CTF-CRYPTO-RSA-Repeat

原创

大熊何在 于 2020-11-18 18:30:19 发布 232 收藏 1

分类专栏： CRYPTO

本文链接：https://blog.csdn.net/zippo1234/article/details/109780861

CTF-CRYPTO-RSA-Repeat

每天一题，只能多不能少

RSA-Repeat

题目分析

1. Coppersmith’s Attack
2. next_prime生成素数
3. urandom生成伪随机数

开始

1.题目

给出加密脚本

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Util.number import *
import gmpy2
import os
from secret import flag

p = gmpy2.next_prime(bytes_to_long(os.urandom(32)*10))
q = getPrime(2048)
n = p * q
m = bytes_to_long(flag)
e = 65537
c = pow(m, e, n)
print 'n = ' + str(n)
print 'c = ' + str(c)

# n = 57238032311054380676218793127312872219343570975692681664205200400380290657611525538104940053878905062333995841472465380770022084996394378039102458173429523724581713657524800557538443753056104743952828227429431029043207413638126527052616827473366172793326449519051817059377054546825854923456192822715498501912668417991153552012254818946654690062430789888381810494769855230951832110833068888983308557168434114244931322443780065084445943295439295213278787173145432593978423911414452380732951830642040388773580533461609042467667440580352607009061862629914183440280369270768982247755133139921352157288597330039183287073619992956286793249879415441085169664402003690172070317020102921251001668268941595777617744451007772114158622939775997369045352999140655342029800628249657654108887383205525045763903753487029936463945418958276298918424961396707032622457100629521386038579268122939276396857636962762581118487071790349638592092003886119406175755705924289272461226903170851504586386469930305234600570135060162792148949273439234616076464785249838094075924644465043904498685580064408841662837663454573457791376872151253691535441790425345048520718422448219766295243076284626191033682896450153240693758813934147233791277703379919983954506992772099216503102789768305552865720295546766122546509860453059085296059662991708608480568759873253088496178989307614705789592522439754022553309851620454184088 9
# c = 51074548491363493387088568552851327034975785510021052621774712962776323516709974232188006074374636034445121405462464152869166767261917209097653169527279521807862562656886115188573896937353455048163741948680794256577061588143337916679706458249729460807600572408707640212516337432737190819188793119704532957508655088841662075823402399259358521164688711948248716787050948679899399195564256881582364715978817512755233088551431166018191189375253331262627371966776924745700065503958045655894335151161348094298329140322572229421288912830062820603507741669378011966743260277655425663142338280088189961181903486863839788766059372552348153617351729495937830798750112323389007238998661118051548858979207964376376475633147092198902512560173952037257534933310279080337794416457393766938155987406749209801599374790920636521355817506003662361342278992683545316845752419481555568648383112190755507768379914264792487168937589359478485065626668995625455187329084633175717152402970573659947458655669663658528949333690518516699193510578173573091482329130056337251055784936413471441941813909682174209113413438245485677962629371104573351106928982773643848106798219359883662660016298830657005717892839474338140807683040583575795757932646880007431957736130034028787006833144929680696945467933140208310725266454695095301417129685778710892520489077145892514428939708907353846669556447872244851111077603069387624 25
```

2.分析

核心在于生成p。构造一个伪随机数，然后取伪随机数的紧邻素数，然后随机产生一个2048位的素数，进而生成q。

3.coppersmith’s attack

（1）urandom

bytes_to_long(os.urandom(32)*10)可以看成是bytes_to_long((’\x00’*31+’\x01’)*10)的倍数
也就是可以设一个t = bytes_to_long((’\x00’*31+’\x01’)*10)，bytes_to_long(os.urandom(32)*10)=tx

（2）next_prime

next_prime相对于bytes_to_long(os.urandom(32)*10)的偏移量可以设为i
那么next_prime(bytes_to_long(os.urandom(32)*10))=tx+i

（3）coppersmith’s attack

使用sagemath
首先，定义一个由n生成的环，即Zmod(n)
第二步，基于域的多项式环，PolynomialRing(K)
第三步，monic()生成(t*x+i)多项式十进制等效项
第四步，寻找根root < 2^(nbits//2-kbits) with factor >= n^0.45
由此得到

```
#!python2
# -*- coding: utf-8 -*-
# @Time : 2020/11/18 11:29
# @Author : A.James
# @FileName: tt.py
from Crypto.Util.number import *
n = 57238032311054380676218793127312872219343570975692681664205200400380290657611525538104940053878905062333995841472465380770022084996394378039102458173429523724581713657524800557538443753056104743952828227429431029043207413638126527052616827473366172793326449519051817059377054546825854923456192822715498501912668417991153552012254818946654690062430789888381810494769855230951832110833068888983308557168434114244931322443780065084445943295439295213278787173145432593978423911414452380732951830642040388773580533461609042467667440580352607009061862629914183440280369270768982247755133139921352157288597330039183287073619992956286793249879415441085169664402003690172070317020102921251001668268941595776177444510077721141586229397759973690453529991406553420298006282496576541088873832055250457639037534870299364639454189582762989184249613967070326224571006295213860385792681229392763968576369627625811184870717903496385920920038861194061757557059242892724612269031708515045863864699303052346005701350601627921489492734392346160764647852498380940759246444650439044986855800644088416628376634545734577913768721512536915354417904253450485207184224482197662952430762846261910336828964501532406937588139341472337912777703379919983954506992772099216503102789768305552865720295546766122546509860453059085296059662991708608480568759873253088496178989307614705789592522439754022553309851620454184088 9
t=bytes_to_long(('\x00'*31+'\x01')*10)
#print t
K = Zmod(n)
P.<x> = PolynomialRing(K)
for i in range(1,100000):
    r =(x *t +i).monic().small_roots(beta=0.45)
    if r:
        print i,r
```

由此得到

```
581 [49744071297957798354164147045147783551201517511717977573060692579476485779742]
```

（4）得到p

p=t*x+i
q=n//p
有了pq，就是最普通的rsa了。

4.全脚本

```python
#!python3
# -*- coding: utf-8 -*-
# @Time : 2020/11/18 14:17
# @Author : A.James
# @FileName: tt2.py
from Crypto.Util.number import *
import gmpy2
import os
c = 5107454849136349338708856855285132703497578551002105262177471296277632351670997423218800607437463603444512140546246415286916676726191720909765316952727952180786256265688611518857389693735345504816374194868079425657706158143337916679706458249729460807600572408707640212516337432737190819188793119704532957508655088841662075823402399259358521164688711948248716787050948679899399195564256881582364715978817512755233088551431166018191189375253331262627371966776924745700065503958045655894335151161348094298329140322572229421288912830062820603507741669378011966743260277655425663142338280088189961181903486863839788766059372552348153617351729495937830798750112323389007238998661118051548858979207964376376475633147092198902512560173952037257534933310279080337794416457393766938155987406749209801599374790920636521355817506003662361342278992683545316845752419481555568648383112190755507768379914264792487168937589359478485065626668995625455187329084633175717152402970573659947458655669663658528949333690518516699193510578173573091482329130056337251055784936413471441941813909682174209113413438245485677962629371104573351106928982773643848106798219359883662660016298830657005717892839474338140807683040583575795757932646880007431957736130034028787006833144929680696945467933140208310725266454695095301417129685778710892520489077145892514428939708907353846669556447872244851111077603069387624 25
n = 5723803231105438067621879312731287221934357097569268166420520040038029065761152553810494005387890506233399584147246538077002208499639437803910245817342952372458171365752480055753844375305610474395282822742943102904320741363812652705261682747336617279332644951905181705937705454682585492345619282271549850191266841799115355201225481894665469006243078988838181049476985523095183211083306888898330855716843411424493132244378006508444594329543929521327878717314543259397842391141445238073295183064204038877358053346160904246766744058035260700906186262991418344028036927076898224775513313992135215728859733003918328707361999295628679324987941544108516966440200369017207031702010292125100166826894159577761774445100777211415862293977599736904535299914065534202980062824965765410888738320552504576390375348702993646394541895827629891842496139670703262245710062952138603857926812293927639685763696276258111848707179034963859209200388611940617575570592428927246122690317085150458638646993030523460057013506016279214894927343923461607646478524983809407592464446504390449868558006440884166283766345457345779137687215125369153544179042534504852071842244821976629524307628462619103368289645015324069375881393414723379127777033799199839545069927720992165031027897683055528657202955467661225465098604530590852960596629917086084805687598732530884961789893076147057895925224397540225533098516204541840889
e = 65537
t=bytes_to_long(('\x00'*31+'\x01')*10)
#print t
sol = 4974407129795779835416414704514778355120151751171797757306069257947648577974 2
p=(sol*t+581)
print n % p
q= n/p
phi = (p-1)*(q-1)
d = inverse(e, phi)
print(long_to_bytes(pow(c, d, n)))
```

5.get flag

```
flag{some_problem_in_next_prime}
```

结语

coppersmith‘s attack，完全不懂在干嘛，不懂装懂中。。。。。