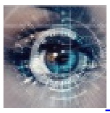
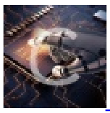
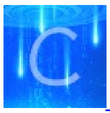
ctf-ichunqiu-crypto(rsa系列)

原创

逃课的小学生 于 2018-07-24 18:41:07 发布 1957 收藏 7
分类专栏： ctf ichunqiu crypto 文章标签： ctf ichunqiu crypto
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/zhang14916/article/details/81183756
版权

ctf 同时被 3 个专栏收录
30 篇文章 2 订阅
订阅专栏

ichunqiu
3 篇文章 0 订阅
订阅专栏

crypto
20 篇文章 1 订阅
订阅专栏

1.rsa256

下载文件，解压得到4个文件，打开message后缀文件里面都是乱码。打开public.key，观察其格式明显是openssl的公钥文件，再根据题目提示可知，我们拿到的message后缀文件是由该公钥rsa加密得到的，用openssl命令获取public.key中的n和e，发现n并不是很大，可以用msieve分解得到p，q。这样我们就可已得到d，用n,d对密文解密，将得到的16进制通过ascii转为字符输出（注意里面存在不可输出字符，直接输出有可能截断字符串导致输出不完整）

```
import sys
import os
import gmpy2
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen
p=302825536744096741518546212761194311477
q=325045504186436346209877301320131277983
e=65537
n=p*q
d=int(gmpy2.invert(e,(p-1)*(q-1)))
with open("encrypted.message1" , "rb") as f:
    s=f.read()
    miwen=long(s.encode('hex'),16)
    mingwenint=pow(miwen,d,n)
    mingwenstr=hex(mingwenint)
    shuchu(mingwenstr)
with open("encrypted.message2" , "rb") as f:
    s=f.read()
    miwen=long(s.encode('hex'),16)
    mingwenint=pow(miwen,d,n)
    mingwenstr=hex(mingwenint)
    shuchu(mingwenstr)
with open("encrypted.message3" , "rb") as f:
    s=f.read()
    miwen=long(s.encode('hex'),16)
    mingwenint=pow(miwen,d,n)
    mingwenstr=hex(mingwenint)
    shuchu(mingwenstr)
```

对结果观察，即可获取明文

2.RSA

打开文件即可看到n，e，c，n比较长，可放入yafu进行分解得到p，q。即可得到d，对c解密输出即可

```
import gmpy2
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen
p=310935513029228809998830208036655366162721470228774287453148308675193510132489142448801010943658159980501
q=310935513029228809998830208036655366162721470228774287453148308675193510132489142448801010943658159980501
e=65537
n=p*q
d=int(gmpy2.invert(e,(p-1)*(q-1)))
c=168502910088858295634315070244377409556567637139736308082186369003227771936407321783557795624279162162305
mingwenint=pow(c,d,n)
mingwenstr=hex(mingwenint)
shuchu(mingwenstr)
```

3.RSA?

打开文件发现e=1，有rsa加密原理可知这里大概率m=c，直接将c转为字符获得答案

4.RSA

打开文件发现已知d，那么直接使用密钥解密即可

5.RSA2

和RSA基本一样，打开文件即可看到n，e，c，n比较长，可放入yafu进行分解得到p，q。即可得到d，对c解密输出即可

```
import gmpy2
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen
p=57970027
q=518629368090170828331048663550229634444384299751272939077168648935075604180676006392464524953128293842996
e=65537
n=p*q
d=int(gmpy2.invert(e,(p-1)*(q-1)))
c='0x3dbf00a02f924a70f44bdd69e73c46241e9f036bfa49a0c92659d8eb0fe47e42068eaf156a9b3ee81651bc0576a91ffed48610
miwen=int(c,16)
mingwenint=pow(miwen,d,n)
mingwenstr=hex(mingwenint)
shuchu(mingwenstr)
```

6.medium RSA

下载文件解压得到一个enc文件，一个pem文件，而enc文件并不是数据包文件，用文本文件打开发现乱码，而pem文件时openssl的公钥文件，根据题目可知enc文件时用RSA公钥文件pem加密得到的密钥文件，用openssl获取pem文件中的n，e，发现n不太大，可放入msieve进行分解得到p，q。即可得到d，对enc文件解密输出即可

```
import sys
import os
import gmpy2
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen
p=275127860351348928173285174381581152299
q=319576316814478949870590164193048041239
e=65537
n=p*q
d=int(gmpy2.invert(e,(p-1)*(q-1)))
with open("flag.enc" , "rb") as f:
    s=f.read()
    miwen=long(s.encode('hex'),16)
    mingwenint=pow(miwen,d,n)
    mingwenstr=hex(mingwenint)
    shuchu(mingwenstr)
```

7.hard RSA

与medium RSA相似，下载文件解压又得到一个enc文件，一个pem文件，且enc文件并不是数据包文件，而pem文件时openssl的公钥文件，根据题目可知enc文件时用RSA公钥文件pem加密得到的密钥文件，用openssl获取pem文件中的n，e，n与medium RSA的n相同，却发现这里的e=2。当e为2时，与n的欧拉函数不再互质，无法求出d，无法再用正常的RSA解密方式来解密。但这里n依旧可以分解为两个素数p，q，且p%4=3，q%4=3，这里我们有Toelli-shanks算法的直接结论使用，对于给定c和p，当p%4=3，当(m^2)=cmodp的解有m=(c^((p+1)/4))modp或m=-(c^((p+1)/4))modp（即m=((c^(1/2))modp)的解），在根据中国剩余定理将模p的解mp和模q的解mq组合为模n的解（这种加解密成为rabin加解密），这里会出现四个解（根据欧几里得扩展原理获得yp*mp+yq*mq=1

r=(yp*p*mq+yq*q*mp)modn

-r=n-r

s=(yp*p*mq-yq*q*mp)modn

-s=n-s

即可得到结果

```
import gmpy2
import libnum
def shuchu(mingwenstr):
    mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen

p=275127860351348928173285174381581152299
q=319576316814478949870590164193048041239
n=p*q
f=open("flag.enc","r")
s=f.read()
f.close()
c=long(s.encode('hex'),16)
#获得c^(1/2)modp,q的解
r=pow(c,(p+1)/4,p)
s=pow(c,(q+1)/4,q)
#使用中国定理组合解
pni=int(gmpy2.invert(p,q))
qni=int(gmpy2.invert(q,p))
a=(s*p*pni+r*q*qni)%n
a1=n-a
b=(s*p*pni-r*q*qni)%n
b1=n-b
shuchu(hex(a))
shuchu(hex(a1))
shuchu(hex(b))
shuchu(hex(b1))
```

8.very hard RSA

下载文件解压又得到两个enc文件，一个py文件，首先打开py文件阅读加密算法，可知这一次是分别使用RSA公钥（N，e1）和（N，e2）对相同密文加密，可以使用RSA共模攻击破解密文

```
import gmpy2
#在a,b较小时可用这种欧几里得扩展
def egcd(a, b):
  if a == 0:
    return (b, 0, 1)
  else:
    g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen

n=0x00b0bee5e3e9e5a7e8d00b493355c618fc8c7d7d03b82e409951c182f398dee3104580e7ba70d383ae5311475656e8a964d380c
e1=17
e2=65537
g,s1,s2=egcd(e1, e2)
fo1 = open('flag.enc1','rb')
fo2 = open('flag.enc2','rb')
data1 = fo1.read()
data2 = fo2.read()
fo1.close()
fo2.close()
c1 = int(data1.encode('hex'),16)
c2 = int(data2.encode('hex'),16)
if s1<0:
    c1=int(gmpy2.invert(c1,n))
    s1=-s1
if s2<0:
    c2=int(gmpy2.invert(c2,n))
    s2=-s2
miwenint=(pow(c1,s1,n)*pow(c2,s2,n))%n
shuchu(hex(miwenint))
```

9.Round Rabins!

已知此题是rabin解密（在hard RSA已经提到如何解密），发现n较长，放在yafu中分解，结果发现n是一个平方数，这样正常的rabin解密无法解密此题（中国剩余定理要求分解得到的因子互素），此时问题变成了如何求解m=(c^(1/2))%(p^2)(其中n=p^2)。我们可以根据Toelli-shanks算法（在hard RSA已经提到）求出m=(c^(1/2))mod(p)，此时问题变为如何通过m=(c^(1/2))mod(p)得到x=(c^(1/2))mod(p^2)，我们假定m+ps=(c^(1/2))mod(p^2)，那么(m+ps)^2=cmod(p^2)，即m^2+2*m*p*s+(p^2)*(s^2)=cmod(p^2)，所以m^2+2*m*p*s=cmod(p^2)，2*m*p*s=(c-m^2)mod(p^2)。我们已知m^2=cmod(p)，即c-m^2可整除p，所以s=((c-m^2)/(2*m*p))modp(简单说明：当mp=npmod(p^2)，则((m-n)*p)%(p^2)=0，要是等式成立，一定有m-n=kp，即m=nmodp)，那么m+ps=(m+p*((c-m^2)/(2*m*p)))，即m+ps=(m+((c-m^2)/(2*m)))，即(c^(1/2))mod(p^2)的解为x=(m+(c-m^2/(2*m)))mod(p^2)（这里只是简单的完成了Hensel's lemma 的事，想知到详情可查看原定理）,根据此原理可解出密文

```
import gmpy2
import libnum
def legendre_symbol(a, p):
    ls = pow(a, (p - 1)/2, p)
    if ls == p - 1:
        return -1
    return ls
def prime_mod_sqrt(a, p):
    a %= p
    if a == 0:
        return [0]
    if p == 2:
        return [a]
    if legendre_symbol(a, p) != 1:
        return []
    if p % 4 == 3:
        x = pow(a, (p + 1)/4, p)
        return [x, p-x]
    q, s = p - 1, 0
    while q % 2 == 0:
        s += 1
        q //= 2
    z = 1
    while legendre_symbol(z, p) != -1:
        z += 1
    c = pow(z, q, p)
    x = pow(a, (q + 1)/2, p)
    t = pow(a, q, p)
    m = s
    while t != 1:
        i, e = 0, 2
        for i in xrange(1, m):
            if pow(t, e, p) == 1:
                break
            e *= 2
        b = pow(c, 2**(m - i - 1), p)
        x = (x * b) % p
        t = (t * b * b) % p
        c = (b * b) % p
        m = i

    return [x, p-x]
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
# This finds a solution for c = x^2 (mod p^2)
def find_solution(c, p):
    n = p ** 2
    r = prime_mod_sqrt(c,p)[0]
    inverse_2_mod_n = modinv(2, n)
    inverse_r_mod_n = modinv(r, n)

    new_r = r - inverse_2_mod_n * (r - c * inverse_r_mod_n)

    return new_r % n

if __name__ == "__main__":
    n = 0x6b612825bd7972986b4c0ccb8ccb2fbcd25fffbadd57350d713f73b1e51ba9fc4a6ae862475efa3c9fe7dfb4c89b4f92e
    p = 0xa5cc6d4e9f6a893c148c6993e1956968c93d9609ed70d8366e3bdf300b78d712e79c5425ffd8d480afcefc71b50d85e09
    c = 0xd9d6345f4f961790abb7830d367bede431f91112d11aabe1ed311c7710f43b9b0d5331f71a1fccbfca71f739ee5be42c1
    solution = find_solution(c, p)
    print hex(solution)[2:-1].decode("hex")
```

10.RSA-5

n=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8330c36f55d36fb129d30
e=0x10001
nextprime(p)*nextprime(q)=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8
c=0x1c3588ac81ec3d1b439cfd2d5e6e8a5a95c8f95aaeff1b0ba49276ade80435323f307a17006ae2ffb4ca321e
在这里我们假设nextprime(p)*nextprime(q)=(p+x)*(q+y)，并设nextprime(p)*nextprime(q)=n1，p*q=n2，将q替换掉，即可得到y*(p^2)+(x*y+n2-n1)p+n2*x=0，这是一个一元二次方程，所以直接对x，y进行爆破，直接用公式求解即可得到p，代码如下

```
import gmpy2
n2=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8330c36f55d36fb129d308c23e5cb8f4d61a
e=0x10001
n1=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8330c36f55d36fb129d308c23e5cb8f4d61a
c=0x1c3588ac81ec3d1b439cfd2d5e6e8a5a95c8f95aaeff1b0ba49276ade80435323f307a17006ae2ffb4ca321e54387d9b33ed7cc
p=1147914946815141439902683714232821831382267846458689095582240247380116337138335805495220097212452997514353
q=1329408022890183362619874153125339530427645969840325481573275294950893078891273549145285072772099404574500

'''
def qiugen(a,b,c):
    ga=gmpy2.mpz(a)
    gb=gmpy2.mpz(b)
    gc=gmpy2.mpz(c)
    delat=gb**2-4*ga*gc
    if delat<=0 or ga==0:
        return 0
    de=gmpy2.iroot(delat,2)
    if de[1]:
        x1=(de[0]-gb)/(2*ga)
        x2=(-de[0]-gb)/(2*ga)
        if x1>0 and n2%x1==0:
            return x1
        if x2>0 and n2%x2==0:
            return x2
    return 0
for x in xrange(0,1500):
    for y in xrange(1,1500):
        a=y
        b=x*y+n2-n1
        c=n2*x
        if not qiugen(a,b,c)==0:
            p=qiugen(a,b,c)
            q=n2/p
            print p,q
'''
def shuchu(mingwenstr):
    if mingwenstr[len(mingwenstr)-1]=='L':
        mingwenstr=mingwenstr[2:len(mingwenstr)-1]
    else:
        mingwenstr=mingwenstr[2:len(mingwenstr)]
    if not len(mingwenstr)%2==0:
            mingwenstr='0'+mingwenstr
    i=len(mingwenstr)
    mingwen=""
    while i>=1:
        str1=mingwenstr[i-2:i]
        if int(str1,16)>33 and int(str1,16)<126:
            mingwen=chr(int(str1,16))+mingwen
        else :
            mingwen=" "+mingwen
        i=i-2
    print mingwen
d=gmpy2.invert(e,(p-1)*(q-1))
mingwen=pow(c,d,n1)
mingwenhex=hex(mingwen)
print mingwenhex
shuchu(mingwenhex)
```

11.HCTF2016-Crypto so interesting

题目不难，观察代码发现这里随机生成RSA的p，q，然后随机生成一个比较小的u，求u对phi_n求逆得到t，再求t对bt求逆获得e，最后求e对phi_n求逆得到d，用e对明文加密，解这道题并不难，已知e，n，先用e对bt求逆获得t，由于u比较小，这里可以使用wiener攻击获得u，phi_n以及p，q，直接可以求解d对密文进行解密，这里的重点是里面使用wiener攻击的代码有利于我们理解对rsa的wiener攻击

```
import libnum
import gmpy2
def pi_b(x):
    bt = 53638095835061605724269141863488059450219210633231722805196706432764209129768763017418363628837823
    return libnum.invmod(x, bt)
def isqrt(n):
    x = n
    y = (x + 1) // 2
    while y < x:
        x = y
        y = (x + n // x) // 2
    if pow(x, 2) == n:
     return x
    else:
     return False
def con_fra(a, b):
 r = []
 while True:
  if a == 1:
   break
  tmp = a/b
  if tmp != 0:
   r.append(tmp)
  a, b = b, (a-tmp*b)
 return r

def wiener_attack(e, n):
 cf = con_fra(e, n)
 for x in xrange(len(cf)):
  k, d = 0, 1
  while x >= 0:
   k, d = d, d*cf[x] + k
   x -= 1
  # print "k: %s\nd: %s\n" %(k, d)
  phi_n = (e*d - 1)/k
  B = n - phi_n + 1
  C = n
  dt = pow(B, 2) - 4*C    # b^2 - 4*a*c
  if dt >= 0 and isqrt(dt) and (B+isqrt(dt)) % 2 == 0:
   print "phi_n: ", hex(phi_n)
   print "p",hex((B+isqrt(dt)) / 2)
   print "q",hex((B-isqrt(dt)) / 2)
   return phi_n
 print "wiener attack fail!"

n=0x763b60d8a9bc44d609847cf9ccb2642d519e9699f13d0242767b30ec151552a4daaf02929a606cb9ad6e974ff38ea33a54ec0f1
e=0x38df4b9719a20d237ea83a24394cd54b0470a22ed2705b4b8b74adc67a8302fdf89296d7daa2b02163bac22384bdf0d396406a9
flag=0x2df4647e8a965e64defe9a4746827d467ab439330641bb98ccd6300ad2c7566763fb19f40eb012aa216b6868216f8e8ec362
t=pi_b(e)
wiener_attack(t,n)
```

12.HCTF2016-Crypto so cool

题目随机生成了p和q_t，相乘获得n_t，取n_t的左边1024/16部分和右边(5*1024)/8部分作为n的左边1024/16部分和右边(5*1024)/16部分，再选择p的左边(5*1024)/16部分做des加密，放入n的中间，q=n/p，如q为偶数则q+1，若q不是素数将异或一个长为1024/16的随机数做异或，直至q变为素数，n=p*q，随机生成e2对明文加密。我们获得n，e，e2，c。我们可以先将n中有关p的部分取出，再使用Coppersmith partial information attack算法还原p，根据n，e，p即可对c进行解密

```
from Crypto.Util.number import getPrime, long_to_bytes, bytes_to_long, isPrime, size
from Crypto.Cipher import DES
from libnum import gcd, invmod
from hashlib import sha512
import signal
import random
import gmpy2
key = "abcdefg1"
k = 2048
def get_bit(number, n_bit, dire):
 '''
 dire:
  1: left
  0: right
 '''

 if dire:
  sn = size(number)
  if sn % 8 != 0:
   sn += (8 - sn % 8)
  return number >> (sn-n_bit)
 else:
  return number & (pow(2, n_bit) - 1)

def pi_b(x, m):
 '''
 m:
  1: encrypt
  0: decrypt
 '''
 enc = DES.new(key,DES.MODE_ECB)
 if m:
  method = enc.encrypt
 else:
  method = enc.decrypt
 s = long_to_bytes(x)
 sp = [s[a:a+8] for a in xrange(0, len(s), 8)]
 r = ""
 for a in sp:
  r += method(a)
 return bytes_to_long(r)
k = 2048
n=0x78861bb9529cd40d3e29f8ebed4b697d2132a3ecb25904cac59f379828a2638f8e507d28c02c2eaadbd723c44733e36e0c522b3
e=0x10001
e2=0x999d
flag=0x73907b2828a84e6af79d748464ec23471db575bd101e203f63bd6c28bea060bc48bb1a4f4b72e8b234557232e75ade97006e
c1=21342536467
m1= 0x2e72c47a71ce9186aa3474e243457364491504260e2712ba9340325f62e1fcb0731bb7a9e35c42d03730b8c1884d1ac4fccda

u=get_bit(get_bit(n,3*k/8,1),5*k/16,0)
```

```
p4=pi_b(u,0)
print hex(p4)
#p,q可由后面的脚本求出
p=0xdfdf6ba55b2e1de773a3b5c2f041380bd3ac13974ce4ebd8f726d7a771acc73097f9670ceb47be920a3a556ac27cf82286a8c0b
q=0x89d1e21689e49a7d6388ea7438ea9563b7b6a37467d8a419e0f7fae1420628f63dd6fd238f7ecd537fa147c04530d943f7b12cb
d=gmpy2.invert(e2,(p-1)*(q-1))
flag=pow(flag,d,n)
print flag
print long_to_bytes(flag)
```

sega脚本：

```
p4=0xdfdf6ba55b2e1de773a3b5c2f041380bd3ac13974ce4ebd8f726d7a771acc73097f9670ceb47be920a3a556ac27cf82286a8c0
n=0x78861bb9529cd40d3e29f8ebed4b697d2132a3ecb25904cac59f379828a2638f8e507d28c02c2eaadbd723c44733e36e0c522b3

pbits = 1024
kbits = pbits - p4.nbits()
print p4.nbits()
p4 = p4 << kbits

PR.<x> = PolynomialRing(Zmod(n))
f = x + p4
x0 = f.small_roots(X=2^kbits, beta=0.4)[0]
print "x: %s" %hex(int(x0))

p = p4+x0
print "p: ", hex(int(p))
assert n % p == 0
q = n/int(p)

print "q: ", hex(int(q))
```

12.HCTF2016-Crypto so amazing

和上一题类似，只是这里的密钥p的一部分是通过D-H函数生成的，所以我们首先拿到通过反向拿到p的一部分，再使用Coppersmith partial information attack算法还原p，根据n，e，p即可对c进行解密，其中pi_sit_x函数虽然使用了hash函数，但其中结构可逆(由于yu = F_hash(H_hash(xu) ^ xl) ^ xu；yl = H_hash(xu) ^ xl，那么存在H_hash(xu)=yl^xl，即yu = F_hash(H_hash(xu) ^ xl) ^ xu可转变为yu = F_hash(yl) ^ xu，xu=yu^F_hash(yl)；xl=yl^H_hash(yu^F_hash(yl))；答案中的xu推到结果不正确，但由于最后取的是后256位，再xl中，所以对结果不影响）

```
from Crypto.Util.number import size, long_to_bytes, bytes_to_long, getRandomNBitInteger
from hashlib import sha512
import itertools
import random
import time
k = 2048
e = 0x10001
o = 1024
m = 256
def get_bit(number, n_bit, dire):
  '''
 dire:
   1: left
   0: right
  '''
 if dire:
  sn = size(number)
  if sn % 8 != 0:
   sn += (8 - sn % 8)
  return number >> (sn-n_bit)
 else:
  return number & (pow(2, n_bit) - 1)

def int_add(x1, x2):
 '''
 bit plus
 '''
 return bytes_to_long(long_to_bytes(x1) + long_to_bytes(x2))

def H_hash(x):
 h = sha512(long_to_bytes(x)).hexdigest()
 return int(h, 16)

def F_hash(x):
 h = sha512(long_to_bytes(x/4)).hexdigest()
 return int(h, 16)

def pi_sit_x2(sit, z):
 '''
 inverse operation
 '''
 zu = get_bit(z, sit/2, 1)
 zl = get_bit(z, sit/2, 0)
 xu = zu ^ F_hash(zl)
 xl = zl ^ H_hash(zu ^ F_hash(zl))
 return int_add(xu, xl)
def sha512_proof(fuzz, prefix, verify):
 y = len(verify)
 while True:
  try:
   padd = "".join(fuzz.next())
  except StopIteration:
   break
  r = sha512(prefix + padd).hexdigest()
  if verify in r:
   return padd

def verify(r):
 r.readuntil("Prefix: ")
 prefix = r.readline()
 prefix = prefix.decode('base64')
 t1 = time.time()
 proof = sha512_proof(fuzz, prefix, "fffffff")
 print time.time() - t1
 r.send(proof.encode('base64'))

def main():
        P=0xe49614ad3a11e66bfbe847e477e1b283630790dd5975d478b8e75c56c89571d9
        b=9223372036854775808
        n=0x8b4abe0fc81d1ac4e027eda051960f6681a0921698ff310e60ffa754fa1b730dbb19ba0cc916b338a80ad1d8536c132
        e=0x10001
        e2=0x840d
        flag=0x80dc303e2ed66eaa76c5b0e1edafd68641df4a6e27401e2d5699e6df3974335e15239640f625df590a8bf325da96
        cipher=1234567890
        Plaintext=0x4356b108d0a28cd1ce4aa23c147da6c2d3ddcdbc3824446421f75fd624e34f053a691b96a58fd4e9d16cf45
        print "n:",n
        print "e:",e
        print "e2:",e2
        print "flag:",flag
        t=get_bit(n,1024,1)
        print "t:",hex(t)
        s=pi_sit_x2(o,t)
        print "s:",hex(s)
        attack_spub=get_bit(s,m,0)
        attack_spriv=pow(attack_spub,b,P)
        print "spub:",hex(attack_spub)
        print "spriv:",hex(attack_spriv)

if __name__ == '__main__':
 main()
```

sage3.py

```
import random

spriv = 0xbc100f24304ac73a357877ce1e57500521d1b0429591d75d931bca82f94f2fea
T = 512 + 64

PRF = random.Random()
PRF.seed(spriv)

def get_p4():
    while True:
        u = PRF.randint(2**(T-1), 2**T)
        yield u

f=get_p4()
print get_p4().next()
```

sage脚本

```
from sage3 import get_p4

n=0x8b4abe0fc81d1ac4e027eda051960f6681a0921698ff310e60ffa754fa1b730dbb19ba0cc916b338a80ad1d8536c132d922022f
pbits = 1024

g_p = get_p4()
while True:
    p4 = g_p.next()
    # p4 = 0x81a722c9fc2b2ed061fdab737e3893506eae71ca6415fce14c0f9a45f8e2300711119fa0a5135a053e654fead010b9
    kbits = pbits - 576
    p4 = p4 << kbits

    PR.<x> = PolynomialRing(Zmod(n))
    f = x + p4
    x0 = f.small_roots(X=2^kbits, beta=0.4)
    if len(x0) == 0:
        continue
    print "x: %s" %hex(int(x0[0]))

    p = p4+x0[0]
    print "p: ", hex(int(p))
    assert n % p == 0
    q = n/int(p)

    print "q: ", hex(int(q))
    print "p4: ", hex(p4)
    break
```

结果

```
from Crypto.Util.number import size, getPrime, long_to_bytes, bytes_to_long, isPrime, getRandomNBitInteger
from libnum import invmod, gcd
from hashlib import sha512
import random
p=0x851f7cd8fe49730a7a481e19e6afe52b91912617358544e310d75622199d7bad447a92cf257583fb34342e78a5e32dec0a34cae1
q=0x10bdcf5be7479deafce95c7eef5cdd721a585fdc9a3b524afa20eaf35666267bd6228f6e5ef46029d1d67b6b83badcf8b3e327a
n=0x8b4abe0fc81d1ac4e027eda051960f6681a0921698ff310e60ffa754fa1b730dbb19ba0cc916b338a80ad1d8536c132d922022f
e2=0x840d
flag=0x80dc303e2ed66eaa76c5b0e1edafd68641df4a6e27401e2d5699e6df3974335e15239640f625df590a8bf325da96c719d6cb
phi_n=(p-1)*(q-1)
d = invmod(e2,phi_n)
enc_flag = pow(flag, d, n)
flag = long_to_bytes(enc_flag)
print enc_flag
print flag
```