Week4-ChenMoFeiJin

Crypto

~~(╥﹏╥)这俩我做完看到 Flag 才知道是是伪随机数生成器，想到头秃才推出解法，结果发现网上有现成的~~

夺宝大冒险1

描述

nc 182.92.108.71 30641

解题思路

观察题目提供的 py 文件，得知本题可简化为：

已知数列 x_i 满足 $x_i \equiv (a \times x_{i-1} + b) \equiv a^i \times x_0 + \frac{a^i-1}{a-1} \times b \pmod{c}$ ~~(习惯性的就求子通项公式呢)~~

1. 已知 a c 和 x_0 x_1 x_2 求 b
2. 已知 c 和 x_0 x_1 x_2 x_3 求 a
3. 已知 x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 求 c

   注：代码中 $x_0 = 123$

下面依次来解决这些问题。

1. 稍加变换 b 的值和以很容易表示出来

$$\because x_i \equiv (a \times x_{i-1} + b) \equiv a^i \times x_0 + \frac{a^i-1}{a-1} \times b \pmod{c}$$
$$\therefore b \equiv x_{i+1} - a \times x_i \equiv (x_j - a^{j-i} \times x_i)/(\frac{a^{j-i}-1}{a-1}) \pmod{c}$$

   代码如下（取特殊情况）

```python
def calc_b(x, a, c):  # 通过 x, a, c 求 b
    return (x[1] - x[0] * a) % c
```

2. 想求 a 就得先消去 b

$\because x_i \equiv (a \times x_{i-1} + b) \equiv a^i \times x_0 + \frac{a^i-1}{a-1} \times b \pmod{c}$

$\therefore x_{i+1} - x_i \equiv (a \times x_i + b) - (a \times x_{i-1} + b) \equiv a \times (x_i - x_{i-1}) \equiv a^i \times (x_1 - x_0) \pmod{c}$

令 $y_i = x_{i+1} - x_i$ 得，$y_i \equiv a \times y_{i-1} \equiv a^i \times y_0 \pmod{c}$

$\therefore a \equiv y_i / y_{i-1} \equiv \sqrt[j-i]{y_j / y_i} \pmod{c}$

   代码如下（取特殊情况）

```python
def calc_a(x, c):  # 通过 x, c 求 a
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    return y[1] * (lambda t: t[0] // t[2])(libnum.xgcd(y[0], c)) % c
    # 使用 invmod 也行，不过遇到 y0 没有逆的情况会报错
    # return y[1] * libnum.invmod(y[0], c) % c
```

3. 想求 c 就得先消去无法提前求解的 a b，由于 2 中已消去过 b 下面消去 a

$$
\begin{aligned}
&\because y_i \equiv a \times y_{i-1} \equiv a^i \times y_0 \pmod{c} \\
&\therefore y_{i+2} \times y_i - y_{i+1}^2 \equiv a^2 \times y_i \times y_i - (a \times y_i)^2 \equiv 0 \pmod{c} \\
&\text{令 } z_i = y_{i+2} \times y_i - y_{i+1}^2 \text{得，} z_i \equiv 0 \pmod{c} \\
&\therefore c | z_i ^{①} \ \therefore c | (z_0, \ldots, z_k) ^{②} \\
&\text{不妨取 } c = (z_0, \ldots, z_k)
\end{aligned}
$$

代码如下（取特殊情况）

```python
def calc_c(x):  # 通过 x 求 a
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    z = [y[i + 2] * y[i] - y[i + 1] * y[i + 1] for i in range(len(y) - 2)]
    return abs(libnum.reduce(libnum.gcd, z))
```

最终代码如下

```python
def calc_b(x, a, c):  # 通过 x, a, c 求 b
    return (x[1] - x[0] * a) % c
def calc_a(x, c):  # 通过 x, c 求 a
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    return y[1] * (lambda t: t[0] // t[2])(libnum.xgcd(y[0], c)) % c
    # 使用 invmod 也行，不过遇到 y0 没有逆的情况会报错
    # return y[1] * libnum.invmod(y[0], c) % c
def calc_c(x):  # 通过 x 求 a
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    z = [y[i + 2] * y[i] - y[i + 1] * y[i + 1] for i in range(len(y) - 2)]
    return abs(libnum.reduce(libnum.gcd, z))

i = 1
while True:  # 由于上述方法求出的 a, b, c 仅为满足条件的特解，不并不一定是服务器所需的解，故多次尝试
    print(f"try {i}"); i += 1
    sh = pwn.remote("182.92.108.71", 30641)  # 借助 pwn 的工具来与服务器交互

    def recv(k):  # 获取一行数据并判断解题进度
        s = sh.recvline()
        if s != b'fail\n': return s
        else: raise Exception(k)

    try:
        # 解决问题 1
        a, c = map(int, recv(0)[1:-2].split(b','))
        x = [123] + [int(recv(0)) for _ in range(2)]
        b = calc_b(x, a, c)
        sh.sendline(str(b))
        # 解决问题 2
        c = int(recv(1))
        x = [123] + [int(recv(1)) for _ in range(3)]
        a = calc_a(x, c)
        b = calc_b(x, a, c)
        sh.sendline(str(a))
        sh.sendline(str(b))
        # 解决问题 3
        x = [123] + [int(recv(2)) for _ in range(7)]
        c = calc_c(x)
        sh.sendline(str(c))
```

```
        # 输出 Flag
        print(recv(3))
        print(recv(3))
        break
    except Exception as err:
        print(f"Wrong in {err}")
```

附本地测试代码

```
class Tester:  # Cxxiff 等效替代版
    x, a, b, c = [0 for _ in range(4)]
    def __init__(self):
        self.x = 123
        self.a, self.b, self.c = [int.from_bytes(os.urandom(8), 'big') for _
in range(3)]
    def next(self):
        self.x = (self.x * self.a + self.b) % self.c
        return self.x

def calc_b(x, a, c):
    return (x[1] - x[0] * a) % c
def calc_a(x, c):
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    return y[1] * libnum.xgcd(y[0], c)[0] % c
def calc_c(x):
    y = [x[i + 1] - x[i] for i in range(len(x) - 1)]
    z = [y[i + 2] * y[i] - y[i + 1] * y[i + 1] for i in range(len(y) - 2)]
    return abs(libnum.reduce(libnum.gcd, z))

i = 1
while True:
    print(f"try {i}");i += 1
    t = Tester()
    x = [123] + [t.next() for _ in range(7)]
    b = calc_b(x, t.a, t.c)
    if b != t.b: print(f"Wrong in 1"); continue
    a = calc_a(x, t.c)
    b = calc_b(x, a, t.c)
    if a != t.a or b != t.b: print(f"Wrong in 2"); continue
    c = calc_c(x)
    if c != t.c: print(f"Wrong in 3"); continue
    print("Successful")
    break
```

最终得到 Flag，`hgame{Cracking^prng_Linear)Congruential&Generators}`

夺宝大冒险2

描述

nc 182.92.108.71 30607

解题思路

观察题目提供的 py 文件，发现其中最关键的函数为 `next`，其代码如下

```
def next(self):
    nextdata = (self.init << 1) & self.lengthmask
    i = self.init & self.mask & self.lengthmask
    output = 0
    while i != 0:
        output ^= (i & 1)
        i = i >> 1
    nextdata ^= output
    self.init = nextdata
    return output
```

易得其逻辑为将 `init` 左移 1 位并在结尾补上一个 `output`，并返回 `output`，且 `init` 的限长为 `41` 位，有效位为 `40` 位。

而 `random` 函数就是将 `next` 函数执行 `nbit` 次，并把这 `nbit` 次 `next` 的返回值连起来变为一个二进制数，并返回相应的值

故只需执行 `[length / nbit]` 次，我们即可将 `init` 完全替换一轮，并通过 `random` 的返回值确定 `init` 的值

下面是代码实现

```
class Tester: # LXFIQNN 等效替代版
    def __init__(self, init):
        self.init, self.mask, self.lengthmask = init, 766820519669, ~-(1 << 40)
    def next(self):
        output = bin(self.init & self.mask & self.lengthmask).count('1') & 1
        self.init = (self.init << 1) & self.lengthmask ^ output
        return output
    def random(self, nbit):
        return functools.reduce(lambda x, y: x << 1 | y, [self.next() for _ in range(nbit)])

sh = pwn.remote("182.92.108.71", 30607)

def send(n): sh.sendline(str(n))
def recv(n): return sh.recvlines(n)

for _ in range(10): send(-1)  # 发送 10 个 -1 来更新 init 的值
init = int("".join(map(lambda l: hex(int(l[28:]))[2:], recv(20)[1::2])), 16)  # 获取前十个正确的数构造 init
prng = Tester(init)  # 使用 init 构造随机数发生器
for _ in range(90): send(str(prng.random(4)))  # 一次性发送 90 个正确的答案
print(recv(181)[-1])  # 输出 Flag
```

附本地测试代码

```
class Tester:  # LXFIQNN 等效替代版
    def __init__(self, init):
        self.init, self.mask, self.lengthmask = init, 766820519669, ~-(1 << 40)
```

```
    def next(self):
        output = bin(self.init & self.mask & self.lengthmask).count('1') & 1
        self.init = (self.init << 1) & self.lengthmask ^ output
        return output
    def random(self, nbit):
        return functools.reduce(lambda x, y: x << 1 | y, [self.next() for _ in range(nbit)])

t = Tester(int.from_bytes(os.urandom(5), 'big'))

init = int("".join([hex(t.random(4))[2:] for _ in range(10)]), 16)
prng = Tester(init)
print(functools.reduce(lambda x, y: x and y, [t.random(4) == prng.random(4) for _ in range(90)]))
```

最终得到 Flag，`hgame{lfsr_121a111y^use-in&crypto}`