# Writeup

程序运行依赖与gmp库和openssl的库，不过也都挺常规的，对应的依赖很容易就能安装。程序很简单，就是输入4组数据进行验证。

逆向出整个程序的流程，IDA里看起来很难受，可能是因为贫穷。不过用cutter得到的反编译结果还是很好的，可以看出gmp大数处理的流程以及后面的AES流程。

之后通过分析算法，分析出该程序为C语言gmp大数库首先的ECC，拿到G点和Q点坐标以及p。

```
p =
19203400062470045015087777889856139712619568731547266452395921118909888843
gx = 14110930956338990549043362706086783525738111853916827918836044437
gy =
18139951285206509280646260207666063344956234353009070757328924243751652725
8
qx =
31891691142228973409459264571840296336020869906635390764988965639428587741
qy =
21814747855707522410995148085528604068209609922250542598357174146875745246
```

知道key1和key2作为ECC的a和b参数，10进制形式输入，key3为`Q=nG`的n，16进制形式输入。所以接下来就求key1&key2&key3。

key1和key2很好求，因为G和Q为ECC上两点，坐标在程序中可以得到，所以 $$ G_{y}^{2} = G_{x}^{3} *+aG_{x}+b(mod| p) | Q_{y}^{2} = Q_{x}^{3} +aG_*{x}+b(mod\ p) \ $$ 求a和b，相当于二元二次方程，还是很好解的。多解也无所谓，反正最后都要mod p，而且最后的key是n，不会受这种a，b多解的影响。

```
F = GF(p)
# two points are enough to recover the curve parameters
M = Matrix(F, [[gx,1],[px,1]])
a,b = M.solve_right(vector([gy^2-gx^3,py^2-px^3]))
```

key3是那个n，这个坑是我在刷cryptohack的时候遇到的，正常来说用sage来解，但是因为选取的是一个有奇异点的ECC，所以在sage里下面这个方法会报错。

```
E = EllipticCurve(GF(p), [a, b])
```

因为这是一个非法的ECC，其判别式为0。 $$ *4a^{3}+27*b^{2}=0 $$ 所以某一点为奇异点，也就是没有切线。这也是sage报错的原因。

顺着sage的报错搜索一些资料就可以找到一些处理方法，预期的做法是可以通过将这个奇异点映射到(0, 0)来求解另一个域的DLP来得到结果。

可以通过sage来求，拿cryptohack里其他人的wp脚本改的数据

```
p = 419203400062470045015087777889856139712619568731547266452395921118909888843
a = 127674435493250896381564887687639140842587059709619094132 5627
b = 3301977122114617416555944695062132400912462210292325552529843659956465331070
g_x = 141109309563389905490433627060867835257381118539168279188360443 7
g_y = 1813995128520650928064626020766606334495623435300907075732892424375165272 58
q_x = 318916911422289734094592645718402963360208699066353907649889656394285877 41
q_y = 218147478557075224109951480855286040682096099222505425983571741468757452 46

gx, gy = g_x, g_y
px, py = q_x, q_y

F = GF(p)
# finding the roots, here we suppose the singular point is a node
# we make sure alpha is the double root
K.<x> = F[]
f = x^3 + a*x + b
roots = f.roots()
if roots[0][1] == 1:
    beta, alpha = roots[0][0], roots[1][0]
else:
    alpha, beta = roots[0][0], roots[1][0]

# transfer
slope = (alpha - beta).sqrt()
u = (gy + slope*(gx-alpha))/(gy - slope*(gx-alpha))
v = (py + slope*(px-alpha))/(py - slope*(px-alpha))

# should take a few seconds, don't worry (largest prime of p-1 is 42 bits only)
n = discrete_log(v, u)

from Crypto.Util.number import inverse, bytes_to_long, long_to_bytes
print(n)
print(hex(n)[2:])
print(long_to_bytes(n))
```

求得key1，key2，key3:

```
1276744354932508963815648876876391408425870597096190941325627
33019771221146174165559446950621324009124622102923255252984365995646533107
0
1636247906719776192850336424656337116834591860
495f346d5f4145535f6b33795f62595f747474
b'I_4m_AES_k3y_bY_ttt'
```

这个问题能解决有一个前提，那就是p-1的最大素因子不可以太大，不然sage算不出来，所以我这里的p是这样生成的：

```python
p = 0
while True:
    p = 2
    while True:
        nbits = randint(2, 40)
        p *= getPrime(nbits)
        if len(bin(p)) >= 250:
            break
    p = p+1
    if isPrime(p):
        break
```

之后最后一个check函数就是检验flag的，key3进行hex2str转化后作为AES的key，iv和密文硬编码在程序中了，注意padding填充，然后直接提取出来求解即可：

```python
from Crypto.Cipher import AES
from Crypto.Util.number import bytes_to_long, long_to_bytes

padding = lambda s, l: s + (l - len(s) % l) * long_to_bytes(0x26)
key = b"I_4m_AES_k3y_bY_ttt"
key = padding(key, 16)
enc_list = [0xd5, 0xf1, 0x9c, 0xf, 0x0, 0x74, 0xba, 0x62, 0x25, 0xb9,
0x5a, 0xd2, 0xc6, 0x15, 0x8e, 0x6c, 0xa8, 0xa1, 0xcb, 0xd7, 0xea, 0x25,
0x72, 0x6c, 0xcc, 0xe3, 0xe8, 0xf2, 0xf8, 0xcb, 0xb5, 0xe9, 0x21, 0x60,
0xa7, 0x9, 0x81, 0x29, 0x23, 0x0, 0x46, 0x6c, 0x96, 0xc5, 0x80, 0xd8,
0x26, 0x72, 0xa, 0x50, 0x4a, 0x85, 0x3a, 0xe8, 0x37, 0x7, 0xf0, 0x13,
0x33, 0x3f, 0xba, 0x5b, 0xde, 0x4b]
enc = bytes(enc_list)

iv_list = [0x66, 0x7b, 0x02, 0xa8, 0x5c, 0x61, 0xc7, 0x86, 0xde, 0xf4,
0x52, 0x1b, 0x06, 0x02, 0x65, 0xe8]
iv = bytes(iv_list)

aes = AES.new(key, AES.MODE_CBC, iv)
flag = aes.decrypt(enc)
print(flag)
```

最终得到flag：

> roarctf{If_ECC_is_singular_then_map_singular_point_to_zero}