Writeup

1. 查看源码，注意到p和q存在一个有趣的关系：

```
hex(p)=hex(q)[::-1]
```

显然，这个关系是一个突破口，我们可以利用它来进行N的分解。

2. 这里，我们爆破的思路类似于 RSA parity oracle。因为p和q是bit翻转的关系，我们已知p最低的k位bit，那么我们就已知了q最高的k位bit。假设，我们已知k位的p和q，记为qh、ph，利用 ph*qh*2^(1024-2k) <= n < (ph+1)*(qh+1)*2^(1024-2k)不等式，我们逐位向低地址爆破，不断收缩不等式的范围，最终便可在求得n值。

3. 当然，直接爆破512bit的话，这依然是一个不小的工作量，这里可以分阶段爆破。

第一步，我们先爆破最高的12位：

```
res = []
for pl in range(1,4096,2):
    ql = invert(pl,4096)*(n%4096) % 4096

    if (ql*pl)%4096 == n%4096:
        qh = reverse(ql,12)
        ph = reverse(pl,12)
        min = (ph*qh) << 1000
        max = ((ph+1)*(qh+1)) << 1000

        if min <= n < max:
            res.append(pl)
```

这样，我们就得到了满足该条件下的所有解。

接着，再去爆破剩余的bit位，就可以大大缩短爆破的时间：

```
for c in range(13,257):
    t_res = []
    mod = 2**c
    for x in res:
        for y in range(2):
            pl = x + (y*2)**(c-1)
            ql = (invert(pl, mod)*(n%mod)) % mod
            if (ql*pl)%mod == n%mod:
                qh = reverse(ql,c)
                ph = reverse(pl,c)
                min = ph*qh << (1024 - c*2)
                max = (ph+1)*(qh+1) << (1024 - c*2)
```

```
                if min <= n < max:
                    t_res.append(pl)
    res = t_res
```

4. 根据前面提到的思路，编写exp.py即可得到flag。