XCTF-攻防世界-密码学crypto-新手练习区-writeup

原创

Ryannn- 于 2019-10-23 22:14:44 发布 7598 收藏 28
分类专栏： 密码学 CTF 文章标签： 密码学 Crypto CTF 攻防世界 WP

目录

0x00 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。

密文：Y3liZXJwZWFjZXtXZWxjb21lX3RvX25ld19Xb3JsZCF9

一眼base64，直接上代码或者工具

```
#!/usr/bin/python
# -*- coding=utf -*-
import base64

cipher = "Y3liZXJwZWFjZXtXZWxjb21lX3RvX25ld19Xb3JsZCF9"
plaintext = base64.b64decode(cipher)
print(plaintext)
```

PS：写Python脚本时遇到一个小坑，如果把py文件命名为base64.py会导致import base64错误，改个文件名即可。

```
AttributeError: module 'base64' has no attribute 'b64decode'
```

0x01 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

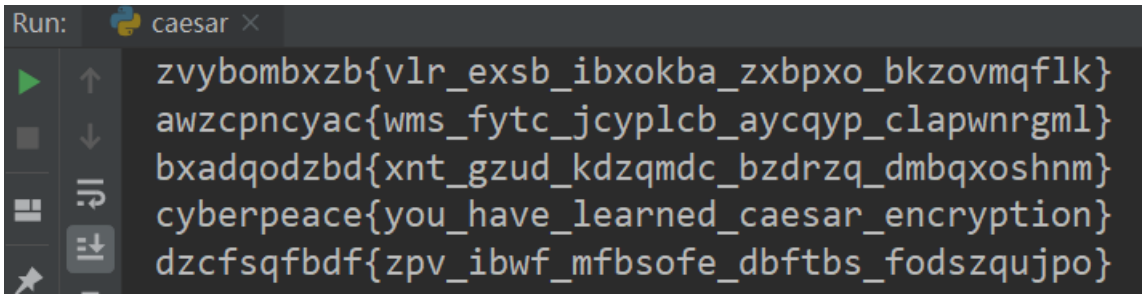
密文：oknqdbqmoq{kag_tmhq_xqmdzqp_omqemd_qzodkbfuaz}

Caesar，凯撒密码

```
#!/usr/bin/python
# -*- coding=utf -*-


def caesar(cipher):
    for j in range(26):
        str_list = list(cipher)
        i = 0
        while i < len(cipher):
            if not str_list[i].isalpha():
                str_list[i] = str_list[i]
            else:
                a = "A" if str_list[i].isupper() else "a"
                str_list[i] = chr((ord(str_list[i]) - ord(a) + j) % 26 + ord(a))
            i = i + 1
        print(''.join(str_list))


if __name__ == '__main__':
    cipher = "oknqdbqmoq{kag_tmhq_xqmdzqp_omqemd_qzodkbfuaz}"
    caesar(cipher)
```



0x02 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{xxxxxxxxx},均为小写)

密文：11 111 010 000 0 1010 111 100 0 00 000 000 111 00 10 1 0 010 0 000 1 00 10 110

Morse摩斯密码

莫尔斯电码表 MADE BY BG1GJP

A	·—	N	—·	1	·— — — —
B	—···	O	— — —	2	··— — —
C	—·—·	P	·— —·	3	···— —
D	—··	Q	— —·—	4	····—
E	·	R	·—·	5	·····
F	··—·	S	···	6	—····
G	— —·	T	—	7	— —···
H	····	U	··—	8	— — —··
I	··	V	···—	9	— — — —·
J	·— — —	W	·— —	0	— — — — —
K	—·—	X	—··—	?	··— —··
L	·—··	Y	—·— —	/	—··—·
M	— —	Z	— —··	。	·—·—·—
()	—·— —·—	—	—···—	@	·— —·—··

此处借鉴一下一位dalao的代码

```
#!/usr/bin/python
# -*- coding=utf -*-

table ={'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': "--..",

        '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-',
        '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.',

        ',': '--..--', '.': '.-.-.-', ':': '---...', ';': '-.-.-.',
        '?': '..--..', '=': '-...-', "'": '.----.', '/': '-..-.',
        '!': '-.-.--', '-': '-....-', '_': '..--.-', '(': '-.--.',
        ')': '-.--.-', '$': '...-..-', '&': '. . . .', '@': '.--.-.'}


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


if __name__ == '__main__':
    file = open(r'D:\CTF\攻防世界\Crypto\Morse.txt', 'r')
    cipher = file.read()
    cipher = cipher.replace('1', '-')
    cipher = cipher.replace('0', '.')
    plaintext = morse(cipher)
    print(plaintext)
```

0x03 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。

密文：ccehgyaefnpeoobe{lcirg}epriec_ora_g

The rail fence cipher栅栏密码

此题有坑，栅栏密码具体操作不止一种，一些在线工具解密也会得到不一样的结果。

栅栏密码加密/解密

明文: cyberpeace{railfence_cipher_gogogo}

栏数: 5

加密 解密

密文: ccehgyaefnpeoobe{lcirg}epriec_ora_g

0x04 转轮机加密

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

密文：

```
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
```

杰斐逊轮转加密视频演示戳这里

首先将每一行按照密钥顺序重新排列，如第二行应至于第一行的位置，以此类推。

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

密钥为：2,3,7,5,13,12,9,1,8,10,4,11,6
密文为：NFQKSEVOQOFNP
```

然后按照密文，将每一行进行循环移位，使得其首位字母与对应的密文相同。

最后就是瞎眼时间，找到通顺的一列明文：fireinthehole

```
< NACZDTRXMJQOYHGVS F UWIKPBEL <
< FHTEQGYXPLOCKBDMA I ZVRNSJUW <
< QGWTHSPYBXIZULVKM R AFDCEONJ <
< KCPMNZQWXYIHFRLAB E UOTSGJVD <
< SXCDERFVBGTYHNUMK I LOPJZQAW <
< EIURYTASBKJDFHGLV N CMXZPQOW <
< VUBMCQWAOIKZGJXPL T DSRFHENY <
< OSFEZWAXJGDLUBVIQ H KYPNTCRM <
< QNOZUTWDCVRJLXKIS E FAPMYGHB <
< OWTGVRSCZQKELMXYI H PUDNAJFB <
< FCUKTEBSXQYIZMJWA O RPLNDVHG <
< NBVCXZQWERTPOIUYA L SKDJFHGM <
< PNYCJBFZDRUSLOQXV E TAMKGHIW <

密文为：NFQKSEVOQOFNP
```

0x05 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。flag格式为cyberpeace{小写的你解出的答案}

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

RSA算法中，d，e，p，q的关系如下：

```
d=e^-1 mod (p-1)(q-1)
```

现已知p，q，e，可用扩展欧几里得算法进行求逆运算：

```
#!/usr/bin/python
# -*- coding=utf -*-


def exgcd(a, n):
    if n == 0:
        return 1, 0
    else:
        x, y = exgcd(n, a % n)
        x, y = y, x - (a // n) * y
        return x, y


def getReverse(a, n):
    re, y = exgcd(a, n)
    while(re < 0):
        re += n
    return re


if __name__ == "__main__":
    p = 473398607161
    q = 4511491
    e = 17
    d = getReverse(e, (p - 1)*(q - 1))
    print('d = ' + str(d))
```

```
d = 125631357777427553
```

0x06 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

本题主要考察工具的使用，附件共2个文件，flag.enc与pubkey.pem

OpenSSL （开放式安全套接层协议）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。

```
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMJjauXD2OQ/+5erCQKPGqxsC/bNPXDr
yigb/+l/vjDdAgMBAAE=
-----END PUBLIC KEY-----
```

祭出kali，自带openssl

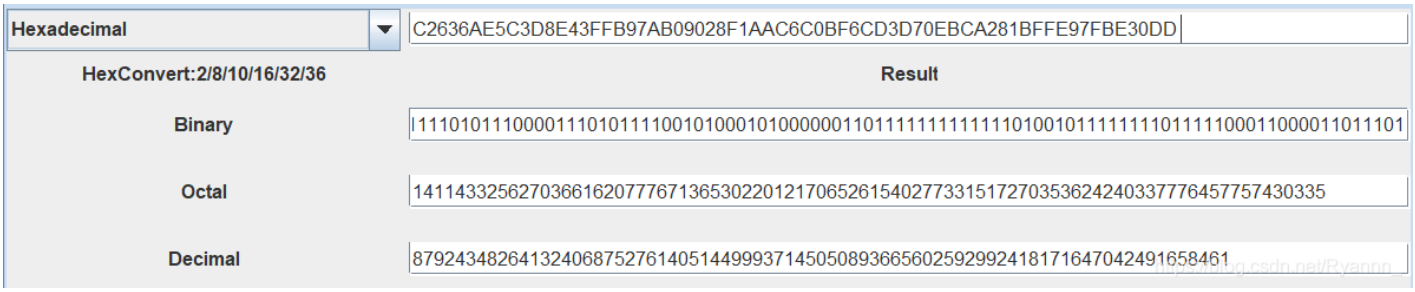
第一步，提取pem文件信息

```
openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem
```

```
root@kali:~/桌面/Normal_RSA# openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem
RSA Public-Key: (256 bit)
Modulus:
    00:c2:63:6a:e5:c3:d8:e4:3f:fb:97:ab:09:02:8f:
    1a:ac:6c:0b:f6:cd:3d:70:eb:ca:28:1b:ff:e9:7f:
    be:30:dd
Exponent: 65537 (0x10001)
Modulus=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMJjauXD2OQ/+5erCQKPGqxsC/bNPXDr
yigb/+l/vjDdAgMBAAE=
-----END PUBLIC KEY-----
```


得到大素数乘积n，此处为16进制，使用工具转换为10进制。

```
Modulus=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD
```



对素数乘积n进行分解，解出素数q、p。大数分解

```
n=879243482641324068752761405144999371450508936656025929924181716470424916584 61

275127860351348928173285174381581152299
319576316814478949870590164193048041239
```

现已知大素数p、q，取随机素数e，使用rsatool生成私钥文件

```
python rsatool.py -o private.pem -e 65537 -p 275127860351348928173285174381581152299 -q 3195763168144789498
```

```
root@kali:~/桌面/Normal_RSA# python rsatool.py -o private.pem -e 65537 -p 2751278
60351348928173285174381581152299 -q 319576316814478949870590164193048041239
Using (p, q) to initialise RSA instance

n =
c2636ae5c3d8e43ffb97ab09028f1aac6c0bf6cd3d70ebca281bffe97fbe30dd

e = 65537 (0x10001)

d =
1806799bd44ce649122b78b43060c786f8b77fb1593e0842da063ba0d8728bf1

p = 275127860351348928173285174381581152299 (0xcefbb2cf7e18a98ebedc36e3e7c3b02b)

q = 319576316814478949870590164193048041239 (0xf06c28e91c8922b9c236e23560c09717)

Saving PEM as private.pem
```

用生成出的私钥文件解密flag

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

0x07 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！”。一种食物,格式为cyberpeace{小写的你解出的答案}

密文：--/.-/-.--/..--.-/-.../././..--.-/..../.-/...-/./..--.-/.-/-./---/-/..../././.-./..--.-/-.././-.-./---/-.././..../..../..../..../.-/.-/.-/.-/.-/-.../.-/.-/-.../-.../-.../.-/.-/-.../-.../.-/.-/.-/.-/.-/.-/.-/.-/-.../.-/.-/-.../.-/-.../.-/.-/.-/.-/.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../-.../.-/.-/.-/-.../-.../.-/.-/-.../.-/.-/.-/.-/-.../.-/-.../.-/.-/-.../.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../.-/.-/.-/-.../.-/.-/-.../.-/-.../-.../.-/.-/-.../-.../-.../.-/-.../.-/.-/.-/-.../.-/-.../.-/-.../-.../.-/.-/.-/-.../-.../.-/-.../.-/.-/.-/-.../.-/.-/-.../.-/.-/.-/.-/-.../-.../.-/-.../-.../.-/.-/-.../-.../.-/.-/-.../.-/.-/-.../.-/.-/.-/-.../.-/.-/-.../.-/.-/-.../.-/-.../.-/.-/-.../-.../.-/-.../.-/.-/.-/.-/-.../-.../.-/-.../-.../.-/-.../.-/.-/-.../-.../.-

emmmm比Morse多了个‘/’，但是没有空格，把‘/’换成空格应该就能用Morse解密了，仍然用上面的morse函数解密。

```
if __name__ == '__main__':
    file = open(r'D:\CTF\攻防世界\Crypto\不仅仅是Morse.txt', 'r')
    cipher = file.read()
    cipher = cipher.replace('/', ' ')
    plaintext = morse(cipher)
    print(plaintext)
```

解密的结果又是个密文

```
may_be_have_another_decodehhhhaaaaabaabbbaabbaaaaaaaabaababaaaaaaabbabaaabbaaabbaabaaaababaabaaabbabaaabaaa
```

培根密码，又名倍康尼密码（英语：Bacon's cipher）是由法兰西斯·培根发明的一种隐写术。加密时，明文中的每个字母都会转换成一组五个英文字母。其转换依靠下表：

A/a	aaaaa	H/h	aabbb	O/o	abbba	V/v	babab
B/b	aaaab	I/i	abaaa	P/p	abbbb	W/w	babba
C/c	aaaba	J/j	abaab	Q/q	baaaa	X/x	babbb
D/d	aaabb	K/k	ababa	R/r	baaab	Y/y	bbaaa
E/e	aabaa	L/l	ababb	S/s	baaba	Z/z	bbaab
F/f	aabab	M/m	abbaa	T/t	baabb		
G/g	aabba	N/n	abbab	U/u	babaa		

```
#!/usr/bin/python
# -*- coding=utf -*-
import re

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'}


def bacon(cipher):
    msg = ''
    codes = re.findall(r'.{5}', cipher)
    for code in codes:
        if code == '':
            msg += ' '
        else:
            UNCODE = dict(map(lambda t: (t[1], t[0]), table.items()))
            msg += UNCODE[code]
    return msg


if __name__ == '__main__':
    cipher = 'aaaaabaabbbaabbaaaaaaaaabaababaaaaaaaabbabaaabbaaabbaabaaaabababaabaaabbabaaabaaabaababbaabbbabaa
    plaintext = bacon(cipher)
    print(plaintext)
```

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。格式为cyberpeace{小写的你解出的答案}

密文（还真是好长 好长）：

```
JiM3NjsmIzEyMjsmIzY5OyYjMTIwOyYjNzk7JiM4MzsmIzU2OyYjMTIwOyYjNzc7JiM2ODsmIzY5OyYjMTE4OyYjNzc7JiM4NDsmIzY1OyY
```

base64->text

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

unicode

```
LzExOS8xMDEvMTA4Lzk5LzExMS8xMDkvMTAxLzExNi8xMTEvOTcvMTE2LzExNi85Ny85OS8xMDcvOTcvMTEwLzEwMC8xMDAvMTAxLzEwMi8
```

再次base64

```
/119/101/108/99/111/109/101/116/111/97/116/116/97/99/107/97/110/100/100/101/102/101/110/99/101/119/111/114/
```

再次unicode

```
welcometoattackanddefenceworld
```

0x09 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。

附件是一个python的.pyc文件，大致可以理解为python程序编译后得到的文件。可以下载uncompyle库后使用uncompyle6将easychallenge.pyc反编译成py文件。

```
uncompyle6 easychallenge.pyc > easychallenge.py
```

```
# uncompyle6 version 3.5.0
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.14+ (default, Dec  5 2017, 15:17:02)
# [GCC 7.2.0]
# Embedded file name: ans.py
# Compiled at: 2018-08-09 11:29:44
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'
# okay decompiling easychallenge.pyc
```

emmmm感觉像是在做逆向题，逆就完事了。

```
#!/usr/bin/python
# -*- coding=utf -*-
import base64


def decode1(s):
    ans = ''
    for i in s:
        x = ord(i) - 25
        x = x ^ 36
        ans += chr(x)
    return ans


def decode2(s):
    ans = ''
    for i in s:
        x = i ^ 36
        x = x - 36
        ans += chr(x)
    return ans


def decode3(ans):
    return base64.b32decode(ans)


if __name__ == "__main__":
    final = 'UC7KOWVXWVNKNIC2XCXKHKK2W5NLBKNOUOSK3LNNVWW3E==='
    flag = decode1(decode2(decode3(final)))
    print(flag)
```

0x0A 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。flag格式为cyberpeace{x+y的值}

这难度系数5颗星有点吓人

```
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)
```

椭圆加密算法（ECC）是一种公钥加密体制，最初由Koblitz和Miller两人于1985年提出，其数学基础是利用椭圆曲线上的有理点构成Abel加法群上椭圆离散对数的计算困难性。公钥密码体制根据其所依据的难题一般分为三类：大素数分解问题类、离散对数问题类、椭圆曲线类。有时也把椭圆曲线类归为离散对数类。

ECC椭圆曲线加密学习笔记

椭圆曲线的奇妙比喻

参考了这位博主的代码

```
import collections


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
```

```
            # 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


EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')
curve = EllipticCurve(
    'secp256k1',
    # Field characteristic.
    p=15424654874903,
    # Curve coefficients.
    a=16546484,
    b=4548674875,
    # Base point.
    g=(6478678675,5636379357093),
    # Subgroup order.
    n=546768,
    # Subgroup cofactor.
    h=1,
)
private_key, public_key = make_keypair()
print("private key:", hex(private_key))
print("public key: (0x{:x}, 0x{:x})".format(*public_key))
print("x + y = " + str(public_key[0] + public_key[1]))
```

```
private key: 0x857d0
public key: (0xcb19fe553fa, 0x50545408eb4)
x + y = 19477226185390
```

0x0B 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。flag为cyberpeace{你解答出的八位大写字母}

密文：88421012204802244040142242024801 22

二进制幂数加密法 编辑 讨论

本词条缺少**概述**、**信息栏**，补充相关内容使词条更完整，还能快速升级，赶紧来编辑吧！

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

二进制幂数加密法就是应用这个原理，由于英文字母只有26个字母，由公式可知，只要2的0、1、2、3、4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来加密。例如

明文：donotpullallyoureggsinonebasket

字母序号：4 15 14 15 20 16 21 12 12 1 12 12 25 15 21 18 5 7 7 19 9 14 15 14 5 2 1 19 11 5 20

由于4=2^2 所以D加密过之后是2；15=2^0+2^1+2^2+2^3所以O加密后是0123。同理得到上述明文的加密后的密文

密文：2 0123/123 0123 24/4 024 23 23/0 23 23/034 0123 024 14/02 012 012 014/03 123 /0123 123 02/1 0 014 013 02 24

其中空格表示字母的间隔，/表示单词的间隔。

然而这题显然不是单纯的二进制幂数加密，因为多了数字8。

有些博主说它是云影密码，实际上好像还是不太一样的。

hint说总共8个字母，密文里恰好7个0分成了8段，每段的数字加起来对应字母就可以了。

```
88421 0 122 0 48 0 2244 0 4 0 142242 0 248 0 122
88421  122     48  2244 4 142242 248 122
23      5  12  12  4 15 14 5
W  E  L  L  D O N E
WELLDONE
```