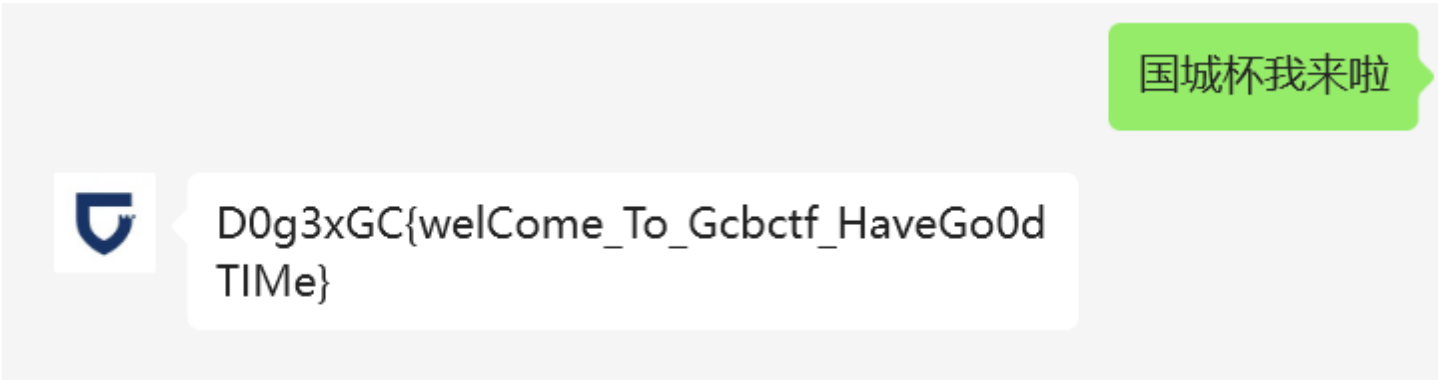
国城杯 MXGA Writeup

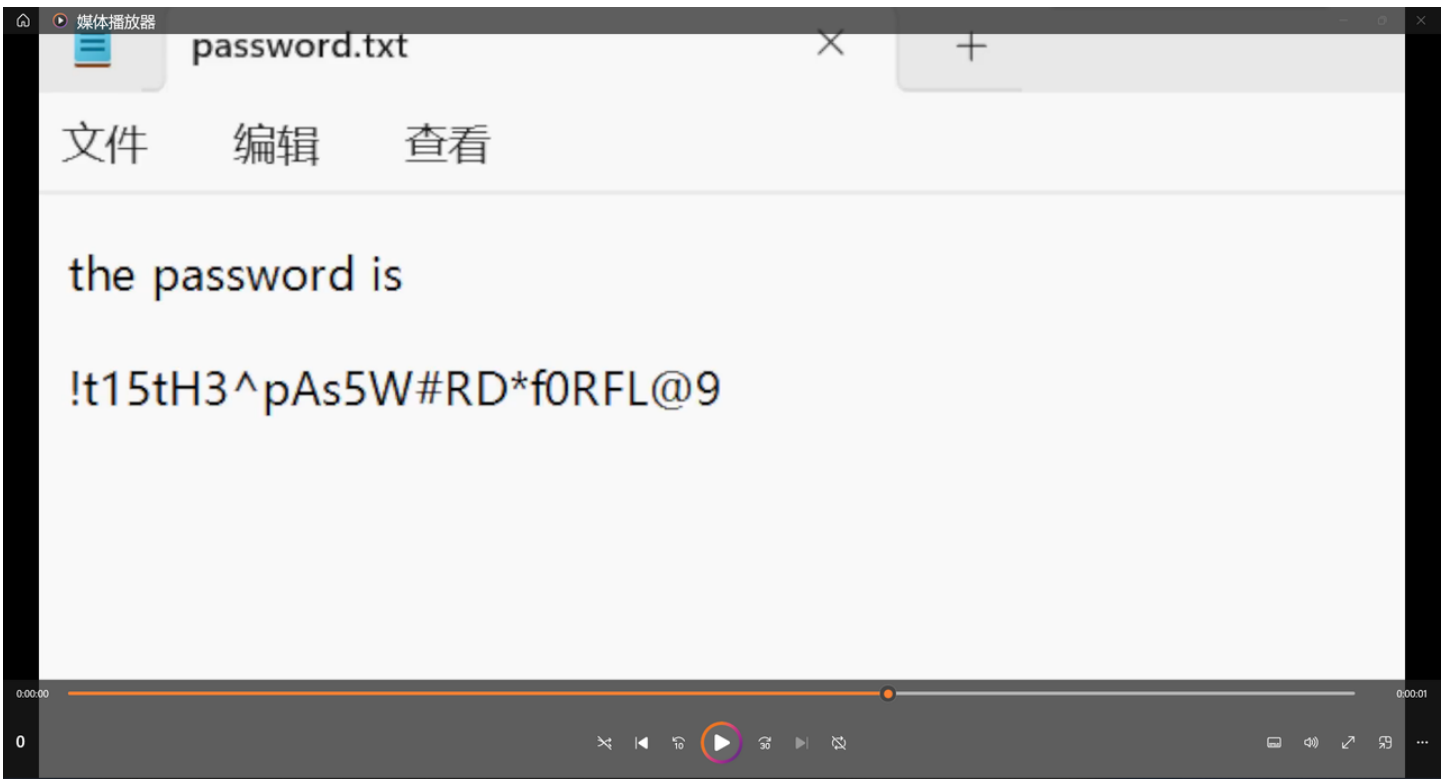
Misc

我是真签到



Tr4ffIc_w1th_Ste90

password.pcapng里只有一段udp流量，导出成.ts播放可以直接看到密码：

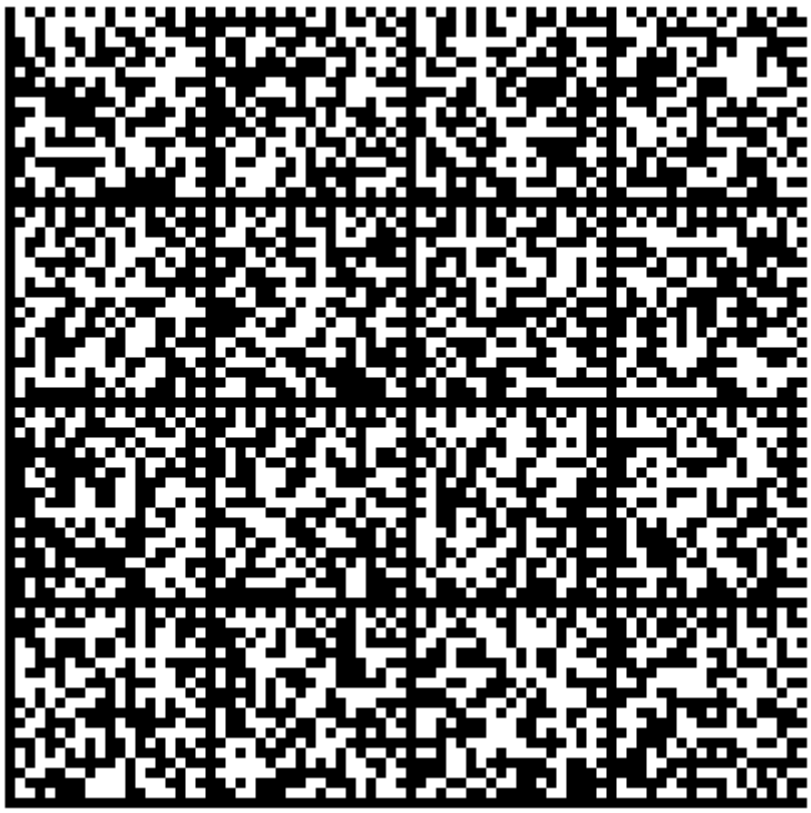


challenge.zip解压拿到加密图片和加密脚本，解密得到一个矩阵码：

```
1 import numpy as np
2 import cv2
```

```python
import os

def decode(input_image, output_folder, seed):
    np.random.seed(seed)  # Set the seed for reproducibility

    # Load the grayscale image
    gray = cv2.imread(input_image, cv2.IMREAD_GRAYSCALE)

    if gray is None:
        print(f"Error: Unable to load image {input_image}")
        return

    # Get the shape of the image
    height, width = gray.shape[:2]

    # Generate the shuffled indices using the same seed
    row_indices = list(range(height))
    col_indices = list(range(width))

    np.random.shuffle(row_indices)
    np.random.shuffle(col_indices)

    # Create an array of original indices by reversing the shuffle operation
    original_row_indices = [0] * len(row_indices)
    original_col_indices = [0] * len(col_indices)

    for i, idx in enumerate(row_indices):
        original_row_indices[idx] = i
    for i, idx in enumerate(col_indices):
        original_col_indices[idx] = i

    # Reorder the rows and columns back to their original order
    gray = gray[original_row_indices, :]
    gray = gray[:, original_col_indices]

    # Convert the grayscale image back to BGR (if necessary)
    restored = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)

    # Create a unique filename based on the seed value
    output_image = os.path.join(output_folder, f"decoded_seed_{seed}.png")

    # Ensure the output directory exists
    os.makedirs(output_folder, exist_ok=True)

    # Save the restored image
    cv2.imwrite(output_image, restored)
    print(f"Decoded image with seed {seed} saved as {output_image}")
```

```python

def main():
    input_image = 'encoded.png'  # Input encoded image path
    output_folder = 'decoded_images'   # Output folder where images will be saved

    # Loop over seeds from 50 to 70 inclusive
    for seed in range(50, 71):
        decode(input_image, output_folder, seed)

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



扫描得到：

```
I randomly found a word list to encrypt the flag. I only remember that Wikipedia said this word list is similar to the NATO phonetic alphabet.

crumpled chairlift freedom chisel island dashboard crucial kickoff crucial chairlift drifter classroom highchair cranky clamshell edict drainage fallout clamshell chatter chairlift goldfish chopper eyetooth endow chairlift edict eyetooth deadbolt fallout egghead chisel eyetooth cranky crucial deadbolt chatter chisel egghead chisel crumpled eyetooth clamshell deadbolt chatter chopper eyetooth classroom chairlift fallout drainage klaxon
```

字典是PGP_wordlist，对应hex转字符拿到flag：

```
pgp_wordlist = {
    'aardvark': '00',
    'absurd': '01',
    'accrue': '02',
    'acme': '03',
    'adrift': '04',
    'adult': '05',
    'afflict': '06',
    'ahead': '07',
    'aimless': '08',
    'Algol': '09',
    'allow': '0A',
    'alone': '0B',
    'ammo': '0C',
    'ancient': '0D',
    'apple': '0E',
    'artist': '0F',
    'assume': '10',
    'Athens': '11',
    'atlas': '12',
    'Aztec': '13',
    'baboon': '14',
    'backfield': '15',
    'backward': '16',
    'banjo': '17',
    'beaming': '18',
    'bedlamp': '19',
    'beehive': '1A',
    'beeswax': '1B',
    'befriend': '1C',
    'Belfast': '1D',
    'berserk': '1E',
    'billiard': '1F',
    'bison': '20',
    'blackjack': '21',
    'blockade': '22',
    'blowtorch': '23',
    'bluebird': '24',
    'bombast': '25',
    'bookshelf': '26',
    'brackish': '27',
    'breadline': '28',
    'breakup': '29',
    'brickyard': '2A',
```

```
    'briefcase': '2B',
    'Burbank': '2C',
    'button': '2D',
    'buzzard': '2E',
    'cement': '2F',
    'chairlift': '30',
    'chatter': '31',
    'checkup': '32',
    'chisel': '33',
    'choking': '34',
    'chopper': '35',
    'Christmas': '36',
    'clamshell': '37',
    'classic': '38',
    'classroom': '39',
    'cleanup': '3A',
    'clockwork': '3B',
    'cobra': '3C',
    'commence': '3D',
    'concert': '3E',
    'cowbell': '3F',
    'crackdown': '40',
    'cranky': '41',
    'crowfoot': '42',
    'crucial': '43',
    'crumpled': '44',
    'crusade': '45',
    'cubic': '46',
    'dashboard': '47',
    'deadbolt': '48',
    'deckhand': '49',
    'dogsled': '4A',
    'dragnet': '4B',
    'drainage': '4C',
    'dreadful': '4D',
    'drifter': '4E',
    'dropper': '4F',
    'drumbeat': '50',
    'drunken': '51',
    'Dupont': '52',
    'dwelling': '53',
    'eating': '54',
    'edict': '55',
    'egghead': '56',
    'eightball': '57',
    'endorse': '58',
    'endow': '59',
```

```
    'enlist': '5A',
    'erase': '5B',
    'escape': '5C',
    'exceed': '5D',
    'eyeglass': '5E',
    'eyetooth': '5F',
    'facial': '60',
    'fallout': '61',
    'flagpole': '62',
    'flatfoot': '63',
    'flytrap': '64',
    'fracture': '65',
    'framework': '66',
    'freedom': '67',
    'frighten': '68',
    'gazelle': '69',
    'Geiger': '6A',
    'glitter': '6B',
    'glucose': '6C',
    'goggles': '6D',
    'goldfish': '6E',
    'gremlin': '6F',
    'guidance': '70',
    'hamlet': '71',
    'highchair': '72',
    'hockey': '73',
    'indoors': '74',
    'indulge': '75',
    'inverse': '76',
    'involve': '77',
    'island': '78',
    'jawbone': '79',
    'keyboard': '7A',
    'kickoff': '7B',
    'kiwi': '7C',
    'klaxon': '7D',
    'locale': '7E',
    'lockup': '7F',
    'merit': '80',
    'minnow': '81',
    'miser': '82',
    'Mohawk': '83',
    'mural': '84',
    'music': '85',
    'necklace': '86',
    'Neptune': '87',
    'newborn': '88',
```

```
    'nightbird': '89',
    'Oakland': '8A',
    'obtuse': '8B',
    'offload': '8C',
    'optic': '8D',
    'orca': '8E',
    'payday': '8F',
    'peachy': '90',
    'pheasant': '91',
    'physique': '92',
    'playhouse': '93',
    'Pluto': '94',
    'preclude': '95',
    'prefer': '96',
    'preshrunk': '97',
    'printer': '98',
    'prowler': '99',
    'pupil': '9A',
    'puppy': '9B',
    'python': '9C',
    'quadrant': '9D',
    'quiver': '9E'
}

# 要查找的单词
search_word = 'crumpled chairlift freedom chisel island dashboard crucial kickoff crucial chairlift drifter classroom highchair cranky clamshell edict drainage fallout clamshell chatter chairlift goldfish chopper eyetooth endow chairlift edict eyetooth deadbolt fallout egghead chisel eyetooth cranky crucial deadbolt chatter chisel egghead chisel crumpled eyetooth clamshell deadbolt chatter chopper eyetooth classroom chairlift fallout drainage klaxon'
encrypted_wordlist = search_word.split(" ")


flag = ""
for word in encrypted_wordlist:
    hex = pgp_wordlist[word]
    flag += chr(int(hex,base=16))
print(flag)
```

D0g3xGC{C0N9rA7ULa710n5_Y0U_HaV3_ACH13V3D_7H15_90aL}

Reverse

Crush's_secret

查看程序发现有 `.SMC` 段，交叉引用VirtualProtect，找到smc解密的函数。

最后一个 `VirtualProtect` 下断点，动态调试，在内存中F5反编译解密后的函数。经过分析是魔改过的XTEA，修改了加密的运算。

```
int __cdecl sub_418000(_DWORD *a1, int a2, int a3)
{
  int v3; // eax
  int v4; // edx
  int v5; // edx
  int v7; // [esp+D4h] [ebp-44h]
  int v8; // [esp+E0h] [ebp-38h]
  unsigned int i; // [esp+ECh] [ebp-2Ch]
  unsigned int v10; // [esp+F8h] [ebp-20h]
  unsigned int v11; // [esp+104h] [ebp-14h]

  sub_41136B();
  if ( a2 > 1 )
  {
    v8 = 52 / a2 + 6;
    v10 = 0;
    v11 = a1[a2 - 1];
    do
    {
      v10 -= 1640531527;
      v7 = (v10 >> 2) & 3;
      for ( i = 0; i < a2 - 1; ++i )
      {
        v3 = ((v11 ^ *(_DWORD *)(a3 + 4 * (v7 ^ i & 3))) + (a1[i + 1] ^ v10)) ^ (((16 * v11) ^ (a1[i + 1] >> 3))
                                                                                + ((4 * a1[i + 1]) ^ (v11 >> 5)));
        v4 = a1[i];
        a1[i] = v3 + v4;
        v11 = v3 + v4;
      }
      v5 = (((v11 ^ *(_DWORD *)(a3 + 4 * (v7 ^ i & 3))) + (*a1 ^ v10)) ^ (((16 * v11) ^ (*a1 >> 3))
                                                                          + ((4 * *a1) ^ (v11 >> 5))))
         + a1[a2 - 1];
      a1[a2 - 1] = v5;
      v11 = v5;
      --v8;
    }
    while ( v8 );
  }
  return sub_411280();
}
```

```c
#include<stdio.h>
void modi_xtea_decry(unsigned int *data, unsigned int *key)
{
    unsigned int d1 = data[0], d2 = data[1];
    unsigned int delta = 0x9e3779b9;
    unsigned int number = delta * 32;
    for (int i = 0; i < 32; i++)
    {
        d2 -= (((d1<<4)^(d1>>3)) + ((d1<<2)^(d1>>5))) ^ (((d1 ^ key[((number>>2) & 3)^ 1 & 3])) + (number ^ d1));
        d1 -= (((d2<<4)^(d2>>3)) + ((d2<<2)^(d2>>5))) ^ (((d2 ^ key[((number>>2) & 3)^ 0 & 3])) + (number ^ d2));
        number -= delta;
    }
    data[0] = d1;
    data[1] = d2;
}
int main()
{
    unsigned int key[] = {0x5201314,0x52013140,0x5201314,0x52013140};
```

```
    unsigned int v11[13];
    v11[0] = 0x5A764F8A;
    v11[1] = 0x5B0DF77;
    v11[2] = 0xF101DF69;
    v11[3] = 0xF9C14EF4;
    v11[4] = 0x27F03590;
    v11[5] = 0x7DF3324F;
    v11[6] = 0x2E322D74;
    v11[7] = 0x8F2A09BC;
    v11[8] = 0xABE2A0D7;
    v11[9] = 0xC2A09FE;
    v11[10] = 0x35892BB2;
    v11[11] = 0x53ABBA12;
    v11[12] = 0;
    for(int i = 0; i < 12; i += 2)
        modi_xtea_decry(v11+i,key);
    printf("%s",(char*)v11);
    return 0;
}
```

round

反编译apk文件，先分析 `MainActivity` 。对输入的用户名进行 `encodeToBase64` ，拿编码后的字符串去cyberchef解码，结果还是乱码。果断用frida来hook这个函数，先随便输入一个字符串看一下编码结果，发现是编码后每4个字符的中间两个交换位置。

```
import frida
import base64
import sys

jscode = """
Java.perform(function(){
    var mp = Java.use("com.example.demo.MakePath");
    var str = Java.use("java.lang.String");
    mp.encodeToBase64.overload("java.lang.String").implementation = function(string){
        console.log("result: " + this.encodeToBase64(str.$new("thisisatextforbase64encode")));
        return this.encodeToBase64(string);
    }
})
"""

def on_message(message, data):
    if message['type'] == 'send':
```

```
        print("[+] {0}".format(message['payload']))
    else:
        print(str(message))

process = frida.get_usb_device(-1).attach("Demo")
print('[*] Attached')
script = process.create_script(jscode)
script.on('message', on_message)
print('[*] Running')
print("Stanard:" + base64.b64encode(b"thisisatextforbase64encode").decode())
script.load()
sys.stdin.read()

'''
[*] Attached
[*] Running
Stanard:dGhpc2lzYXRleHRmb3JiYXNlNjRlbmNvZGU=
result: dhGpcl2zYRXleRHmbJ3iYNXlNRjlbNmvZUG=
'''
```

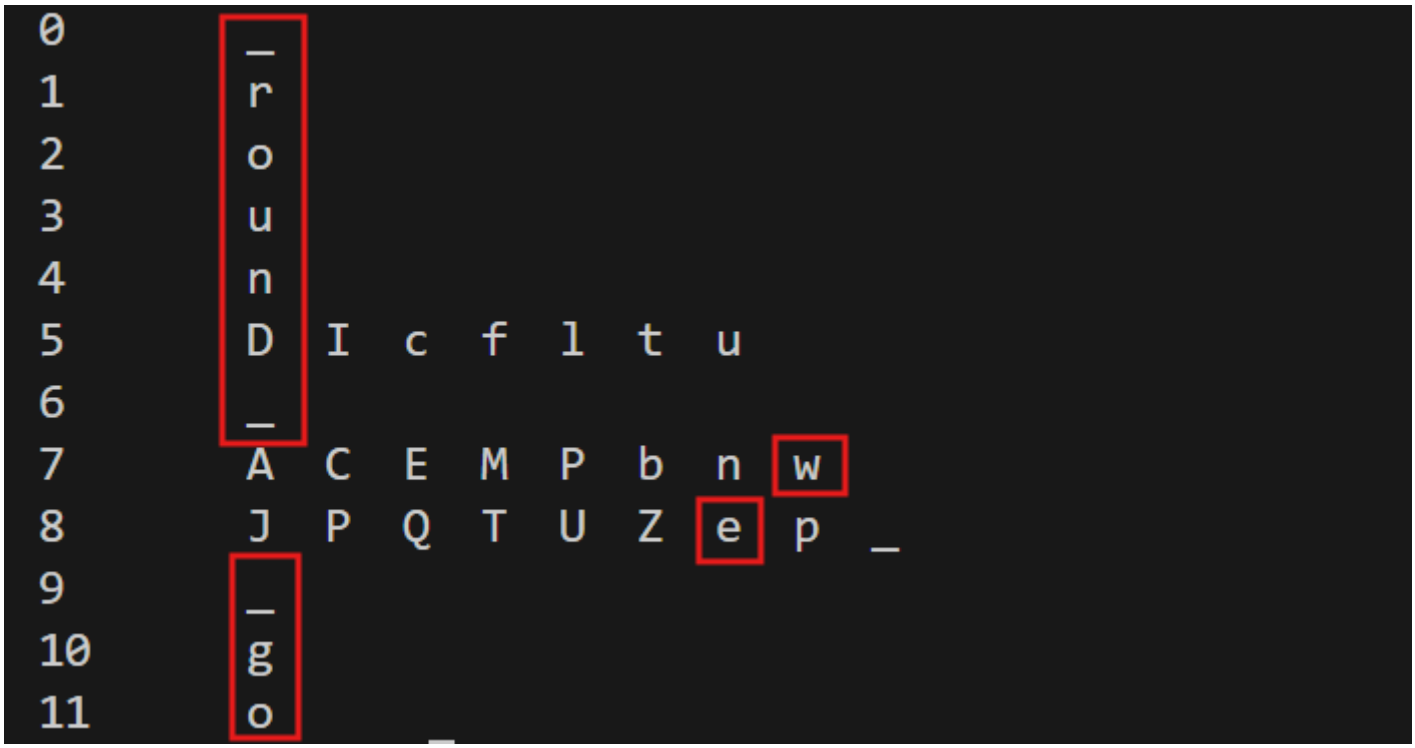
之后可以解码出用户名：round_and

password部分是一个简单的vm，一共12个字符，考虑爆破

```
iArr = []
key = b"c9m1bRmfY5Wk"
for i in range(1024):
    iArr.append((1023-i)^key[i%len(key)])

result = [352, 646, 752, 882, 65, 0, 122, 0, 0, 7, 350, 360]

table = b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_'

rip = 33
lst = [33,0,0,0,0,0,0,0,0,0,0,0,0]
for i2 in range(12):
    if i2==8:
        index = 7
    elif i2 == 9:
        index = 6
    elif i2 == 10:
        index = 0
    else:
        index = 0
    rip = lst[index]
    count = 0
```

```
    print(str(i2)+"\t",end="")

    for charAt in table:
        char = charAt
        i = rip
        for i3 in range(32):
            i4 = (((iArr[i] ^ charAt) % 5) + 5) % 5
            if i4 == 0:
                charAt = (((charAt + iArr[i]) % 1024) + 1024) % 1024
                i = (i + charAt) % 1024
            elif i4 == 1:
                charAt = (((charAt - iArr[i]) % 1024) + 1024) % 1024
                i = (i + charAt) % 1024
            elif (i4 == 2):
                charAt = (iArr[i] ^ charAt) % 1024
                i = (i + charAt) % 1024
            elif (i4 == 3):
                charAt = (charAt >> 3) % 1024
                i = (i + charAt) % 1024
            elif (i4 == 4):
                charAt = (charAt << 3) % 1024
                i = (i + charAt) % 1024
            else:
                print("wrong")

        if result[i2] == charAt:
            lst[count] = i
            count += 1
            print(chr(char),end="  ")

    print("")
```

后面有多解的情况，手动慢慢调整，最终可以得到完整的password



按照apk中主活动的提示，把用户名和密码连起来，套上flag格式即可。

Web

调查问卷

D0g3xGC{Thanks_for_your_participation}

Crypto

Curve

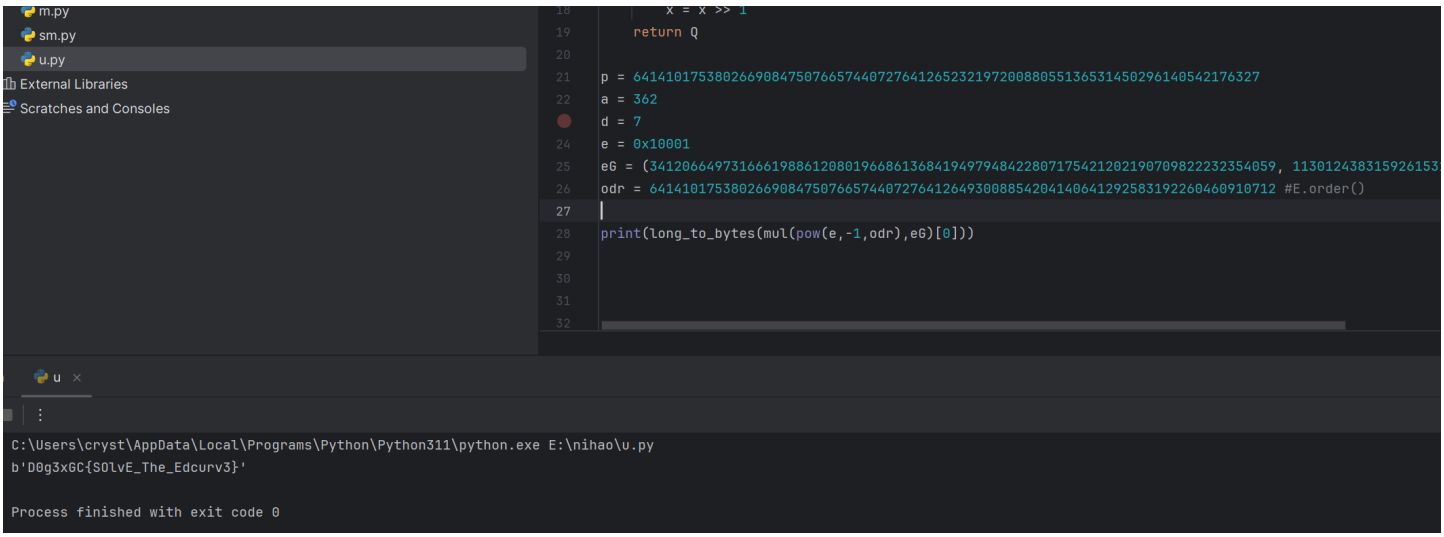
给定的椭圆曲线的加法/乘法法则是在Twisted Edwards曲线上进行（可以通过解析加法/乘法法则进行，assert亦提示了其方程形式ax^2+y^2 = 1+dx^2y^2(mod p)）。

将该曲线参数与其点的x坐标依次变换为Montgomery曲线、Weierstrass曲线，在Sage中构造EllipticCurve求曲线的阶，最终将结果点乘以e在E.order()下的逆元即可获得原点。

hint：Crypto秉持优先使用现成代码的原则(*变换不改变椭圆曲线的阶，将"私钥"代回到Twisted Edwards的add和mul函数中使用比乘了一个Weierstrass的点再把坐标变换回去方便多了

In [166]:
```
from Crypto.Util.number import *

p = 641410175380266908475076657440727641265232197200880551365314502961405421763 27
a = 362
d = 7
e = 0x10001

x2, y2 =  (341206649731666198861208019668613684194979484228071754212021907090822232354059, 113012438315926153126244574438832835294675323900282167
assert (a*x2^2+y2^2)%p==(1+d*x2^2*y2^2)%p

#一阶段
A = 2*(a+d)/(a-d)
B = 4/(a-d)

#二阶段
aa = (3-A^2)/(3*B^2)
bb = (2*A^3-9*A) / (27*B^3)

E = EllipticCurve(Zmod(p), [0, 0, 0, aa, bb])
d = int(pow(e, -1, E.order()))
print(E.order())
```

```
641410175380266908475076657440727641264930088542041406412925831922604609107 12
```



Ez_sign

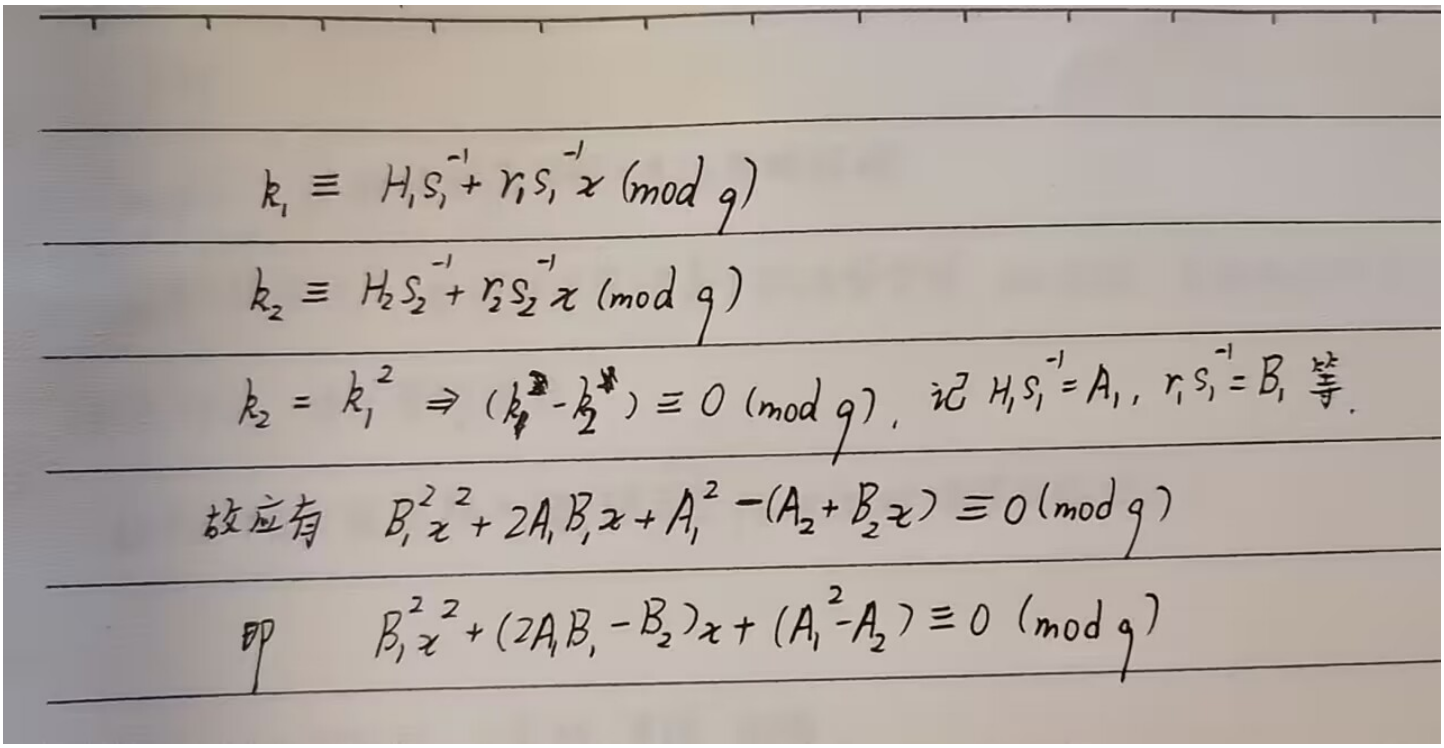
前言：苯宝宝脑子犯了三次迷糊，导致十分钟的题被拖了三个小时.sage

本题分为两部分

第一部分使用DSA对两个H进行了签名，但两次签名的随机数满足k2=k1^2，需要恢复私钥x

第二部分是一个平方和问题

第一部分懒得（也不会）敲代码表示了，直接上草稿



最终用sage解同余式方程，第二个解可以被decode，拿到e（sage中变量A,B与草稿对应关系相反，凑合下=-=）

In [196]:
```
from Crypto.Util.number import *
(H1, r1, s1) = 65978740188354568581745722185285422664454132457 1, 33487845286497881906193099706506193744946434541 1, 28211979327315621449743360 3
(H2, r2, s2) = 15646741452410031387842179839643308145620159983 3, 58411455669950911169533756554182920533694036035 4, 82737152224092106679047704 85
q = 82939641117154047558775576286620318410119523820 7

ss1 = int(pow(s1,-1,q))
ss2 = int(pow(s2,-1,q))

A1 = r1 * ss1
A2 = r2 * ss2
B1 = H1 * ss1
B2 = H2 * ss2

P.<x>=Zmod(q)[]
eq = A1^2*x^2 + (2*A1*B1-A2)*x + (B1^2-B2)
print(eq.roots())
```

```
[(91973915966463187834053272623425597244095846333, 1), (1865444199836044046649, 1)]
```

In [197]:
```
long_to_bytes(int(1865444199836044046649))
```

Out[197]:
```
b'e = 44519'
```

第二部分求解参考了ask.sagemath.org上的对平方和问题多解的求解思路（divisors）

代码非常干净利落，逐个尝试找(p,q)是两个质数的组合就可以丢进最终的RSA里求解

In [167]:
```
def all_two_squares(n):
    return [(abs(d[0]),abs(d[1])) for d in divisors(GaussianIntegers()(n)) if norm(d)==n]

probs = all_two_squares(C)
for p,q in probs:
    if not isPrime(int(p)) or not isPrime(int(q)):
        continue
    e = 44519
    c = 189477930083641543660829910468779775624485491869430437563263657511169362247521
    n = p * q
    d = pow(e,-1,(p-1)*(q-1))
    print(long_to_bytes(int(pow(c,d,n))))
```

```
b'DOg3xGC{EZ_DSA_@nd_COmplex_QAQ}'
b'DOg3xGC{EZ_DSA_@nd_COmplex_QAQ}'
```

BabyRSA

真·BabyRSA

实际上本题是RSA的衍生版Schmidt-Samoa密码体系

该给的私钥都给了，一点点数论就可以爆出来n，随后幂模收工

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

n = 539403894871945779827202174061302970341082455928364137444962844359039924160163196863639732747261316352083923762760392271
d = 581697553864087293946688319478567570604074231260149287054470584683555488615694525227343051883880177643210187704351927671
c = 823639350806888284036878164074142451901975207632747913363218099385553527292923725117507208746367331703187838649048604021

pq = gcd( *integers: pow(2,n*d,n)-2,n)
print(long_to_bytes(pow(c,d,pq)))
```