BuuCTF_crypto(2021.10.8新-->旧）

原创

虎娃de爹 于 2021-10-08 17:56:30 发布 31 收藏
文章标签： python php
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/zihuocc/article/details/120651135
版权

2021能源安全大赛RSA-number

```
import gmpy2
from Crypto.Util.number import *
nbits = 1024
p = getPrime(nbits)
q = getPrime(nbits)
e = 65537
n = p*q
phi = (p-1)*(q-1)
d = gmpy2.invert(e, phi)
pinv = gmpy2.invert(p, q)
qinv = gmpy2.invert(q, p)
flag = "********************"
m = bytes_to_long(flag)
c = pow(m, e, n)
print(e)
print(phi)
print(c)
print(pinv)
print(qinv)
#e=65537
#phi=162141966774098421917496632132319662234635666532035684007298140720649564618617522781665536129087156565
#c=106040184041128292076531094264213308727649603175841359429768463327749687469634952494717914543320182100 24
#pinv=412275251226417310278035763531215022168868985665300907549108222338858209386305710298515814731205914 77
#qinv=966821675314314328280197355535733446413004972610098198463673583620424556655915874429567650319522853 58
```

根据题意，易得 d = gmpy2.invert(e,phi)

有了 c , d , pinv, qinv 求m ： m = c^d % n , n = p * q

根据费马小定理

$$p^{q-1} \equiv 1 \ mod \ q \rightarrow pinv = p^{q-2} \ mod \ q$$
$$q^{p-1} \equiv 1 \ mod \ p \rightarrow qinv = q^{p-2} \ mod \ p$$
$$p^{(q-1)*(p-1)} \equiv p^{(1-p)\%q} \equiv 1 \ mod \ q$$

sympy解方程

c = pow(m, 2, r)

```
from sympy.ntheory.residue_ntheory import nthroot_mod

# nthroot_mod(a,n,p) Find the solutions to x**n = a mod p

m = nthroot_mod(c,2,r)

from Crypto.Util.number import *

print(long_to_bytes(m))
```

换表

在Python 2.x中。该表是256个字符的字符串。可以使用string.maketrans：

```
>>> import string
>>> tbl = string.maketrans('ac', 'dx')
>>> "abcabc".translate(tbl)
'dbxdbx'
```

在Python 3.x中

```
>>> "abcabc".translate({ord('a'): 'd', ord('c'): 'x'})
'dbxdbx'
```

[2021]XOR

参考文章 [GKCTF 2021]XOR_m0_57291352的博客-CSDN博客[GKCTF 2021]XOR题目from Crypto.Util.number import *from hashlib import md5a = getPrime(512)b = getPrime(512)c = getPrime(512)d = getPrime(512)d1 = int(bin(d)[2:][::-1] , 2)n1 = a*bx1 = a^bn2 = c*dx2 = c^d1flag = md5(str(a+b+c+d).encode()).hexdi https://blog.csdn.net/m0_57291352/article/details/119974008DASCTF | June GKCTF X DASCTF应急挑战杯WriteUP-MISC&CRYPTO篇 - 云+社区 - 腾讯云本次竞赛涵盖WEB、CRYPTO、MISC、PWN、REVERSE常规CTF五大类赛题。 https://cloud.tencent.com/developer/article/1844889

题目

```
from Crypto.Util.number import *
from hashlib import md5

a = getPrime(512)
b = getPrime(512)
c = getPrime(512)
d = getPrime(512)
d1 = int(bin(d)[2:][::-1] , 2)
n1 = a*b
x1 = a^b
n2 = c*d
x2 = c^d1
flag = md5(str(a+b+c+d).encode()).hexdigest()
print("n1 =",n1)
print("x1 =",x1)
print("n2 =",n2)
print("x2 =",x2)

#n1 = 838763494437926958008581070260411839823209237328177881964030384369078520459686780327443648205912546530
#x1 = 470074176751536775598897975923770635978979028109069024580032435083767762464518452611002794398395269020
#n2 = 652881484543771018418888718488067046944779065870107552864512166327018684577228481396960369285618888500
#x2 = 360438668861232087414353226298838456221365979857858321089214326157690828111222335667890008387032752720
```

分析可知，a_bit、b_bit的低位逐位爆破4种可能[0 0, 0 1, 1 1 ,1 0],查看是否满足两个条件

（1）a_bit ^ b_bit == x1_bit

(2) a、b低i位数相乘取余的结果与 a*b取余的结果一致，即 $(a[:-i]*b[:-i])mod(2^i) == (a*b)mod(2^i)$

分析可知，c, d的运算中有

```
d1 = int(bin(d)[2:][::-1] , 2) # 取d的二进制倒序
n2 = c*d
x2 = c^(d1)
```

因此爆破时，同时爆破4个位，即c、d的高位c_high_bit、d_high_bit，c、d的低位c_low_bit、d_low_bit 。此时验证条件改为：

（1）c_low_bit ^ d_high_bit == x2_low_bit

（2）c_high_bit ^ d_low_bit == x2_high_bit

（3）(c_low_bit[:-i] * d_low_bit[:-i]) mod (2**i)== n2_low_bit mod (2**i)

（4）(c_high_bits + c_low_bits) * (d_high_bits + d_low_bits) <= n2

（5）(c_high_bits + c_low_bits + 0b01111111110) * (d_high_bits + d_low_bits + 0b01111111110) >= n2

0b0111111110 是c或d的填充部分，根据i位数改变长度， 即中间部分全部填充1

```
#coding:utf-8
import itertools
def get_ab():
 n1 = 838763494437926958008581070260411839823209237328177881964030384369078520459686780327443648205912546530
 x1 = 470074176751536775598897975923770635978979028109069024580032435083767762464518452611002794398395269020
```

```
 a_list, b_list = [0], [0]

 cur_mod = 1
 for i in range(513):
  cur_mod *= 2
  nxt_as, nxt_bs = [], []
  for al, bl in zip(a_list, b_list):
   for ah, bh in itertools.product([0, 1], repeat=2):
    aa, bb = ah*(cur_mod // 2) + al, bh*(cur_mod // 2) + bl
    if ((aa * bb % cur_mod == n1 % cur_mod) and ((aa ^ bb) == x1 % cur_mod)):
     nxt_as.append(aa)
     nxt_bs.append(bb)

  a_list, b_list = nxt_as, nxt_bs

 for a, b in zip(a_list, b_list):
  if a * b == n1 and a*b-n1==0 and (a^b)-x1==0:
   break

 print("a = ",a)
 print("b = ",b)
 #a =78361471396106552237114697472001640694848788946261668706647406377866094681645553548746194977532775602 8
 #b =10703774182571073361112791376032380096360697926840362483242105878115552437021674861528714598089603406 0
 return a,b

import itertools
def get_cd():
 n1 = 652881484543771018418888718488067046944779065870107552864512166327018684577228481396960369285618888 50

 x1 = 360438668861232087414353226298838456221365979857858321089214326157690828111222335667890008387032752 72

 a_list, b_list, aa_list, bb_list = [0], [0], [0], [0]

 x1_bits = [int(x) for x in f'{x1:0512b}'[::-1]]
 #print(f'{x1:0512b}') == print(bin(x1)[2:].rjust(512,'0'))

 cur_mod = 1
 for i in range(256):
  cur_mod *= 2
  nxt_as, nxt_bs, nxt_aas, nxt_bbs = [], [], [], []
  for al, bl, a2, b2 in zip(a_list, b_list, aa_list, bb_list):
   for ah, bh, ah2, bh2 in itertools.product([0, 1], repeat=4):
    # [0 0 0 0
    # 0 0 0 1
    # 0 0 1 0
    # 0 0 1 1
    # 0 1 0 0
    # 0 1 0 1
    # 0 1 1 0
    # 0 1 1 1
    # 1 0 0 0
    # 1 0 0 1
    # 1 0 1 0
    # 1 0 1 1
    # 1 1 0 0
    # 1 1 0 1
    # 1 1 1 0
    # 1 1 1 1]
    aa, bb, aa2, bb2 = ah*(cur_mod // 2) + al, bh*(cur_mod // 2) + bl, ah2*(cur_mod // 2) + a2, bh2*(cur_mo
    bb2_rev = f'{bb2:0512b}'[::-1]
```

```
    bb2_rev = int(bb2_rev, 2)
    aa2_rev = f'{aa2:0512b}'[::-1]
    aa2_rev = int(aa2_rev, 2)

    gujie = '0' * (i+1) + '1' * (510 - 2 * i) + '0' * (i+1)
    gujie = int(gujie, 2)
    if ((aa * bb % cur_mod == n1 % cur_mod) and ((ah ^ bh2) == x1_bits[i]) and (ah2 ^ bh == x1_bits[511-i])
     nxt_as.append(aa)
     nxt_bs.append(bb)
     nxt_aas.append(aa2)
     nxt_bbs.append(bb2)

  a_list, b_list, aa_list, bb_list = nxt_as, nxt_bs, nxt_aas, nxt_bbs

 for a, b, aa2, bb2 in zip(a_list, b_list, aa_list, bb_list):
  aa2_rev = f'{aa2:0512b}'[::-1]
  aa2_rev = int(aa2_rev, 2)
  bb2_rev = f'{bb2:0512b}'[::-1]
  bb2_rev = int(bb2_rev, 2)
  a = aa2_rev + a
  b = bb2_rev + b
  if (a * b == n1):
   break
 # print(a * b - n1)
 # print((a ^ int(bin(b)[2:][::-1],2)) - x1)
 print('c=',a)
 print('d=',b)

 return a,b
a,b = get_ab()
c,d = get_cd()
from hashlib import md5
flag = md5(str(a+b+c+d).encode()).hexdigest()
print('flag=','GKCTF{'+str(flag)+'}')
```

题目[GKCTF 2021] RRRRsa

参考

[GKCTF 2021]RRRRsa_Irving-的博客-CSDN博客复现[GKCTF 2021]RRRRsa拿到题目后，准备操作，额，，，看了看大佬的wp，恍然大悟，分享一下解题的思路。（部分题目如下）一般看到这种像hint1的式子，都需要用gcd（，n）去分解出q或p。推导过程如下（官方题解）推导过程需要用到二项式定理和费马定理。原理如下：1）二项式定理：2）费马定理：a^p = a mod p；了解原理后，在看题解就容易多了。总结了一下：1）拿到两个式子后，先把括号去掉，然后把常数项去掉；2）之后得到的式子应该是俩个只含p或q的式子，让两个式子的p（https://blog.csdn.net/weixin_51867782/article/details/118573717

题目描述

```
from Crypto.Util.number import *
from gmpy2 import gcd

flag = b'xxxxxxxxxxxxx'
p = getPrime(512)
q = getPrime(512)
m = bytes_to_long(flag)
n = p*q
e = 65537
c = pow(m,e,n)
print('c={}'.format(c))

p1 = getPrime(512)
q1 = getPrime(512)
n1 = p1*q1
e1 = 65537
assert gcd(e1,(p1-1)*(q1-1)) == 1
c1 = pow(p,e1,n1)
print('n1={}'.format(n1))
print('c1={}'.format(c1))
hint1 = pow(2020 * p1 + q1, 202020, n1)
hint2 = pow(2021 * p1 + 212121, q1, n1)
print('hint1={}'.format(hint1))
print('hint2={}'.format(hint2))

p2 = getPrime(512)
q2 = getPrime(512)
n2 = p2*q2
e2 = 65537
assert gcd(e1,(p2-1)*(q2-1)) == 1
c2 = pow(q,e2,n2)
hint3 = pow(2020 * p2 + 2021 * q2, 202020, n2)
hint4 = pow(2021 * p2 + 2020 * q2, 212121, n2)
print('n2={}'.format(n2))
print('c2={}'.format(c2))
print('hint3={}'.format(hint3))
print('hint4={}'.format(hint4))

#c=134923927174698178668834314754537709518374762413719897146837375583957697314165223008519178879579457661320
#n1=750035573790802522195178259989901832266591170197707350805234095617572258836510408825475197481075887194900
#c1=681119010920278130070996278938968385174269710828772040471104047878232792115081837834688914746613651399300
#hint1=235520907163817694849907841168755588957155528969833134067640424163187100762561664724265535202402650200
#hint2=527232296985307678979794339144708311532688270083723072396303871007522268507980233624444992119449967700
#n2=114535923043375970380117920548097404729043079895540320742847840364455024050473125998926311644172960176400
#c2=670542036669016911812152625874471809102254733391432601008311183135214710298893041762354341296322371169900
#hint3=255909234167568135438805549638875769607073336073778894010337184193012788021572048810391163503218721600
#hint4=104100726926923869566862741238876132366916970864374562947844669556403268955625670105641264367038885700
```

看到题目 n1/n2/n3 均不可以直接分解

RoarCTF2019 RSA

题目:

常规解法:

```
A=(((y%x)**5)%(x%y))**2019+y**316+(y+1)/x
p=next_prime(z*x*y)
q=next_prime(z)
A =  26833491826787145242474695127934760098610147810049249054841274803081613777681928680615618865770486464 3
n =  11793080604350737432598229182302728514880723911798736960958351535388981485608809967145439434081676124 2
c =  41971850275428383625653350824107291609587853887037624239544762751558838294718672159979929266922528917 9

#分析：gmpy2.iroot(A,316)=(mpz(83), False),确定了y的范围
 #False表示不能完全开方,例如,iroot(1000,2)=(31,False)
#破解x,y:
A =  26833491826787145242474695127934760098610147810049249054841274803081613777681928680615618865770486464 3
for x in range(1,90):
    for y in range(1,90):
        try:
            if (((y%x)**5)%(x%y))**2019+y**316+(y+1)/x == A:
                print x,y
        except:
            pass

解p和q：
import gmpy2
import sympy
n =  11793080604350737432598229182302728514880723911798736960958351535388981485608809967145439434081676124 2
n1 = n/166
np = (gmpy2.iroot(n1,2))[0]
p = sympy.nextprime(np)
q = n/p
print q,p

破解flag：
import gmpy2
import sympy

p=13991609558311089513359683322750669367930670987317402487689102335586078198117591644632304473291306688078 6
q=84286804568139093453973995920184755228498017995887966793307845395096856615166214726700629357176546313727 0
e=0x10001
c=41971850275428383625653350824107291609587853887037624239544762751558838294718672159979929266922528917912 1
n=p*q
phin=(p-1)*(q-1)

e=3
while(e<0x10003):
    try:
        #d=gmpy2.invert(e,phin)
        #print d,e
        d=gmpy2.invert(e,phin)
        m=pow(c,d,n)
        flag = hex(m)[2:].decode('hex')
        if "CTF" in flag:
            print flag
    except:
        pass
    e = sympy.nextprime(e)
```