求hint的rsa由于$|p-q|$比较小，可以费马分解跑出

```
def fetmat(n):
    a = gmpy2.iroot(n,2)[0] + 1
    while True:
        tmp = pow(a,2) - n
        root = gmpy2.iroot(tmp,2)
        if root[1]:
            b = root[0]
            break
        else:
            a += 1
    p = a + b
    assert n % p == 0
    return p,n // p
```

得到p和q之后可以解出hint，也就可以解出后面lcg使用的p

之后的lcg给了三组连续值，可以用三组连续值求出a和b

$a\equiv(s3-s2) \times (s2 - 1)^{-1} \bmod p$

$b \equiv s2 - a \times s1 \bmod p$

得到a和b之后再往前推一个就得到flag了。

```
from Crypto.Util.number import *
import gmpy2

def fetmat(n):
    a = gmpy2.iroot(n,2)[0] + 1
    while True:
        tmp = pow(a,2) - n
        root = gmpy2.iroot(tmp,2)
        if root[1]:
            b = root[0]
            break
        else:
            a += 1
    p = a + b
    assert n % p == 0
    return p,n // p

e = 0x10001
n1 =
15952018276550856311557026433440740283497664025474231224687394501040735445814071
34814284884645777922986893835542193369430052619017614526611562744730461291973505
70197214073758586734534700614489167819948591187080614587499218447273903704692119
61348221533742210513796307632406749693799398592526415874541790634286841356334828
56909469983991532142100019372070432980426251535613973129785996619559603087870302
75147648397118471462113830565235265359316549188942626788167754184552938338446482
46769442743756611106801468115567791566843278107839468782404143014084880934870119
79860990138951251029153682179431455059007372069670376 4713
```

```
c1 =
10395269685989014892273456005433910393081781200834054976329530058041309338740089
37056252355502788678433292457472980595443393918188270891002152094361881893372112
99284942083247960207542234898898749182831968068913408788108753252231602606021394
96251803627559661276715065387806153029069117358735240502793160480938423552399686
43314310576242044828260808777609952046258260266523171535977189930473840115765618
78314327994880907391127349276110563337420339377918808210968286608790107188073575
18509216909529689205256959919470215087228827106585408889091740420744686391068632
92508514769869085112768821404201067218910074719256546180l
xorp =
13316591235813967314531080218762385608142365136919308297518939087169745892778249
08128193850635843914584626755935212410536289807601625272919287091092292346O
output =
[71679004365618244196809092753326493654201735386925517495049573803856957044569952
67039213180880965574480657325490656588988177909881954299542605168795293493,
81570485195536556447832105473118960678837206520919503063377529584967515123171715
61414409131544996344715089151480325717210751622011971475760356517772149320,
85977338863515168050133030160142658490441771645849528815616502032011156323790496
12415304316876669100762719059486218313534835875811780214809538696496453473]

p1,q1 = fetmat(n1)
phi1 = (int(p1) - 1) * (int(q1) - 1)
d1 = gmpy2.invert(e,phi1)
hint = int(pow(c1,d1,n1))
p = xorp ^ hint
a = (output[2] - output[1]) * gmpy2.invert(output[1] - output[0],p) % p
b = (output[2] - a * output[1]) % p
flag = (output[0] - b) * gmpy2.invert(a,p) % p
print(long_to_bytes(flag))
```