199550535621493275739660498890452752788152569922681675081117258 74
# 68012902395912366750429698851498492941778231469792547055140436676384252222473196755614344923802547933552445240 52
# 84809905275847072503884676147156002291668439697185689523907470163991559393734652288002322985886758283330035285 52
# 36518879515729324540951852578285878189714979510284763555829862844960870761107862675158580382039723214654254061 08
# 05030544269106363601607123674434305441200230935215097519699208684015784681438517600014766080990945582887888051 05
# 67691467922078058541096968581 0613
```

2.分析

P.S：这题还是有点玄学的。。。正常情况下出题人的意图应该是要我们用variant of Wiener's attack但是用普通的wiener's attack也可以。。。尴尬。只能说是d的位数选择有问题。这里放一下ASIS的原题

```
def make_pubpri(nbit):
    p, q, r = [ getPrime(nbit) for _ in xrange(3)]
    n = p * q * r
    phi = (p-1)*(q-1)*(r-1)
    l = min([p, q, r])
    d = getPrime(1)
    e = inverse(d, phi)
    a = gensafeprime.generate(2*nbit)
    while True:
        g = getRandomRange(2, a)
        if pow(g, 2, a) != 1 and pow(g, a/2, a) != 1:
            break
    return (n, e, a, g), (n, d, a, g)
```

可以看到原题里面对d的位数是做了限制的，导致d的位数有概率略微大于N ，这样就触发了variant wiener's attack。

（1）原理

其实不能说是原理，原理完全看不懂，只看懂了要怎么做。。。

```
The attack says that in case d is a few bits greater than N1/4 candidates for private key exponent are of the form equation, where equation an
d equation is (m+1)th and (m)th convergent of continued fraction of e/n
```

也就是说，当

$$d略微大于n$$

$$d=$$

> 连分数（continued fraction）是特殊繁分数。如果a0，a1，a2，…an，…都是整数，则将分别称为无限连分数和有限连分数。可简记为a0，a1，a2，…，an，…和a0，a1，a2，…，an。一般一个有限连分数表示一个有理数，一个无限连分数表示一个无理数。如果a0，a1，a2，…，an，…都是实数，可将上述形式连分数分别叫无限连分数和有限连分数 。近代数学的计算需要，还可将连分数中的a0，a1 ，a2，…，an，…取成以x为变元的多项式。在近代计算数学中它常与某些微分方程式差分方程有关，与某些递推关系有关的函数构造的应用相联系。

所以题目是diffwiener

（2）脚本

```
def wiener(e, n):
    m = 12345
    c = pow(m, e, n)
    q0 = 1

    list1 = continued_fraction(Integer(e) / Integer(n))
    conv = list1.convergents()
    for i in conv:
        k = i.numerator()
        q1 = i.denominator()

        for r in range(20):
            for s in range(20):
                d = r * q1 + s * q0
                m1 = pow(c, d, n)
                if m1 == m:
                    return d
        q0 = q1

n = 24200322442558234307996132315038234291566474519323322540031775165952710648103551437950063774775381153977868195132332338919927142384782851834944584049671557065886464448365645392580003704294348888878310913622766905323895183517166684050749514166506919963464578828710127415058556721750312679968361404291172479016184720986271746746194594910738777543006924611831417771761053109353585286099624819160593315923436130620583343846696678457320860705838883256091132654147170897370138755770600976491762344317367318540966952202958903712647862039756479182970170584421630109314805541595107467920368677464138395140713150840889483059672487008731260933948853067331225426140389200258623125071080485277662734888713129347872342984947294421926566652570406674229942752756313091331710461527286965509797927178483706997377629703776589734722685235492155691867381318550306602338609083168669177566612064716082636800890882628665911588384899661113900855181 9
e = 32963234673870704100445517346726497971982993497766509899224202662312397669756300349164499114559175218253229186693311615325427232711638761383500258584918830185277912612710898987039028167403746482881534150596158363526761447467237298059458062763728934113200075225201456826996881146381342779410067107410327501722012036389269324184712128956362264639576266842507672245563781712095063063372239028055380474848165050656976410475849553306689714864806843343931417281313586724417944220669345219721476918092257794646506975837526585201903620205391913840258273909980240744857368033412917998560674588220531279826739195334643103706381118877150714082841558545758189124566868829356678708089607175412037832923557697472440928879717252423173712752914948903611818017157311504909832683777873218259882575516020781071765763768182067416736542622924995399554315177526266036914268826526183248984711430488854040474167645714482331733098641587043972119818 7
print wiener(e, n)
```

得到d

```
7964539378197021248197485179797985167864055334928177083056633611793675168002545140033012770427375142956412029854190701104956633580329087696830930874275843
```

3.常规rsa

```
import libnum
d = 79645393781970212481974851797979851678640553349281770830566336117936751680025451400330127704273751429564120298541907011049566335803290876968309308742758 43
c = 77946857064750780999513551023189412264862767880123382013538836593457179168709582111830183257444952761894680942965886767587338966662910653689851387548189259931668882179310233797413884692197633746651479318443667352845784023804467579655634728482805128504881806679171647745950343416691608521141456599299657881476306864288949256982779969635810173118915137167502611622570806797104893419955053562149327573966049889045275278815256992268167508111725874680129023959123667504296988514984929417782314697925470551404366763842522224731967556143449238025479335524452405284809905275847072503884676147156002291668439697185689523907470163991559393734652288002322985886758283330035285523651887951572932454095185257828587818971497951028476355582986284496087076110786267515858038203972321465425406108050305442691063636016071236744343054412002309352150975196992086840157846814385176000147660809909455828878880510567691467922078058541096968581061 3
n = 24200322442558234307996132315038234291566474519323322540031775165952710648103551437950063774775381153977868195132332338919927142384782851834944584049671557065886464448365645392580003704294348888878310913622766905323895183517166684050749514166506919963464578828710127415058556721750312679968361404291172479016184720986271746746194594910738777543006924611831417771761053109353585286099624819160593315923436130620583343846696678457320860705838883256091132654147170897370138755770600976491762344317367318540966952202958903712647862039756479182970170584421630109314805541595107467920368677464138395140713150840889483059672487008731260933948853067331225426140389200258623125071080485277662734888713129347872342984947294421926566652570406674229942752756313091331710461527286965509797927178483706997377629703776589734722685235492155691867381318550306602338609083168669177566612064716082636800890882628665911588384899661113900855 1819
m = pow(c,d,n)
print (libnum.n2s(m))
```

4.get flag

```
flag{d33p_@n@lysis_0f_wi3n3r_@tt@ck}
```

结语

关键在于d的大小，所以wiener用不了就用variant wiener。