赛题名称：Crypto2

# 解题步骤（WriteUp）

**第一步：**

p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f

a = 0

b = 7

xG = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798

yG = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

G = (xG, yG)

n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141

h = 1

zero = (0,0)

这段代码利用椭圆曲线加密（ECC）算法生成了一些参数，并用AES加密生成了一个密文形式的flag。victory\_encrypt函数使用了一个类似维吉尼亚加密的简单移位加密方法对flag进行了初步加密。最终，经过AES-CBC模式的加密，生成了以hex编码的加密flag输出。

**第二步：**我们可以利用椭圆曲线数字签名算法（ECDSA）中的两个签名 r1, s1, z1 和 r2, s2, z2 来恢复出 dA，因为它们都使用了相同的随机数 k。

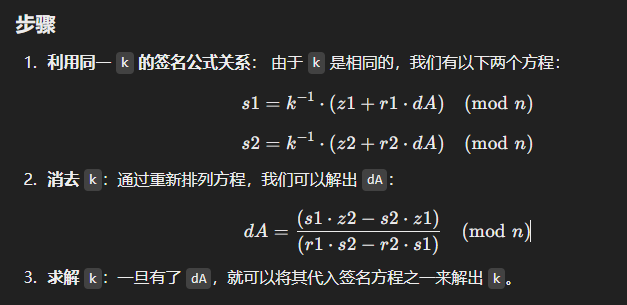
在ECDSA中，签名的公式如下：



其中：

* s 是签名的一部分，
* k 是签名过程中的随机数（在这题中，r1 和 r2 对应的 k 是相同的），
* z 是消息摘要，
* r 是签名中的椭圆曲线点的横坐标，
* dA 是私钥，
* n 是椭圆曲线的阶数。

给定相同的 k 值，我们有两个不同的签名 (r1, s1, z1) 和 (r2, s2, z2)。我们可以通过以下步骤恢复 k 和 dA。



**第三步：**代码解出dA

from gmpy2 import invert

# 椭圆曲线的阶数

n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141

# 已知参数

r1 = 68097957214892959097808634206491566533643222966619139257346810421081585723930

r2 = 68097957214892959097808634206491566533643222966619139257346810421081585723930

s1 = 67249409472936638291163196053877339220525780634461864450594029472181758074828

s2 = 73373457041635669316699455116837076942349235043285345596412317434825715497785

z1 = 80195815881183635999191380380485539364733792579032143197993380599095251892429

z2 = 3634916156299659718916511729962752254806535817011719083804498430258130628828

# 计算 dA

numerator = (s1 \* z2 - s2 \* z1) % n

denominator = (r1 \* s2 - r2 \* s1) % n

dA = (numerator \* invert(denominator, n)) % n

print("Recovered dA:", dA)

……

**第N步：解出AES**

import binascii

from hashlib import sha256

from Crypto.Cipher import AES

from Crypto.Util.Padding import unpad

from Crypto.Util.number import long\_to\_bytes

# 使用之前恢复的 dA 计算 AES 密钥

key = sha256(long\_to\_bytes(dA)).digest()

# 加载已知的密文（在加密输出中）

encrypted\_flag\_hex = "507be52e0721bba75c9dc23180a78de9af0cf21842dd27e16ae27b43de0e99b5a6bf661ef5ebf56d9d7bf9c6ed207a9d753aa94adfa51cbc3d4cb234eb3f177d"

encrypted\_flag = binascii.unhexlify(encrypted\_flag\_hex)

# 提取初始化向量（IV）和密文

iv = encrypted\_flag[:16]

ciphertext = encrypted\_flag[16:]

# 使用AES CBC模式解密

cipher = AES.new(key, AES.MODE\_CBC, iv)

decrypted\_victory\_encrypted\_flag = unpad(cipher.decrypt(ciphertext), AES.block\_size)

# 将 decrypted\_victory\_encrypted\_flag 转换为字符串

decrypted\_victory\_encrypted\_flag = decrypted\_victory\_encrypted\_flag.decode()

print("Decrypted victory encrypted flag:", decrypted\_victory\_encrypted\_flag)

# 解密 victory\_encrypted\_flag（维吉尼亚加密的逆过程）

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) % 26 + ord('A'))

            plaintext += decrypted\_char

        else:

            plaintext += char

    return plaintext

# 定义 victory\_key

victory\_key = "WANGDINGCUP"

# 使用 victory\_key 对 decrypted\_victory\_encrypted\_flag 解密

flag = victory\_decrypt(decrypted\_victory\_encrypted\_flag, victory\_key)

print("Recovered flag:", flag)

最终得出flag

Decrypted victory encrypted flag: SDSRDO{6F0PAA686M480WU0EN9143H85P42N046}

Recovered flag: WDFLAG{6D0AEA686E480CF0EA9143B85A42A046}