XCTF新手区Crypto writeup

原创

KogRow 于 2020-06-30 20:09:53 发布 474 收藏 2
分类专栏： CTF 文章标签： CTF crypto

1.base64

直接拿去base64 decode就行：cyberpeace{Welcome_to_new_World!}

2.Caesar

这题凯撒密码。

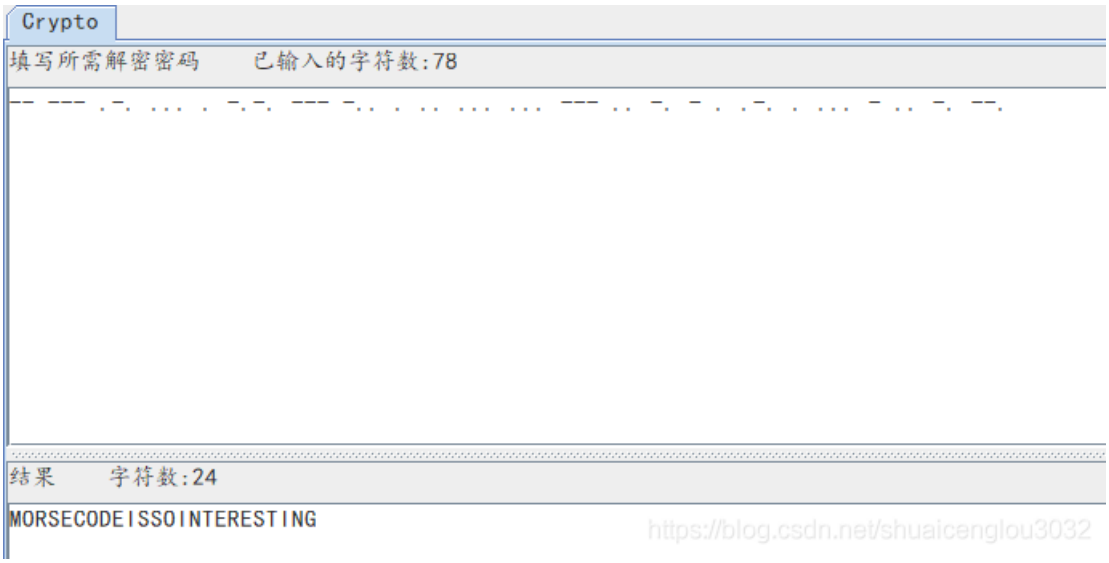
分别设置位移量为1-25，输出25条结果后看到位移量为12时有意义，故位移量为12：
cyberpeace{you_have_learned_caesar_encryption}

3.Morse

摩斯密码，我们在记事本里将1全部替换成“-”，将0全部替换成“.”，得到密文：

-- --- .-. ... . -.-. --- -.. . .. . --- .. -. - . .-. . ... - .. -. --.

摩斯密码解码：



将其转成小写就拿到了flag。

4.混合编码

原文：
JiM3NjsmlzEyMjsmlzY5OyYjMTIwOyYjNzk7JiM4MzsmlzU2OyYjMTIwOyYjNzc7JiM2ODsmlzY5OyYjMTE4OyYj

一看就是base64,先解码，解码出来是这样：

```
&#76;&#122;&#69;&#120;&#79;&#83;&#56;&#120;&#77;&#68;&#69;&#118;&#77;&#84;&#65;&#52;&#76;&#122;&#107;&#53;&
```

猜想是unicode的HEX编码，拿去解码：

```
LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8
```

再次猜想是baae64:

/119/101/108/99/111/109/101/116/111/97/116/116/97/99/107/97/110/100/100/101/102/101/110/99/101/119/111

这里大佬说是ASCII,把它放进eclipse，将所有“/”替换为“,”，构建数组，转为字符串：

```
public class Main {
 public static void main(String[] args) {
  int[] a = {119,101,108,99,111,109,101,116,111,97,116,116,97,99,107,97,110,100,100,101,102,101,110,99,101,
  String result = "";
  for(int i:a){
   char c = (char)i;
   result+=c;
  }
  System.out.println(result);
 }
}
```

得到结果：

```
<terminated> Main [Java Application] C:\Program Files\Java\jre
welcometoattackanddefenceworld
```

5.幂数加密

密文：88421012204802244040142242024801 22

提示：flag为八位大写字母

这里学习一波幂数加密的知识：

```
二进制幂数加密法：

二进制幂数加密法，由于英文字母只有26个字母。只要2的0、1、2、3、4、5次幂就可以表示31个单元。通过用二进制幂数表示字母序号数

二进制数除了0和1的表示方法外，在由二进制转换成十进制的时候，还可以表示成2的N次方的形式。例如：

15=2^0+2^1+2^2+2^3

并且我们发现，任意的十进制数都可以用2^n或2^n+2^m+……的形式表示出来，可以表示的单元数由使用的max n来决定。

可表示的单元数=2^(n+1)-1
```

也就是说，根据提示，在0之前分，将密文分为8段：88421 0122 048 02244 04 0142242 0248 0122

但是如果是2的幂的话只需要0-5就行，密文中不会出现8，因此这和上面说的还有点不太一样。

百度了一下，说是另一种01248密码：

```
01248密码，又称云影密码…与二进制幂加密不同，这个加密法采用的是0作间隔，其他非0数隔开后组合起来相加表示序号1-26之一的字母
```

上python3代码：

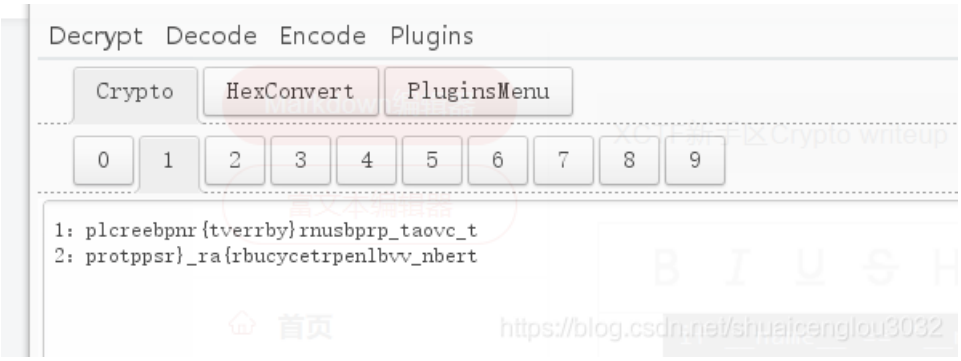
```
#01248密码
def zeronetfe(m):
    alpha = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
    m = m.split("0")
    c = ""
    for i in range(0,len(m)):
        str = m[i]
        num = 0;
        for j in range(0,len(str)):
            num += int(str[j])
        c += alpha[num-1]
    return c
def main():
    print(zeronetfe("884210122048022440401422420248012 2"))
if __name__ == '__main__':
    main()
```

运行之后就得到结果。

6.Railfence

密文：ccehgyaefnpeoobe{lcirg}epriec_ora_g

这题是栅栏密码，但是和常见的栅栏密码不一样，拿去在工具里解密失败：



根据提示：

```
到了一看，这个谜面看起来就已经有点像答案了样子了，旁边还画着一张画，是一副农家小院的 图画，上面画着一个农妇在栅栏里面喂5只
```

暗示栅栏是5。

百度了一下发现是一种叫WWW的变种：

```
此段介绍来自https://blog.csdn.net/qq_43504939/article/details/98473847
1 2 3 4 5 6 # key=3 Rail-fence Cipher

1 . . . 5 . ↘          ↗ ↘
. 2 . 4 . 6       ↘     ↗
. . 3 . . .        ↘ ↗

结果为 1 5 2 4 6 3
```

此处贴一波其他大佬的python加解密代码：

```
def encode(string, key):#需要加密的字符串以及加密栏数
    i = 0
    enlist = []
    for j in range(0, key):
        enlist.append('')#添加分组，列表的一个元素相当于一个分组

    while i < len(string):#分组重排进行加密
        for k in range(0, key):
            if i >= len(string):
                break
            enlist[k] += string[i]
            i += 1
        for k in range(1, key-1):
            if i >= len(string):
                break
            enlist[key-1-k] += string[i]
            i += 1
        enstr = ''
    for i in range(key):
        enstr += enlist[i]
    return enstr

代码来自https://blog.csdn.net/qinying001/article/details/96134356

def decode(string, key):#解密字符串以及解密栏数
    de_key = 2*key - 2#一个部分的长度
    length = len(string)//de_key#确定有多少个完整部分
    r = len(string)%de_key#最后不完整部分的长度
    delist = []
    for i in range(key):
        delist.append('')#重新排布分组
    #确定第一个分组
    if r == 0:
        delist[0] += string[0:length]
        s = length
    else:
        delist[0] += string[0:length+1]
        s = length+1
    #确定第二个到第key-1个分组
    for i in range(1, key-1):
        l = length*2#这几个分组长度至少是完整部分数量的两倍
        #最后一个不完整部分对应当前分组有几个元素
        if r > i:
            l += 1
        if r > de_key-i:
            l += 1
```

```
        delist[i] += string[s:s+l]
        s = s+l
    #确定最后一个分组
    delist[key-1] += string[s:]
#排布分组确定原文字符串
    destr = ''
    j = 0
    for i in range(0, len(string)):
        destr += delist[j][0]
        delist[j] = delist[j][1:]
        if j == key-1:
            flag = 0
        if j == 0:
            flag = 1
        if flag:
            j += 1
        else:
            j -= 1
    return destr
```

运行之，拿到明文：

```
cyberpeace{railfence_cipher_gogogo}

Process finished with exit code 0
```

7.easy_RSA

```
在一次RSA密钥对生成中，假设p=473398607161，q=4511491，e=17
求解出d
```

这道题简单粗暴，直接上python3代码：

```
#已知p,q,e,求解密密钥d
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('模数分解失败')
    else:
      return x % m
def main():
    e=17
    p=473398607161
    q=4511491
    d = modinv(e,(p-1)*(q-1))
    print(d)
if __name__ == '__main__':
    main()
```

执行上述代码就得到flag:

8.不仅仅是Morse

--/.-/-.--/..--.-/-..././..--.-/..../.-/...-/./..--.-/.-/-./---/-/...././.-./..--.-/-.././-.-./---/-.././..../..../..../..../.-/.-/.-/.-/.-/-.../.-/.-/-.../-.../-
.../.-/.-/-.../-.../.-/.-/.-/.-/.-/.-/.-/.-/-.../.-/.-/-.../.-/-.../.-/.-/.-/.-/.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../-.../.-/.-/.-/-.../-.../.-/.-/-.../.-/.-/.-
/.-/-.../.-/-.../.-/.-/-.../.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../.-/.-/.-/-.../.-/.-/-.../.-/-.../-.../.-/.-/-.../-.../-.../.-/-.../.-/.-/.-/-.../.-/-.../.-/-.../-
.../.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../.-/.-/-.../.-/.-/-.../.-/.-/.-/.-/-.../-.../.-/-.../-.../.-/.-/-.../-.../.-/.-/-.../.-/.-/-.../.-/.-/.-/-.../.-/.-/-.../.-
/.-/-.../.-/.-/-.../.-/-.../.-/.-/-.../-.../.-/-.../.-/.-/.-/.-/-.../-.../.-/-.../.-/.-/-.../-.../.-

将斜杠去除得到摩斯密码：

-- .- -.-- ..--.- -... . ..--.- .... .- ...- . ..--.- .- -. --- - .... . .-. ..--.- -.. . -.-. --- -.. . ...- .- .- .- .- -... .- .- -... -... -... .- .- -... -... .- .- .- .- .- .- .- .- -... .- .- -... .- -... .- .- .- .- .- .- .- -... -... .- -... .- .- .- -... -... .- .- .- -... -... .- .- .- -... .- .- .- -... .- .- -... .- -... -... .- .- .- -... -... .- .- -... .- .- .- -... .- .- -... .- .- -... -... .- .- .- -... -... .- -... .- .- .- -... .- .- -... .- .- -... .- .- .- .- -... -... .- -... -... .- .- -... -... .- .- -... .- .- .- .- -... .- .- -... .- .- -... .- .- -... .- -... .- .- -... .- .- -... -... .-

解码：

MAYnullBEnullHAVEnullANOTHERnullDECODEHHHHAAAAABAABBBAABBAAAAAAAABAABABAAAAAAABBA

may be have another decode
hhhh AAAAABAABBBAABBAAAAAAAABAABABAAAAAAABBABAAABBAAABBAABAAAABABAABAAABBABAAAB

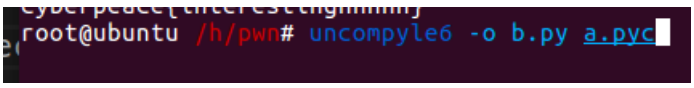
根据提示，后面这截由A和B组成的估计是培根密码，直接拿来解密，得到:

这里也可以自己写python代码：

```
https://blog.csdn.net/weixin_42109012/article/details/97644262
```

9.easychallenge

下载下来是一个pyc文件，使用反编译工具反编译：



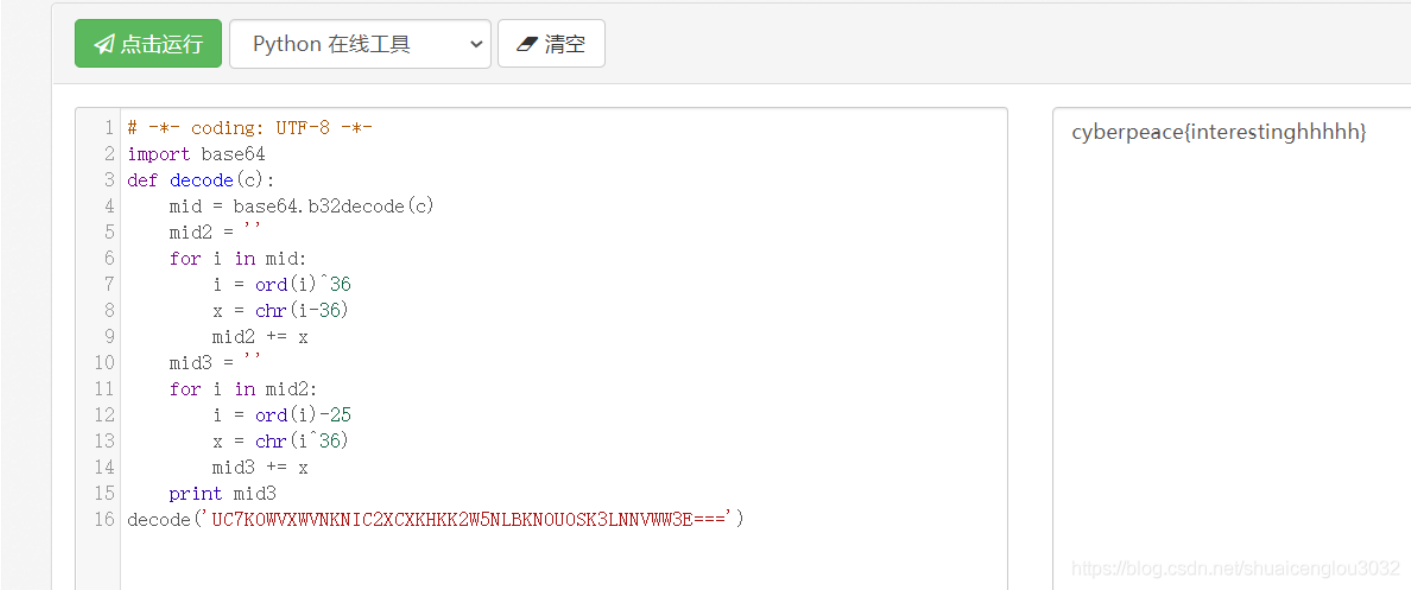
得到源代码：

```
# uncompyle6 version 3.7.2
# Python bytecode 2.7 (62211)
# Decompiled from: Python 3.5.2 (default, Apr 16 2020, 17:47:17)
# [GCC 5.4.0 20160609]
# Embedded file name: ans.py
# Compiled at: 2018-08-08 20:29:44
import base64

def encode1(ans):
    s = ''
    for i in ans:
        x = ord(i) ^ 36
        x = x + 25
        s += chr(x)

    return s


def encode2(ans):
    s = ''
    for i in ans:
        x = ord(i) + 36
        x = x ^ 36
        s += chr(x)

    return s


def encode3(ans):
    return base64.b32encode(ans)


flag = ' '
print 'Please Input your flag:'
flag = raw_input()
final = 'UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E==='
if encode3(encode2(encode1(flag))) == final:
    print 'correct'
else:
    print 'wrong'
```

根据源代码编写一个decode方法：

```
import base64
def decode(c):
    mid = base64.b32decode(c)
    mid2 = ''
    for i in mid:
        i = ord(i)^36
        x = chr(i-36)
        mid2 += x
    mid3 = ''
    for i in mid2:
        i = ord(i)-25
        x = chr(i^36)
        mid3 += x
    print mid3
decode('UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E===')
```

执行上述代码得到flag:



10.

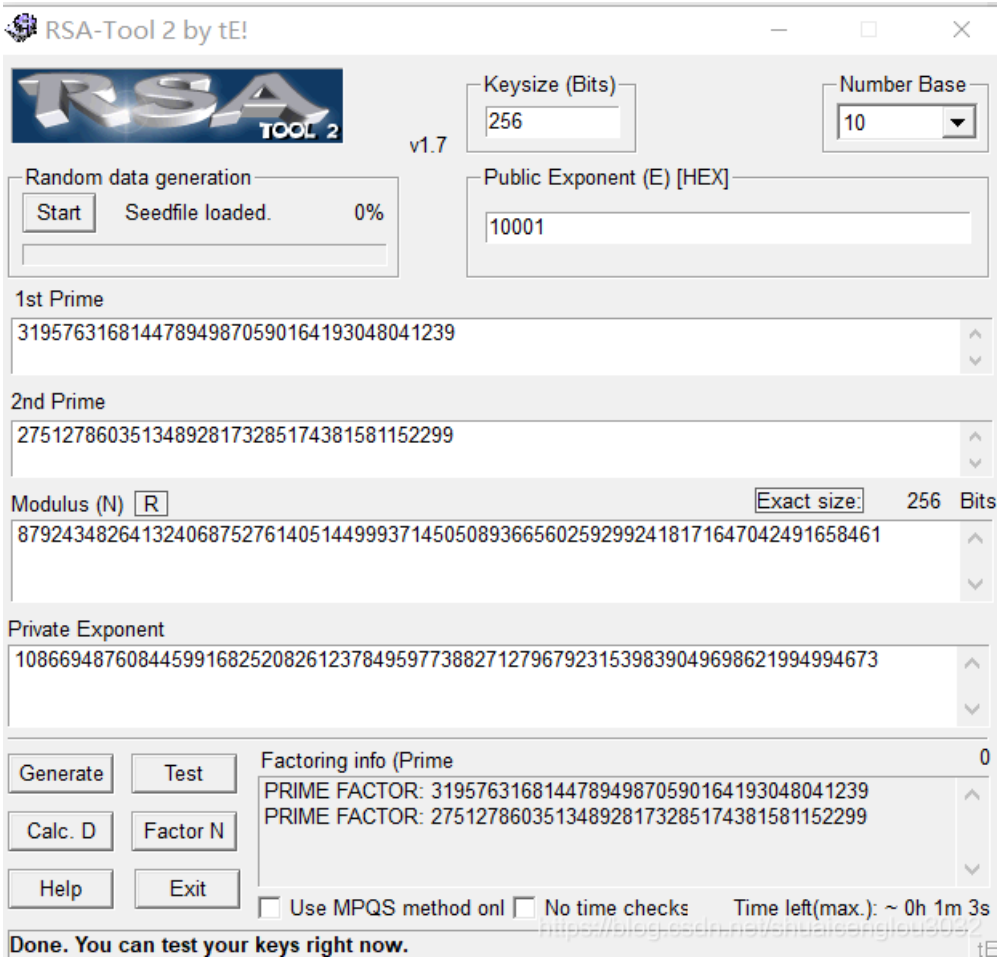
11.Normal_RSA

下载下来是一个enc和一个pem文件。

在linux里使用命令：openssl rsa -pubin -text modulus in warmup -in pubkey.pem

提取公钥：

将十六进制的公钥放进RSATool，暴力分解，20分钟之后得到结果：



p:319576316814478949870590164193048041239

q:275127860351348928173285174381581152299

然后点击Calc D，计算私钥d:

108669487608445991682520826123784959773882712796792315398390496986219949946 73

然后根据上述生成私钥文件：

```
python rsatool.py -f PEM -o private.pem -p 275127860351348928173285174381581152299 -q 319576316814478949870
```

再使用peivate.pem解密enc文件：

openssl rsautl -decrypt -in flag.enc -inkey private.pem -out flag.txt
就能拿到flag

12.easy_ECC

已知椭圆曲线加密Ep(a,b)参数为

p = 15424654874903

a = 16546484

b = 4548674875

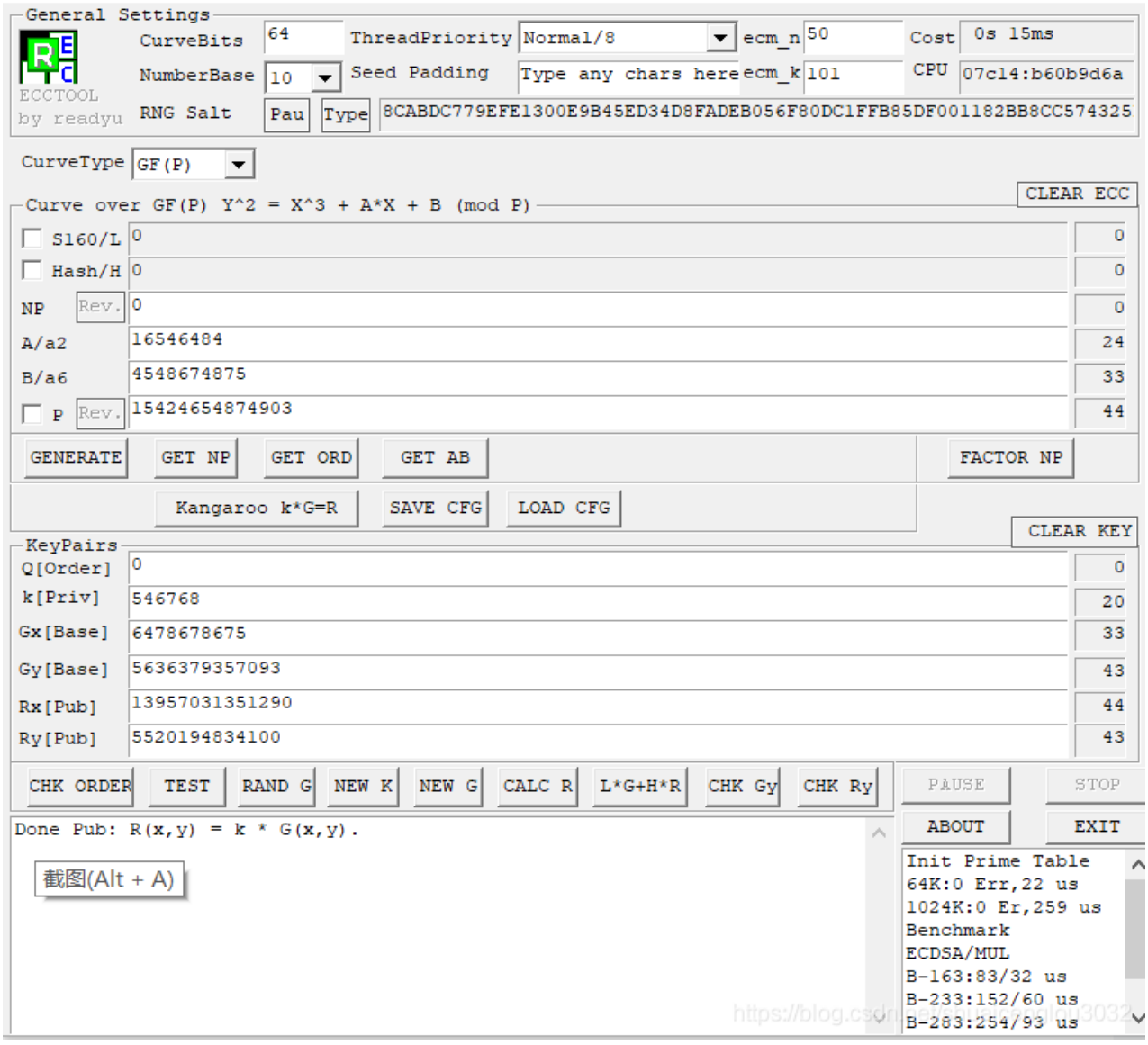
G(6478678675,5636379357093)

私钥为

k = 546768

求公钥K(x,y)

使用大佬写的工具ecctool，参数设置如下：



点击calc R就得到结果 R(x,y)