攻防世界crypto新手区writeup

原创

a370793934 于 2019-11-27 15:44:57 发布 2869 收藏 4
分类专栏： WriteUp 文章标签： 攻防世界 crypto writeup ctf

本文链接：https://blog.csdn.net/a370793934/article/details/103277055

0x01 base64

元宵节灯谜是一种古老的传统民间观灯猜谜的习俗。 因为谜语能启迪智慧又饶有兴趣，灯谜增添节日气氛，是一项很有趣的活动。 你也很喜欢这个游戏，这不，今年元宵节，心里有个黑客梦的你，约上你青梅竹马的好伙伴小鱼， 来到了cyberpeace的攻防世界猜谜大会，也想着一展身手。 你们一起来到了小孩子叽叽喳喳吵吵闹闹的地方，你俩抬头一看，上面的大红灯笼上写着一些奇奇怪怪的 字符串，小鱼正纳闷呢，你神秘一笑，我知道这是什么了。

base64解密即可

```
#!/user/bin/env python
# -*-coding:utf-8 -*-
import base64

a = open(r'crypto1.txt','r')
s = a.read()

print(base64.b64decode(s))
```

cyberpeace{Welcome_to_new_World!}

0x02 Caesar

你成功的解出了来了灯谜，小鱼一脸的意想不到“没想到你懂得这么多啊！”你心里面有点小得意，“那可不是，论学习我没你成绩好轮别的我知道的可不比你少，走我们去看看下一个”你们继续走，看到前面也是热热闹闹的，同样的大红灯笼高高挂起，旁边呢好多人叽叽喳喳说个不停。你一看 大灯笼，上面还是一对字符，你正冥思苦想呢，小鱼神秘一笑，对你说道，我知道这个的答案是什么了

凯撒码解密即可

参考

```
#!/user/bin/env python
# -*-coding:utf-8 -*-
```

```
a = open(r'crypto2.txt','r')
ciphertext = a.read()
b='abcdefghijklmnopqrstuvwxyz'

for key in range(26):
    flag = ''
    for i in ciphertext:
        if i in b:
            num = b.find(i)
            num = num - key

if num<0:
                num = num + len(b)
            flag = flag + b[num]
        else:
            flag = flag + i
    print('key %s :%s'%(key,flag))
```

cyberpeace{you_have_learned_caesar_encryption}

0x03 Morse

小鱼得意的瞟了你一眼，神神气气的拿走了答对谜语的奖励，你心里暗暗较劲 想着下一个谜题一定要比小鱼更快的解出来。不知不觉你们走到了下一个谜题的地方，这个地方有些奇怪。 上面没什么提示信息，只是刻着一些0和1，感觉有着一些奇怪的规律，你觉得有些熟悉，但是就是想不起来 这些01代表着什么意思。一旁的小鱼看你眉头紧锁的样子，扑哧一笑，对你讲“不好意思我又猜到答案了。”(flag格式为cyberpeace{xxxxxxxxxx},均为小写)

1变为-,0变为.,摩斯码解密即可

参考

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

CODE_TABLE = {
    # 26 个英文字符
    'A': '.-', 'B': '-...', 'C': '-.-.',
    'D': '-..', 'E': '.', 'F': '..-.',
    'G': '--.', 'H': '....', 'I': '..',
    'J': '.---', 'K': '-.-', 'L': '.-..',
    'M': '--', 'N': '-.', 'O': '---',
    'P': '.--.', 'Q': '--.-', 'R': '.-.',
    'S': '...', 'T': '-', 'U': '..-',
    'V': '...-', 'W': '.--', 'X': '-..-',
    'Y': '-.--', 'Z': '--..',
```

```
# 10 个数字
    '0': '-----', '1': '.----', '2': '..---',
    '3': '...--', '4': '....-', '5': '.....',
    '6': '-....', '7': '--...', '8': '---..',
    '9': '----.',

# 16 个特殊字符
    ',': '--..--', '.': '.-.-.-', ':': '---...', ';': '-.-.-.',
    '?': '..--..', '=': '-...-', '"': '.----.', '/': '-..-.',
    '!': '-.-.--', '-': '-....-', '_': '..--.-', '(': '-.--.',
    ')': '-.--.-', '$': '...-..-', '&': '. . . .', '@': '.--.-.'

# 你还可以自定义

}

def morsedecode(morse):
    msg =''
    codes = morse.split(' ')
    for code in codes:
        if code =='':
            msg += ' '
        else:
            UNCODE =dict(map(lambda t:(t[1],t[0]),CODE_TABLE.items()))
            msg += UNCODE[code]
    return msg

a = open(r'crypto3.txt','r')
ciphertext = a.read()

ciphertext = ciphertext.replace('1','-')
ciphertext = ciphertext.replace('0','.')

FLAG = morsedecode(ciphertext)
flag = FLAG.lower()
flag = 'cyberpeace{'+flag+'}'
print('flag is ',flag)

cyberpeace{morsecodeissointeresting}
```

0x04 Railfence

被小鱼一连将了两军，你心里更加不服气了。两个人一起继续往前走， 一路上杂耍卖艺的很多，但是你俩毫无兴趣，直直的就冲着下一个谜题的地方去了。 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只小鸡，你嘿嘿一笑对着小鱼说这次可是我先找到答案了。

题目直接提示为栅栏密码

对于ccehgyaefnpeoobe{lcirg}epriec_ora_g，似乎无法栅出来，}应该在最后<T_T>???

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

a = open(r'crypto4.txt','r')
ciphertext = a.read()
b = len(ciphertext)

print('The ciphertext :',ciphertext)
print('The length of ciphertext is',b)

# 将字符串转化为数组形式
ciphertext = [ciphertext[i:i+1] for i in range(0,b,1)]

# 获取可以解密的栅栏数
c =[]
num = 1
while num <= b:
    if b%num == 0:
        c.append(num)
    else:
        pass
    num += 1
print('每栏个数可为:',c)
# 将密文按照栅栏数进行分组
for step in c:
    d = [ciphertext[i:i+step] for i in range(0,b,step)]
    flag = ''
    for i in range(step):
        for x in d:
            e = x[i]
            flag += e
    print('解密时每栏个数:',step,'\n',d,'\n','result:',flag)
```

待定。。。。

0x05 不仅仅是Morse

“这个题目和我们刚刚做的那个好像啊但是为什么按照刚刚的方法做出来答案却不对呢”，你奇怪的问了问小鱼，“可能是因为还有一些奇怪的加密方式在里面吧，我们在仔细观察观察”。两个人 安安静静的坐下来开始思考，很耐心的把自己可以想到的加密方式一种种的过了一遍，十多分钟后两个人 异口同声的说“我想到了！”

将/化为空格，转化为标准morse码，进行morse解密，得到一串类似培根加密后的密文，对培根码解密即可得到flag.(这里附上培根解密脚本)

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import re
```

```
# 密文转化为指定格式
s =
'AAAAABAABBBAABBAAAAAAAABAABABAAAAAAABBABAAABBAAABBAABAAAABABAABAAABBABAAABAAAE
a = s.lower()
```

```
# 字典
CODE_TABLE = {
    'a':'aaaaa','b':'aaaab','c':'aaaba','d':'aaabb','e':'aabaa','f':'aabab','g':'aabba',
    'h':'aabbb','i':'abaaa','j':'abaab','k':'ababa','l':'ababb','m':'abbaa','n':'abbab',
    'o':'abbba','p':'abbbb','q':'baaaa','r':'baaab','s':'baaba','t':'baabb','u':'babaa',
    'v':'babab','w':'babba','x':'babbb','y':'bbaaa','z':'bbaab'
}

# 5个一组进行切割并解密
def peigendecode(peigen):
    msg =''
    codes = re.findall(r'.{5}', a)
    for code in codes:
        if code =='':
            msg += ' '
        else:
            UNCODE =dict(map(lambda t:(t[1],t[0]),CODE_TABLE.items()))
            msg += UNCODE[code]
    return msg

flag = peigendecode(a)
print('flag is ',flag)
```

```
cyberpeace{attackanddefenceworldisinteresting}
```

0x06 easy_RSA

解答出来了上一个题目的你现在可是春风得意，你们走向了下一个题目所处的地方 你一看这个题目傻眼了，这明明是一个数学题啊！！！可是你的数学并不好。扭头看向小鱼，小鱼哈哈一笑，让你在学校里面不好好听讲现在傻眼了吧~来我来！三下五除二，小鱼便把这个题目轻轻松松的搞定了

RSA的算法涉及三个参数，n、e、d。
其中，n是两个大质数p、q的积，n的二进制表示所占用的位数，就是所谓的密钥长度。
e和d是一对相关的值，e可以任意取，但要求e与(p-1)(q-1)互质；再选择d，要求(de)mod((p-1)*(q-1))=1。
（n，e),(n，d)就是密钥对。其中(n，e)为公钥，(n，d)为私钥。
RSA加解密的算法完全相同，设A为明文，B为密文，则：A=B^d mod n；B=A^e mod n；（公钥加密体制中，一般用公钥加密，私钥解密）
e和d可以互换使用，即：
A=B^d mod n；B=A^e mod n

直接给了p,q,e，求d，即为暴力破解RSA的私钥之一的d

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
```

```
import gmpy2

p = 473398607161
q = 4511491
e = 17

s = (p-1)*(q-1)
d = gmpy2.invert(e,s)
print('flag is :',d)
```

cyberpeace{125631357777427553}

0x07 Railfence 转轮机加密

题目描述

你俩继续往前走，来到了前面的下一个关卡，这个铺面墙上写了好多奇奇怪怪的 英文字母，排列的的整整齐齐，店面前面还有一个大大的类似于土耳其旋转烤肉的架子，上面一圈圈的 也刻着很多英文字母，你是一个小历史迷，对于二战时候的历史刚好特别熟悉，一拍大腿：“嗨呀！我知道 是什么东西了！”。提示：托马斯·杰斐逊

附件：转轮机加密-crypto.txt

1: < ZWAXJGDLUBVIQHKYPNTCRMOSFE <

2: < KPBELNACZDTRXMJQOYHGVSFUWI <

3: < BDMAIZVRNSJUWFHTEQGYXPLOCK <

4: < RPLNDVHGFCUKTEBSXQYIZMJWAO <

5: < IHFRLABEUOTSGJVDKCPMNZQWXY <

6: < AMKGHIWPNYCJBFZDRUSLOQXVET <

7: < GWTHSPYBXIZULVKMRAFDCEONJQ <

8: < NOZUTWDCVRJLXKISEFAPMYGHBQ <

9: < XPLTDSRFHENYVUBMCQWAOIKZGJ <

10: < UDNAJFBOWTGVRSCZQKELMXYIHP <

11： < MNBVCXZQWERTPOIUYALSKDJFHG <

12： < LVNCMXZPQOWEIURYTASBKJDFHG <

13： < JZQAWSXCDERFVBGTYHNUMKILOP <

密钥为： 2,3,7,5,13,12,9,1,8,10,4,11,6

密文为：NFQKSEVOQOFNP

分析

转轮机加密

参看：传统密码学(三)——转轮密码机 是个多表替换密码。 基本操作是“旋转”，其余的和本题是没有关系。

加密表和密钥、密文的特点

加密表每一行都有 26 个不同的字母，密钥的长度、密文长度、表格的行数都是 13，密文中字母是有重复字母的。

猜想一：

由此推断，密文是滚轮机的加密表某一列的字母连起来。 然而经过尝试并非如此，每一行对应密文中的字母对应的序号与下一行的字母对应的序号之间的差值不能根据密钥推出！ 调整调整！！！是否有其他解法？

猜想二：

跳出之前的思维，之前的思考角度是试图逆向想出它的加密方案， 但是如果同时有密文和密钥，其实只要知道解密方案就好了， 虽然加密和解密是逆向的过程，但是方法不同。所以可以双向思考、双向破解：

根据密文密钥的特点，猜测加密方法，对密码进行攻击。

根据密文密钥，推测解密方法，使用解密方法对密文进行解密。

既然猜想一似乎没有出路了，接下来尝试 2 对应的思路。 构造解密和加密的映射，加密 - 解密，密文 - 明文，密钥 - 密钥； 现在的明文：NFQKSEVOQOFNP 现在的密钥：2,3,7,5,13,12,9,1,8,10,4,11,6 密钥的每个数字对应一行，那么明文的字母如果对应一列（比如以这个字母为首）,就能唯一确定一个表格。然后加密表的某一列应该会有密文（映射原来的明文）。

题解

方法一：“手动”转

转前：

OSFEZWAXJGDLUBVIQHKYPNTCRM

NACZDTRXMJQOYHGVSFUWIKPBEL

FHTEQGYXPLOCKBDMAIZVRNSJUW

FCUKTEBSXQYIZMJWAORPLNDVHG

KCPMNZQWXYIHFRLABEUOTSGJVD

PNYCJBFZDRUSLOQXVETAMKGHIW

QGWTHSPYBXIZULVKMRAFDCEONJ

QNOZUTWDCVRJLXKISEFAPMYGHB

VUBMCQWAOIKZGJXPLTDSRFHENY

OWTGVRSCZQKELMXYIHPUDNAJFB

NBVCXZQWERTPOIUYALSKDJFHGM

EIURYTASBKJDFHGLVNCMXZPQOW

SXCDERFVBGTYHNUMKILOPJZQAW

转后：

NACZDTRXMJQOYHGVSFUWIKPBEL

FHTEQGYXPLOCKBDMAIZVRNSJUW

QGWTHSPYBXIZULVKMRAFDCEONJ

KCPMNZQWXYIHFRLABEUOTSGJVD

SXCDERFVBGTYHNUMKILOPJZQAW

EIURYTASBKJDFHGLVNCMXZPQOW

VUBMCQWAOIKZGJXPLTDSRFHENY

OSFEZWAXJGDLUBVIQHKYPNTCRM

QNOZUTWDCVRJLXKISEFAPMYGHB

OWTGVRSCZQKELMXYIHPUDNAJFB

FCUKTEBSXQYIZMJWAORPLNDVHG

NBVCXZQWERTPOIUYALSKDJFHGM

PNYCJBFZDRUSLOQXVETAMKGHIW

密钥为：2,3,7,5,13,12,9,1,8,10,4,11,6

密文为：N F Q K S E V O Q O F N P

在第 17 行发现有语义的：FIREINTHEHOLE

cyberpeaceP{fireinthehole}

方法写 Python 脚本（别人的 writeup）

```
import re

sss = '1: < ZWAXJGDLUBVIQHKYPNTCRMOSFE < 2: < KPBELNACZDTRXMJQOYHGVSFUWI < 3: < BDMAIZVRNSJUWFHTEQGYXPLOCK < 4: < RPLNDVHGFCUKTEBSXQYIZMJWAO < 5: < IHFRLABEUOTSGJVDKCPMNZQWXY < 6: < AMKGHIWPNYCJBFZDRUSLOQXVET < 7: < GWTHSPYBXIZULVKMRAFDCEONJQ < 8: < NOZUTWDCVRJLXKISEFAPMYGHBQ < 9: < XPLTDSRFHENYVUBMCQWAOIKZGJ < 10: < UDNAJFBOWTGVRSCZQKELMXYIHP < 11 < MNBVCXZQWERTPOIUYALSKDJFHG < 12 < LVNCMXZPQOWEIURYTASBKJDFHG < 13 < JZQAWSXCDERFVBGTYHNUMKILOP <'

m = 'NFQKSEVOQOFNP'

# 将sss转化为列表形式

content=re.findall(r'< (.*?) <',sss,re.S)

# re.S:DOTALL，此模式下，"."的匹配不受限制，可匹配任何字符，包括换行符

iv=[2,3,7,5,13,12,9,1,8,10,4,11,6]

print(content)

vvv=[]

for i in range(13):

    index=content[iv[i]-1].index(m[i])

    vvv.append(index)

print(vvv)


for i in range(0,26):

    flag=""

    for j in range(13):

        flag += content[iv[j]-1][(vvv[j]+i)%26]

    print(flag.lower())
```

提交fireinthehole

0x08 混合编码

经过了前面那么多题目的历练，耐心细致在解题当中是 必不可少的品质，刚巧你们都有，你和小鱼越来越入迷。那么走向了下一个题目，这个题目好长 好长，你知道你们只要细心细致，答案总会被你们做出来的，你们开始慢慢的尝试，慢慢的猜想 ，功夫不负有心人，在你们耐心的一步步的解答下，答案跃然纸上，你俩默契一笑，相视击掌 走向了下面的挑战。

多种加密，主要是base64与unicode解密

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import base64

a = open(r'crypto8.txt','r')
s = a.read()

# base64解密一下
b = base64.b64decode(s).decode('ascii')
# 对解密后的字符串进行处理
b = b.strip('&#;')
c = []
c = b.split(';&#')
# unicode解密
d = ''
for i in c:
    d += chr(int(i))
# base64再次解密
e = base64.b64decode(d).decode('ascii')
# 对字符进行处理
e = e.strip('/')
f = []
f = e.split('/')
# 转化为ascii码
flag =''
for i in f:
    flag += chr(int(i))
print('flag is ',flag)
```

cyberpeace{welcometoattackanddefenceworld}

0x09 Normal_RSA

你和小鱼走啊走走啊走，走到下一个题目一看你又一愣，怎么还是一个数学题啊 小鱼又一笑，hhhh数学在密码学里面很重要的！现在知道吃亏了吧！你哼一声不服气，我知道数学 很重要了！但是工具也很重要的，你看我拿工具把他解出来！你打开电脑折腾了一会还真的把答案 做了出来，小鱼有些吃惊，向你投过来一个赞叹的目光

OpenSSL 使用 PEM 文件格式存储证书和密钥。PEM 实质上是 Base64 编码的二进制内容，再加上开始和结束行，如证书文件的
-----BEGIN CERTIFICATE-----
和
-----END CERTIFICATE-----
在这些标记外面可以有额外的信息，如编码内容的文字表示。文件是 ASCII 的，可以用任何文本编辑程序打开它们。

流程：
1、openssl提取出pubkey.pem中的参数；
openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem

2、把hex转decimal后得到十进制数，用yafu进行分解，得到p和q ；

在线：http://www.factordb.com/

factor()

3、用rsatool生成私钥文件: private.pem

python rsatool.py -o private.pem -e 65537 -p XXX -q XXX

4、用private.pem解密flag.enc

openssl rsautl -decrypt -in flag.enc -inkey private.pem

PCTF{256b_i5_m3dium}

0x10 easychallenge

你们走到了一个冷冷清清的谜题前面，小鱼看着题目给的信息束手无策，丈二和尚摸不着头脑 ，你嘿嘿一笑，拿出来了你随身带着的笔记本电脑，噼里啪啦的敲起来了键盘，清晰的函数逻辑和流程出现在 了电脑屏幕上，你敲敲键盘，更改了几处地方，运行以后答案变出现在了电脑屏幕上。

解释型语言和编译型语言的区别
计算机是不能够识别高级语言的，所以当运行一个高级语言程序时，就需要一个“翻译机”来从事把高级语言转变成计算机能读懂的机器语言的过程。这个过程分成两类，第一种是编译，第二种是解释。
（1）编译型语言：在程序执行之前，先会通过编译器对程序执行一个编译的过程，把程序转变成机器语言。运行时就不需要翻译，而直接执行就可以了。最典型的例子就是C语言。
（2）解释型语言：没有编译的过程，而是在程序运行时，通过解释器对程序逐行解释，然后直接运行，最典型的例子是Ruby。
（3）编译型语言与解释型语言的优缺点：编译型语言在程序运行之前就已经对程序做出了“翻译”，所以在运行时就少掉了“翻译”的过程，所以效率比较高。但是我们也不能一概而论。
（4）先编译后解释的语言：Java首先是通过编译器编译成字节码文件，然后在运行时通过解释器给解释成机器文件。Java等基于虚拟机的语言的存在，我们又不能把语言纯粹地分成解释型和编译型这两种。Python也是一门基于虚拟机的语言。当我们在命令行中输入python hello.py时，其实是激活了Python的“解释器”，告诉“解释器”：你要开始工作了。可是在“解释”之前，其实执行的第一项工作和Java一样，是编译。

Python的运行过程
关于PyCodeObject和pyc文件：在硬盘上看到的pyc文件，其实PyCodeObject才是Python编译器真正编译成的结果。当python程序运行时，编译的结果是保存在位于内存中的PyCodeObject中，当Python程序运行结束时，Python解释器则将PyCodeObject写回到pyc文件中。当python程序第二次运行时，首先程序会在硬盘中寻找pyc文件，如果找到，则直接载入，否则就重复上面的过程。所以，我们可以说pyc文件其实是PyCodeObject的一种持久化保存方式。

对于py文件，可以执行下面命令来生成pyc文件。
python -m foo.py

.py与.pyc文件区别
原来Python的程序中，是把原始程序代码放在.py文件里，而Python会在执行.py文件的时候。将.py形式的程序编译成中间式文件（byte-compiled）的.pyc文件，这么做的目的就是为了加快下次执行文件的速度。所以，在我们运行python文件的时候，就会自动首先查看是否具有.pyc文件，如果有的话，而且.py文件的修改时间和.pyc的修改时间一样，就会读取.pyc文件，否则，Python就会读原来的.py文件。
其实并不是所有的.py文件在与运行的时候都会产生.pyc文件，只有在import相应的.py文件的时候，才会生成相应的.pyc文件

给的文件为.pyc文件，先进行反编译为.py文件，再写解密脚本

反编译.pyc文件：
· 反编译可以使用uncompyle6或者在线反编译pyc反编译
· 安装uncompyle6 ： pip install uncompyle6
· 反编译命令 uncompyle6 crypto11.pyc

反编译结果：

```
#!/usr/bin/env python
# encoding: utf-8
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'
```

解密脚本
涉及python3下ord() expected string of length 1, but int found

```
#!/usr/bin/env python
# encoding: utf-8

import base64
'''
a = 'UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E==='
b = base64.b32decode(a)
'''
# 先将字符串base32解码出来，然后将其变为字符串再进行ord等操作。
b = "\xa0\xbe\xa7Z\xb7\xb5Z\xa6\xa0Z\xb8\xae\xa3\xa9Z\xb7Z\xb0\xa9\xae\xa3\xa4\xad\xad\xad\xad\xad\xb2"
s = ''
for i in b:
    s += chr((ord(i) ^ 36) - 36)
l = ''
for i in s:
    l += chr((ord(i) - 25) ^ 36)
print ('flag is ',l)
```

cyberpeace{interestinghhhhh}

0x11 幂数加密

你和小鱼终于走到了最后的一个谜题所在的地方，上面写着一段话“亲爱的朋友， 很开心你对网络安全有这么大的兴趣，希望你一直坚持下去，不要放弃 ，学到一些知识， 走进广阔的安全大世界”，你和小鱼接过谜题，开始了耐心细致的解答。

(O.A.O)>>给的密文是云影密码

```
#!/user/bin/env python
# -*-coding:utf-8 -*-

a = open(r'crypto11.txt','r')
ciphertext = a.read()

s = ciphertext.split('0')
```

```
flag = ''
for i in range(len(s)):
    list = []
    for j in s[i]:
        list.append(j)
    b = 0
    for k in list:
        b += int(k)
    # 字母ascii值与字母顺序相差为96
    flag += chr(b+96)
print('flag is ',flag)
```

cyberpeace{welldone}

0x12 easy_ECC

转眼两个人又走到了下一个谜题的地方，这又是一种经典的密码学加密方式 而你刚好没有这个的工具，你对小鱼说“小鱼我知道数学真的很重要了，有了工具只是方便我们使用 懂了原理才能做到，小鱼你教我一下这个缇努怎么做吧！”在小鱼的一步步带领下，你终于明白了ECC 的基本原理，成功的解开了这个题目，两个人相视一笑，快步走向了下一个题目所在的位置。

椭圆曲线加密原理

ECC椭圆曲线详解

ECC加密算法入门介绍

Alice选定一条椭圆曲线E，并取椭圆曲线上一点作为基点G 假设选定E29(4,20)，基点G(13,23) , 基点G的阶数n=37

Alice选择一个私有密钥k（k<n），并生成公开密钥K=kG 比如25, K= kG = 25G = (14,6）

Alice将E和点K、G传给Bob

Bob收到信息后，将待传输的明文编码到上的一点M（编码方法略），并产生一个随机整数r（r<n,n为G的阶数） 假设r=6 要加密的信息为3,因为M也要在E29(4,20) 所以M=(3,28)

Bob计算点C1=M+rK和C2=rG C1= M+6K= M+625G=M+2G=(3,28)+(27,27)=(6,12) C2=6G=(5,7)

Bob将C1、C2传给Alice

Alice收到信息后，计算C1-kC2，结果就应该是点M C1-kC2 =(6,12)-25C2 =(6,12)-25*6G =(6,12)-2G =(6,12)-(27,27) =(6,12)+(27,2) =(3,28)

数学原来上能解密是因为:C1-kC2=M+rK-krG=M+rkG-krG-M

参考

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
```

```
def get_inverse(mu, p):
    """
    获取y的负元
    """
    for i in range(1, p):
        if (i*mu)%p == 1:
            return i
    return -1

def get_gcd(zi, mu):
    """
    获取最大公约数
    """
    if mu:
        return get_gcd(mu, zi%mu)
    else:
        return zi

def get_np(x1, y1, x2, y2, a, p):
    """
    获取n*p，每次+p，直到求解阶数np=-p
    """
    flag = 1  # 定义符号位（+/-）

# 如果 p=q  k=(3x2+a)/2y1mod p
    if x1 == x2 and y1 == y2:
        zi = 3 * (x1 ** 2) + a  # 计算分子      【求导】
        mu = 2 * y1    # 计算分母

# 若P≠Q，则k=(y2-y1)/(x2-x1) mod p
    else:
        zi = y2 - y1
        mu = x2 - x1
        if zi* mu < 0:
            flag = 0        # 符号0为-（负数）
            zi = abs(zi)
            mu = abs(mu)

# 将分子和分母化为最简
    gcd_value = get_gcd(zi, mu)    # 最大公約數
    zi = zi // gcd_value           # 整除
    mu = mu // gcd_value
    # 求分母的逆元  逆元： ∀a ∈G ， ∃b∈G 使得 ab = ba = e
    # P(x,y)的负元是 (x,-y mod p)= (x,p-y) ，有P+(-P)= O∞
    inverse_value = get_inverse(mu, p)
    k = (zi * inverse_value)
```

```
if flag == 0:                # 斜率负数 flag==0
        k = -k
    k = k % p
    # 计算x3,y3 P+Q
    x3 = (k ** 2 - x1 - x2) % p
    y3 = (k * (x1 - x3) - y1) % p
    return x3,y3

def get_rank(x0, y0, a, b, p):
    """
    获取椭圆曲线的阶
    """
    x1 = x0            #-p的x坐标
    y1 = (-1*y0)%p     #-p的y坐标
    tempX = x0
    tempY = y0
    n = 1
    while True:
        n += 1
        # 求p+q的和，得到n*p，直到求出阶
        p_x,p_y = get_np(tempX, tempY, x0, y0, a, p)
        # 如果 == -p,那么阶数+1，返回
        if p_x == x1 and p_y == y1:
            return n+1
        tempX = p_x
        tempY = p_y

def get_param(x0, a, b, p):
    """
    计算p与-p
    """
    y0 = -1
    for i in range(p):
        # 满足取模约束条件，椭圆曲线Ep(a,b)，p为质数，x,y∈[0,p-1]
        if i**2%p == (x0**3 + a*x0 + b)%p:
            y0 = i
            break

# 如果y0没有，返回false
    if y0 == -1:
        return False

# 计算-y（负数取模）
    x1 = x0
    y1 = (-1*y0) % p
    return x0,y0,x1,y1
```

```
def get_ng(G_x, G_y, key, a, p):
    """
    计算nG
    """
    temp_x = G_x
    temp_y = G_y
    while key != 1:
        temp_x,temp_y = get_np(temp_x,temp_y, G_x, G_y, a, p)
        key -= 1
    return temp_x,temp_y

def ecc_main():
    while True:
        a = int(input("请输入椭圆曲线参数a(a>0)的值："))
        b = int(input("请输入椭圆曲线参数b(b>0)的值："))
        p = int(input("请输入椭圆曲线参数p(p为素数)的值："))  #用作模运算

# 条件满足判断
        if (4*(a**3)+27*(b**2))%p == 0:
            print("您输入的参数有误，请重新输入！！！\n")
        else:
            break

# 选点作为G点
    print("在如上坐标系中选一个值为G的坐标")
    G_x = int(input("请输入选取的x坐标值："))
    G_y = int(input("请输入选取的y坐标值："))

# 获取椭圆曲线的阶
    n = get_rank(G_x, G_y, a, b, p)

# user1生成私钥，小key
    key = int(input("请输入私钥小key（<{}）：".format(n)))

# user1生成公钥，大KEY
    KEY_x,kEY_y = get_ng(G_x, G_y, key, a, p)
    print('flag is ',KEY_,KEY_Y)

if __name__ == "__main__":
    ecc_main()
```

p有点大，跑不动

来自 <https://www.jianshu.com/p/c43776370840>

另一个脚本：

```
#!/usr/bin/env python3

# -*- coding:utf-8 -*-

import collections
```

```
import random

EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')

curve = EllipticCurve(
    'secp256k1',
    # Field characteristic.
    p=int(input('p=')),
    # Curve coefficients.
    a=int(input('a=')),
    b=int(input('b=')),
    # Base point.
    g=(int(input('Gx=')),
       int(input('Gy='))),
    # Subgroup order.
    n=int(input('k=')),
    # Subgroup cofactor.
    h=1,
)


# Modular arithmetic ##########################################################

def inverse_mod(k, p):
    """Returns the inverse of k modulo p.

   This function returns the only integer x such that (x * k) % p == 1.

   k must be non-zero and p must be a prime.
   """
    if k == 0:
```

```
        raise ZeroDivisionError('division by zero')

    if k < 0:
        # k ** -1 = p - (-k) ** -1 (mod p)
        return p - inverse_mod(-k, p)

    # Extended Euclidean algorithm.
    s, old_s = 0, 1
    t, old_t = 1, 0
    r, old_r = p, k

    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t

    gcd, x, y = old_r, old_s, old_t

    assert gcd == 1
    assert (k * x) % p == 1

    return x % p


# Functions that work on curve points #############################################

def is_on_curve(point):
    """Returns True if the given point lies on the elliptic curve."""
    if point is None:
        # None represents the point at infinity.
```

```
        return True

    x, y = point

    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0


def point_neg(point):
    """Returns -point."""
    assert is_on_curve(point)

    if point is None:
        # -0 = 0
        return None

    x, y = point
    result = (x, -y % curve.p)

    assert is_on_curve(result)

    return result


def point_add(point1, point2):
    """Returns the result of point1 + point2 according to the group law."""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
```

```
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None

    if x1 == x2:
        # This is the case point1 == point2.
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # This is the case point1 != point2.
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

    x3 = m * m - x1 - x2
    y3 = y1 + m * (x3 - x1)
    result = (x3 % curve.p,
              -y3 % curve.p)

    assert is_on_curve(result)

    return result


def scalar_mult(k, point):
    """Returns k * point computed using the double and point_add algorithm."""
    assert is_on_curve(point)
```

```
    if k < 0:
        # k * point = -k * (-point)
        return scalar_mult(-k, point_neg(point))

    result = None
    addend = point

    while k:
        if k & 1:
            # Add.
            result = point_add(result, addend)

        # Double.
        addend = point_add(addend, addend)

        k >>= 1

    assert is_on_curve(result)

    return result


# Keypair generation and ECDHE ################################################

def make_keypair():
    """Generates a random private-public key pair."""
    private_key = curve.n
    public_key = scalar_mult(private_key, curve.g)
```

```
    return private_key, public_key


private_key, public_key = make_keypair()
print("private key:", hex(private_key))
print("public key: (0x{:x}, 0x{:x})".format(*public_key))
```

解出：

```
C:\Users\cong\Desktop>python 1.py
p=15424654874903
a=16546484
b=4548674875
Gx=6478678675
Gy=5636379357093
k=546768
('private key:', '0x857d0')
public key: (0xcb19fe553fa, 0x50545408eb4)

C:\Users\cong\Desktop>python
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2019, 01:37:19) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x=0xcb19fe553fa
>>> y=0x50545408eb4
>>> print x+y
19477226185390
>>>
```

cyberpeace{19477226185390}