ECSC Estonia Prequalifier - dlog

Challenge script:

```
from Crypto.Util.number import isPrime
import random

p = int(input("give me a prime!: "))
assert isPrime(p)

for i in range(5):
    g = 4
    v = random.randint(2,2**312)
    print(f'{g}^x mod {p} = {pow(g,v,p)}')
    x = input("what was x?: ")
    if int(x) == v:
        print("correct!")
    else:
        print(f"wrong! v = {v}")
        exit()

print(f'well done!')

with open("flag.txt", "r") as f:
    flag = f.read()
    print(flag)
```

We are allowed to choose the prime P whose discrete log we have to calculate. This means that we can pick a prime p that is not safe and is smooth to perform Pohlig-Hellmann attack to calculate the discrete logarithm.

Script to generate a smooth prime p:

```
from Crypto.Util.number import isPrime, getPrime
from sage.all import factor

cur_number = 2
cur_factor = 3
while cur_number.bit_length() < 311:
    cur_number *= cur_factor
    cur_factor += 1

while not isPrime(cur_number + 1):
    cur_number *= cur_factor
```

```
    cur_factor += 1

print(cur_number)

with open('factors.txt', 'w') as f:
    f.write(str(list(factor(cur_number))))
```

Script to solve the challenge:

```
# Pohlig Helmann ftw
from sage.all import discrete_log, Mod
from pwn import *
from Crypto.Util.number import long_to_bytes

factors_n_1 = [(2, 70), (3, 34), (5, 16), (7, 11), (11, 6), (13, 5), (17,
4), (19, 3), (23, 3), (29, 2), (31, 2), (37, 1), (41, 1), (43, 1), (47,
1), (53, 1), (59, 1), (61, 1), (67, 1), (71, 1), (73, 1)]

p = 1
for i, j in factors_n_1:
    p *= i ** j

p += 1 #
44701154615126843408912571381250511100768007002829050158190800923704221040
67183317016903680000000000000001

HOST = 'dlog.hkn'
PORT = 9999
conn = remote(HOST, PORT)
conn.recvuntil(b'give me a prime!: ', timeout=3)
conn.sendline(str(p))

for i in range(5):
    line = conn.recvline(timeout=3)
    val = int(line.split(b'=')[1].strip())
    conn.recvuntil(b'what was x?: ', timeout=3)
    conn.sendline(str(discrete_log(Mod(val, p), Mod(4, p))))
    conn.recvline()

print(conn.recvall())
```