# Crypto02

题目：

```
# coding: utf-8
#!/usr/bin/env python2

import gmpy2
import random
import binascii
from hashlib import sha256
from sympy import nextprime
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import long_to_bytes
from FLAG import flag
#flag = 'wdflag{123}'

def victory_encrypt(plaintext, key):
    key = key.upper()
    key_length = len(key)
    plaintext = plaintext.upper()
    ciphertext = ''

    for i, char in enumerate(plaintext):
        if char.isalpha():
            shift = ord(key[i % key_length]) - ord('A')
            encrypted_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
            ciphertext += encrypted_char
        else:
            ciphertext += char

    return ciphertext

victory_key = "WANGDINGCUP"
victory_encrypted_flag = victory_encrypt(flag, victory_key)

p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
a = 0
b = 7
xG = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
yG = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
G = (xG, yG)
n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
h = 1
zero = (0,0)

dA = nextprime(random.randint(0, n))

if dA > n:
    print("warning!!")

def addition(t1, t2):
    if t1 == zero:
```

```
        return t2
    if t2 == zero:
        return t2
    (m1, n1) = t1
    (m2, n2) = t2
    if m1 == m2:
        if n1 == 0 or n1 != n2:
            return zero
        else:
            k = (3 * m1 * m1 + a) % p * gmpy2.invert(2 * n1 , p) % p
    else:
        k = (n2 - n1 + p) % p * gmpy2.invert((m2 - m1 + p) % p, p) % p
    m3 = (k * k % p - m1 - m2 + p * 2) % p
    n3 = (k * (m1 - m3) % p - n1 + p) % p
    return (int(m3),int(n3))

def multiplication(x, k):
    ans = zero
    t = 1
    while(t <= k):
        if (k &t )>0:
            ans = addition(ans, x)
        x = addition(x, x)
        t <<= 1
    return ans

def getrs(z, k):
    (xp, yp) = P
    r = xp
    s = (z + r * dA % n) % n * gmpy2.invert(k, n) % n
    return r,s

z1 = random.randint(0, p)
z2 = random.randint(0, p)
k = random.randint(0, n)
P = multiplication(G, k)
hA = multiplication(G, dA)
r1, s1 = getrs(z1, k)
r2, s2 = getrs(z2, k)

print("r1 = {}".format(r1))
print("r2 = {}".format(r2))
print("s1 = {}".format(s1))
print("s2 = {}".format(s2))
print("z1 = {}".format(z1))
print("z2 = {}".format(z2))

key = sha256(long_to_bytes(dA)).digest()
cipher = AES.new(key, AES.MODE_CBC)
iv = cipher.iv
encrypted_flag = cipher.encrypt(pad(victory_encrypted_flag.encode(),
AES.block_size))
encrypted_flag_hex = binascii.hexlify(iv + encrypted_flag).decode('utf-8')

print("Encrypted flag (AES in CBC mode, hex):", encrypted_flag_hex)
```

```
# output
# r1 =
76712729228617953759327460769502934288442352683563219334681162937413283736816
# r2 =
76712729228617953759327460769502934288442352683563219334681162937413283736816
# s1 =
61142716522536931000933884258275974451672976799526648226417247076297644105637
# s2 =
11032531284466899369548757218026209335943031109867937743543229694802999765038
# z1 =
98842149708744845426265508894541731680137086618921535447693509433269270288237
# z2 =
78136168258634736524120345789104208327951990634981072456222624757958532742853
# ('Encrypted flag (AES in CBC mode, hex):',
u'12d5174f2548c179287c7a2ce98a60b20a2dea24611a10d05a8dda9e7bda5f00e260e0c75e62ef6
cb1e3341361ddb36b665b471be124ae3a9271da69a21ce8d6')
```

解答：

一步一步分析即可，首先逆向解出k和dA，这个直接做运算即可

然后把三轮的r和s放进去

将s分为两部分 取前十六位作为初始化的向量 后面是正常的加密的部分

然后计算私钥 `dA` 对应的 SHA-256 哈希值作为密钥 `key_temp`。使用CBC 模式，以 `key_temp` 为密钥，`flag_1` 为初始化向量，对 `enflag` 进行解密，得到 `decrypted_flag`。

最后用题目提供的解密函数进行解密即可，得到flag

py：

```
from gmpy2 import *
p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
a = 0
b = 7
xG = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
yG = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
G = (xG, yG)
n = 0xfffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
h = 1
zero = (0,0)

r1=76712729228617953759327460769502934288442352683563219334681162937413283736816

s1 =
61142716522536931000933884258275974451672976799526648226417247076297644105637
s2 =
11032531284466899369548757218026209335943031109867937743543229694802999765038
z1 =
98842149708744845426265508894541731680137086618921535447693509433269270288237
z2 =
78136168258634736524120345789104208327951990634981072456222624757958532742853

k=(gmpy2.invert(s1-s2,n)*(z1-z2)+n)%n
dA=gmpy2.invert(r1,n)*(k*s1-z1)%n
```

```
s=u'12d5174f2548c179287c7a2ce98a60b20a2dea24611a10d05a8dda9e7bda5f00e260e0c75e62e
f6cb1e3341361ddb36b665b471be124ae3a9271da69a21ce8d6'
ans=binascii.unhexlify(s)
flag_1=ans[:16]
enflag=ans[16:]
key_temp = sha256(long_to_bytes(dA)).digest()
cipher = AES.new(key_temp, AES.MODE_CBC,flag_1)
decrypted_flag = unpad(cipher.decrypt(enflag), AES.block_size)

def victory_decrypt(ciphertext, key):
    key = key.upper()
    key_length = len(key)
    plaintext = ''

    for i, char in enumerate(ciphertext):
        if char.isalpha():
            shift = ord(key[i % key_length]) - ord('A')
            decrypted_char = chr((ord(char) - ord('A') - shift) % 26 + ord('A'))
            plaintext += decrypted_char
        else:
            plaintext += char

    return plaintext

victory_encrypted_flag = decrypted_flag.decode()
victory_key = "WANGDINGCUP"
flag = victory_decrypt(victory_encrypted_flag, victory_key)
print(flag.lower())
```

解得：`wdflag{d4d98e3c6224cb3f641483f31d33ce58}`