Hgame week4 ---Conner

CRYPTO

1. ECC
2. PRNG

crypto

1.ECC

ECC，用椭圆曲线的一种加密方法。由此学习了它的加密原理等，开始做题。根据源码，可以知道，flag被m点的两个坐标值给加密了，m点是解题的关键。

最开始，还一直在找G点怎么求，www，后面发现根本不重要，m是直接用c1-k*c2求出来的。

之后就很简单了，用虚拟机下的sagemath编写简单的脚本就可以得出了flag两部分的值，放到python n2s一下，得到flag了。

2.PRNG

呜呜呜，常规思路先上网查查PRNG是啥意思，嗯嗯伪随机数，根据题目给的output和源码，应该是利用随机的几个数加密了flag，问题就是如何找到用来加密的那些数。

第一次尝试，还没上网搜，就想着能不能直接找出随机数的种子，1到1<<32范围里面运气好能不能挑出来，要是结果和那624个随机数一样，就有了嘛。就利用了源代码，写了个循环，跑了半天才跑到几千万，Python确实是慢了点，主要是这个随机数生成器算法也较于复杂，放弃。

后面觉得624这个数字过于特殊，就加这上网查阅，发现是MT19937伪随机数生成算法，于我们的题目一模一样，解决方法主要就是将624个随机数逆回去，得到初始state，就能用他们得到我们想要的后面的21个随机数了，以下是网上直接用的代码，extract_number里逆回去异或的思路有学习到，y和y<<18异或异或，前十八位还是一样的，因此可以通过结果异或出原来的y,算式和原来都一样的感觉。呜呜呜，自己现在肯定写不出来，希望以后能。

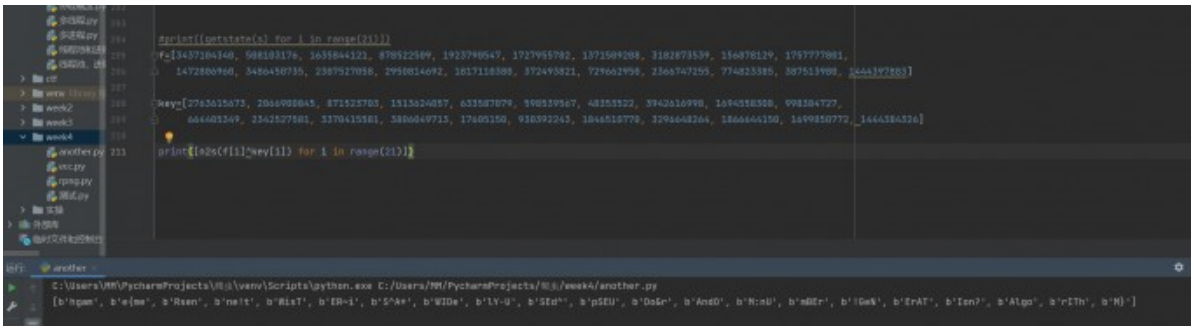
```
def inverse_right(res, shift, mask=0xffffffff, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift & mask
    return tmp


def inverse_left(res, shift, mask=0xffffffff, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift & mask
    return tmp


def extract_number(y):
    y = y ^ y >> 11
    y = y ^ y << 7 & 2636928640
```

```
    y = y ^ y << 15 & 4022730752
    y = y ^ y >> 18
    return y & 0xffffffff


def recover(y):
    y = inverse_right(y, 18)
    y = inverse_left(y, 15, 0xefc60000)
    y = inverse_left(y, 7, 0x9d2c5680)
    y = inverse_right(y, 11)
    return y & 0xffffffff


def _int32(x):
    return int(0xFFFFFFFF & x)


mti = 0


def getstate(s):
    global mti
    if mti == 0:
        twist(s)
    y = s[mti]
    y = y ^ y >> 11
    y = y ^ y << 7 & 0x9d2c5680
    y = y ^ y << 15 & 0xefc60000
    y = y ^ y >> 18
    mti = (mti + 1) % 624
    return _int32(y)


def twist(s):
    for i in range(0, 624):
        y = _int32((s[i] & 0x80000000) + (s[(i + 1) % 624] & 0x7fffffff))
        s[i] = (y >> 1) ^ s[(i + 397) % 624]
        if y % 2 != 0:
            s[i] = s[i] ^ 0x9908b0df
```

```
def inverse_right(res, shift, mask=0xffffffff, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift & mask
    return tmp


def inverse_left(res, shift, mask=0xffffffff, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift & mask
    return tmp


def extract_number(y):
    y = y ^ y >> 11
    y = y ^ y << 7 & 2636928640
```

```
    y = y ^ y << 15 & 4022730752
    y = y ^ y >> 18
    return y & 0xffffffff


def recover(y):
    y = inverse_right(y, 18)
    y = inverse_left(y, 15, 0xefc60000)
    y = inverse_left(y, 7, 0x9d2c5680)
    y = inverse_right(y, 11)
    return y & 0xffffffff


def _int32(x):
    return int(0xFFFFFFFF & x)


mti = 0


def getstate(s):
    global mti
    if mti == 0:
        twist(s)
    y = s[mti]
    y = y ^ y >> 11
    y = y ^ y << 7 & 0x9d2c5680
    y = y ^ y << 15 & 0xefc60000
    y = y ^ y >> 18
    mti = (mti + 1) % 624
    return _int32(y)


def twist(s):
    for i in range(0, 624):
        y = _int32((s[i] & 0x80000000) + (s[(i + 1) % 624] & 0x7fffffff))
        s[i] = (y >> 1) ^ s[(i + 397) % 624]
        if y % 2 != 0:
            s[i] = s[i] ^ 0x9908b0df
```

用得出来的结果，把加密过的flag解密回来就好啦！



4周的hgame终于结束了，能力有限，要是能做出更多题就好了，www,过年以后还是惰怠了，基础什么都没打牢，看到一题学一题，但也学了挺多的，寒假没有很摸，就挺好的。