

Interesting\_math

Flag 的开头是已知的，我们可以利用这个得到四个式子的线性表达式，然后可以系数提取出来转换成矩阵求解，由于是多解，需要遍历其他所有可能去恢复

```
from sage.all import *
from Crypto.Cipher import AES
from Crypto.Util.number import getPrime, long_to_bytes
from lll_cvp import polynomials_to_matrix
from Crypto.Util.Padding import pad
```

```
p = 38873
hashes = [30328,18774,1692,31709]
ciphertext =
```

```
F = GF(p)
```

```
r = list(polygens(F, "r", 5))
flag = b"flag{"
```

```
eqs = []
for i in range(4):
    h = flag[i]
    for j in range(5):
        h = h + (j + 1) * r[j]
        r[j] = h
    print(f"hash[{i}] = {h}")
    eqs.append(h - hashes[i])
M, monos = polynomials_to_matrix(eqs)
assert monos[-1] == 1
A = M[:, :-1]
b = vector(-M[:, -1])
```

```
rk = A.right_kernel_matrix()[0]
s0 = A.solve_right(b)
for t in range(p):
    s = s0 + t * rk
    assert all([e*s == 0 for e in eqs])
    key = 0
    for rr in [f*s for f in r]:
        key += int(rr)
    key *= 2**16
    key = pad(long_to_bytes(key), 16)
    aes = AES.new(key, AES.MODE_ECB)
```

```
flag = aes.decrypt(ciphertext)
if b"flag{" in flag:
    print(flag)
    break
```