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：

$$s = k^{-1} \cdot (z + r \cdot dA) \pmod{n}$$

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`。

步骤

1. **利用同一 `k` 的签名公式关系：** 由于 `k` 是相同的，我们有以下两个方程：

$$s1 = k^{-1} \cdot (z1 + r1 \cdot dA) \pmod{n}$$

$$s2 = k^{-1} \cdot (z2 + r2 \cdot dA) \pmod{n}$$

2. **消去 `k`：** 通过重新排列方程，我们可以解出 `dA`：

$$dA = \frac{(s1 \cdot z2 - s2 \cdot z1)}{(r1 \cdot s2 - r2 \cdot s1)} \pmod{n}$$

3. **求解 `k`：** 一旦有了 `dA`，就可以将其代入签名方程之一来解出 `k`。

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 =
68097957214892959097808634206491566533643222966619139257346810421081585
723930
r2 =
68097957214892959097808634206491566533643222966619139257346810421081585
723930
s1 =
67249409472936638291163196053877339220525780634461864450594029472181758
074828
s2 =
73373457041635669316699455116837076942349235043285345596412317434825715
497785
z1 =
80195815881183635999191380380485539364733792579032143197993380599095251
892429
```

```
z2 =
36349161562996597189165117299627522548065358170117190838044984302581306
28828
```

```
# 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 =
"507be52e0721bba75c9dc23180a78de9af0cf21842dd27e16ae27b43de0e99b5a6bf66
1ef5ebf56d9d7bf9c6ed207a9d753aa94adfa51cbc3d4cb234eb3f177d"
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}