FlagBot

设：Sender 的私钥为 s ，公钥为 S[i]=s·G[i]；Receiver[i]的私钥为 r[i] ，公钥为 R[i]=r[i]·G[i]；

从 bot.sage 中得知，Sender 与 7 位 Receiver 进行 ECDH 密钥交换的 ECC 曲线不同，但是私钥 s 一直都是同一个。（注意！每位 Receiver 的私钥 r[i]并不相同！）

从 output.txt 中可以提取出 7 个 ECC 曲线的参数(p,a,b,G) 和 交换的公钥(S[i],R[i])
得到 p,a,b,G,S[i],R[i]之后，使用 SageMath 求 G 点的阶，和阶的因子

```
F=FiniteField(p)
E=EllipticCurve(F,[a,b])
n=G.order();
factor(n)
```

得到了 7 条 ECC 曲线的 G 点阶的素因子

```
factor(G.order())=3 * 17 * 19 * 18717749 * 3645306673788202170491387831328793215764972129275226753932628337247
factor(G.order())=3^2 * 31 * 47 * 359 * 498931 * 346306357 * 69837885241 * 2001494609667086357874176864404002135285579589
factor(G.order())=2^2 * 3^2 * 7 * 23 * 326467 * 1190458786233403 * 1862220633704788040456328276089934182668956558853749
factor(G.order())=2^3 * 13 * 53 * 14293 * 3413941 * 960528293896228331404942482721 * 326728648737703880573448077356873
factor(G.order())=3 * 72138680149 * 29449254966031 * 128628689096246333 * 109575705046574513209422505790541601
factor(G.order())=31 * 167 * 1487 * 26993767 * 647535301012562933723 * 613440946101630803240982376082042364949889
factor(G.order())=2 * 32561 * 13221139 * 223919006521031181343 * 382209799078745023745203046466431038264965991
```

每个曲线的阶都有较大的素因子，看似无法使用 Pohlig-Hellman 算法攻击，但是将多条曲线得到的 素因子 和 对应的离散对数 组合起来，一起用中国剩余定理即可求出 Sender 的私钥 s

以第一条曲线(curves[0])为例：

```
p=66116745352204681519469100961724220561156687225236967561841501424810928307807
a=25080904022674360699928001742955972930647134245095368243176041782631007842721
b=51653146499008013562938728057440216556983928277479897024444219497340655463785
```

```
G=(63081569813401008552324671243982371480394903382778267836374
074361871201577834,
23474141625196059564605908362235632708979984522174825228263753550289099376123)
S=(24019555976552900590169231571461552054483009204136674938458
250215680500736104,
28011367537644480596653140995440810446354950450235617877156869800605294049216)
R=(21964916650662314987863240651434170404291932859105698307245
034338578258435153,
35342442175625429390144720170949410218699162882078535647858248442359306505200)


F=FiniteField(p)
E=EllipticCurve(F,[a,b])
G=E.point(G)
S=E.point(S)
R=E.point(R)
n=G.order()
factor(n)


primes = [3 , 17 , 19 , 18717749] #Abandon larger prime factors
dlogs = []


for fac in primes:
        t = int(G.order() / fac)
        dlog = discrete_log(t*S,t*G,operation="+")
        dlogs += [dlog]
        print("factor: "+str(fac)+", Discrete Log: "+str(dlog
```

可以得到 素因子 和 对应的离散对数

此处可以求出椭圆曲线离散对数，是因为 S 和 G 乘上 t 之后，就把问题从 n 上的 ECDLP，简化到了 fac 上的 ECDLP 问题，对于较小的 fac，椭圆曲线离散对数还是很好求的。

```
primes = [3, 17, 19, 18717749]
dlogs = [1, 3, 12, 2818256]
```

对 7 条曲线都进行如上操作，并将 primes 和 dlogs 分别相接，使用 中国剩余定理 求 Sender 的私钥 s

```
primes = [3, 17, 19, 18717749, 9, 31, 47, 359, 498931, 346306357, 4, 9, 7, 23, 326467, 8, 13, 53, 14293, 3413941, 31, 167, 1487, 26993767, 2, 32561, 13221139]
dlogs = [1, 3, 12, 2818256, 1, 24, 42, 220, 128456, 333616735, 1, 1, 4, 11, 254999, 1, 2, 36, 3714, 2228292, 24, 61, 270, 6115151, 1, 1011, 11121225]
```

s=crt(dlogs,primes)
得到 Sender 的私钥
s=47541809389785067168282163393632668193016293539381139194441901184004531371545
将 s 与 R[i]相乘，便可得到 Sender 与 Receiver[i]的共享密钥（曲线上的一点）
然后再根据 bot.sage 中的方式求出 key 和 iv

```
px = (s*R).xy()[0]
_hash = sha256(long_to_bytes(px)).digest()
key = _hash[:16]
iv = _hash[16:]
```

把 encrypted_msg[i]从 Base64 解码后，用 AES-128-CBC-PKCS7Padding 解密即可得到 flag

Hi, here is your flag: n1ctf{0ops_I_R3used_7h3_S4M3_Pr1vK3y}

Curve

题目的意思就是，让我们来选定 ECC 曲线的参数(p,a,b)以及基点(G1,G2)，然后他们来选定任意的 a0,a1∈ [1,E.order()-1]，然后给你三个点：
第一个点是 a0*G1，第二个点是 a1*G2，问你第三个点是不是(a0*a1)*G1？如果是，就选 0；如果不是，就选 1。
总共要你选择 30 轮，蒙对的概率 2^-30 并不高，而且出题者在最开头要我们 sha256 提供 PoW 杜绝了随机猜的方法。

题目还特别要求，ECC 曲线所在的 Fp 的 p 要大于等于 512 比特，而且不能是超奇异椭圆曲线。
这就是 ECDLP 问题，乍一看好像很困难的样子……但是实际上不难

其实，我们要做的只是弱化曲线的安全性，使得我们能够求出这条曲线上的 ECDLP 问题。
先按照正常步骤在 SageMath 里面随便生成一条曲线。

```
p = random_prime(2 ^ 512-1, False, 2 ^ 511)
a = 0
b = 7
E = EllipticCurve(GF(p), [a, b])
G = E.gens()[0]
n = G.order();
```

我得到的曲线如下：

p=11115951156331305339124436868299773824780306758938007873558
401973671324532347782694265245485730366046174505485581649583053750468162584601749993494363573457

a=0

b=7

G=(52747751748077461011833137460474861267511814920274461317623
0820835898296269072883819374014684285986999045724759369722022
5769547843765622524285015720020830 :
6598410216888973202959473822912665266581049713706265124239286
9575318555039699282821479179781504691159803875470556322256759
49759236643324101983965946851027 : 1)

n=185265852605521755652073947804996230413005112648966797892640
0328945220755391324191993042933221672547489382973861859650170
288794203668673846196962915158718

降低安全性，最重要的就是降低 G 点的阶，我们分解因式看一下

factor(n)=2 * 3 * 283 * 18363073381 *
59417219389797833596320653773529251097045660743880444206515686
4515152121877808241001285502283710103139649911238866619181029
74316276266471811

我们可以把 G 点的阶降低至 其中任意个素因子的乘积。这个阶其实非常好，里面包含几个小的素因数！

G 点的阶也不用降到太小，因为如果阶非常小，很容易导致 a0 * a1 = c (mod n)，使得 (a0*a1)*G=c*G，也就是说，我们无法辨别第三个点到底 应该选 0 还是 应该选 1。

所以说，我决定把 G 点的阶降低为 283，这个大小正合适！

G=G*(n/283)

这一步的意思就是，把除了 283 以外的因子都乘到 G 上，达到降阶的目的

比如已知 a*b*c*G=O ，O 是无穷远点，即 G 点的阶为(a*b*c)

如果令 G'=a*b*G，那么必定有 c*G'=O

此时 G'点的阶就是 c，安全性被弱化了

所以在我们重新设定了 G 点之后，G 点的阶已经非常低了

G=(26854392406201796844546812740063730212089713199588619689009
1650584127900520413830417476723331644350909909318635519973380
4710186695898796258730372308787459 :
1002301067856260249552855720846287913609902385228118987691568
6055103242896121203515719092498204233247052168049674151567826
186785256731061798143303516455834 : 1)

n=G.order()=283

才 283 个点，直接列个表，穷举出来就完事了

```
for num in range(1,283):
    print(num,num*G);
```

我不知道出题者让我们提供两个 G 点，可以提供不同的 G1 和 G2 的目的是什么？我觉得只要一个点就够了啊……

反正我给服务器的参数就是

p =
111159511563313053391244368682997738247803067589380078735584019736713245323477826942652454857303660461745054855816495830537504681625846017499934943635 73457

a = 0

b = 7

x1 = x2 =
268543924062017968445468127400637302120897131995886196890091650584127900520413830417476723331644350909909318635519973380471018669589879625873037230878 7459

y1 = y2 =
100230106785626024955285572084628791360990238522811898769156860551032428961212035157190924982042332470521680496741515678261867852567310617981433035164 55834

当收到服务器发送的三个点之后，直接在表里查找第一个点，第二个点，得到 a0 mod n 和 a1 mod n

我们自己计算 a0*a1*G（或者查表也行），判断与服务器发来的第三个点是否相等就行了！

完成 30 轮选择之后，就能得到这道题的 flag 了

Thank you! Here's your reward: n1ctf{your_curve_is_very_friendly_:3}