WriteUp of ottol r3pus

The main idea of this problem is OL (orthogonal lattice). From the attachment, it's not hard to see that the final goal is to recover the prime factors p. After obtaining 15 samples, compute the orthogonal lattice $\mathcal{L}_x^\perp$ for x and the orthogonal lattice $\mathcal{L}_y^\perp$ for y. Then, find the intersection of these two lattices, denoted as $\mathcal{L}_s$. Since $u+v$ is relatively small, compute the orthogonal lattice of $\mathcal{L}_s$, then use LLL to reduce the basis of the orthogonal lattice; $u+v$ will appear in the basis.

The remaining steps are straightforward: use x, y, and $u+v$ to recover the prime factor p. There are many ways to do this step. There are many ways to do this step. The method I used is to compute the orthogonal lattice of $x-(u+v)$ and then brute-force search for small combinations to find u. For the detailed steps, see the script below.

Below is the script I used for testing.

```
from sage.all import *
from Crypto.Util.number import isPrime
from pwn import process, remote
from re import findall
from tqdm import tqdm
import sys
from itertools import product

range_values = range(-2, 3)

triplets = list(product(range_values, repeat=3))
def Solve(x_v, y_v, n):
    Mxk = Matrix(ZZ, x_v).right_kernel_matrix()
    Myk = Matrix(ZZ, y_v).right_kernel_matrix()
    Lxk = IntegralLattice(identity_matrix(ZZ, n), Mxk)
    Lyk = IntegralLattice(identity_matrix(ZZ, n), Myk)
    Lxyk = Lxk.intersection(Lyk)
    Mxyk = Lxyk.basis_matrix()
    MxykL = Mxyk.LLL()
    Mek = MxykL[:11].right_kernel_matrix()
    m_v = Mek[0]

    Myk = Matrix(ZZ, y_v - m_v).right_kernel_matrix()
    Lyk = IntegralLattice(identity_matrix(ZZ, n), Myk)
    Lxyk = Lxk.intersection(Lyk)
    Mxyk = Lxyk.basis_matrix()
    MxykL = Mxyk.LLL()



    Muk =  MxykL[:12].right_kernel_matrix()
    for triple in triplets:
        u_v = vector(triple) * Muk
        if gcd(list(x_v - u_v)) > 2 ** 100 and
```

```
isPrime(gcd(list(x_v - u_v))):
            return gcd(list(x_v - u_v))

# conn = process(["python", **********])
xs = []
ys = []
for _ in tqdm(range(15)):
    resp = conn.recvuntil(b"0\n").decode()
    conn.sendline("1")
    resp = conn.recvuntil(b"number: ").decode()
    xs.append(int(findall("x = (.*)", resp)[0]))
    ys.append(int(findall("y = (.*)", resp)[0]))
    conn.sendline("1")

x_v = vector(xs)
y_v = vector(ys)



resp = conn.recvuntil(b"0\n").decode()
res = str(Solve(x_v, y_v, 15))
conn.close()
```