hpcurve

发表于 2021-05-25 更新于 2021-05-26 分类于 Challenge， 2021， 第四届红帽杯网络安全大赛， Crypto
Challenge | 2021 | 第四届红帽杯网络安全大赛 | Crypto | hpcurve

点击此处获得更好的阅读体验

WriteUp来源

来自Bellin的博客

题目描述

> *你的数学学的怎么样？*

题目考点

解题思路

然后这个题也是一次CTF的原题： hxpCTF 2020。

基本就改了加密的时候异或的密钥长度比明文长度长，所以取余即可。

server:

```
#!/usr/bin/env sage
import struct
from random import SystemRandom

p = 10000000000000001119
R.<x> = GF(p)[]
y=x
f = y + y^7
C = HyperellipticCurve(f, 0)
J = C.jacobian()

es = [SystemRandom().randrange(p**3) for _ in range(3)]
Ds = [J(C(x, min(f(x).sqrt(0,1)))) for x in (11,22,33)]
q = []

def clk():
  global Ds,es
  Ds = [e*D for e,D in zip(es, Ds)]
  return Ds

def generate():

    u,v = sum(clk())
    rs = [u[i] for i in range(3)] + [v[i] for i in range(3)]
    assert 0 not in rs and 1 not in rs
    q = struct.pack('<'+'Q'*len(rs), *rs)
    return q


flag = "flag{xxxxxxx}"
text = 'a'*20+flag
t = ''
keys = generate()
leng = len(keys)
i = 0
for x in text:
    t += chr(ord(keys[i%leng])^^ord(x))
    i+=1
print t.encode('hex')
#for x,y in zip(RNG(),flag):
```

solve

```
from itertools import product
import struct
p = 10000000000000001119

K = GF(p)
R.<x> = K[]; y=x
f = y + prod(map(eval, 'yyyyyyy'))
C = HyperellipticCurve(f, 0)
J = C.jacobian()

def get_u_from_out(output, known_input):
    res = []
    for i in range(24):
        res.append(output[i]^^known_input[i])
    res = bytes(res)
    u0, u1, u2 = struct.unpack("<QQQ", res)
    u = x^3+x^2*u2+x*u1+u0
    return u

def get_v_from_u(u):
    Kbar = GF(p^6)
    Rbar.<t> = Kbar["t"]
    u2 = u.change_ring(Rbar)
    roots = [x[0] for x in u2.roots()]
    ys = []
    for root in roots:
        ys.append(f(root).sqrt(0,1))
    res = []
    for perm in product(range(2), repeat=3):
        poly = Rbar.lagrange_polynomial([(roots[i], ys[i][perm[i]]) for i in range(3)])
        if poly[0] in K:
            res.append(R(poly))
    return res

def try_decode(output, u, v):
    rs = [u[0], u[1], u[2], v[0], v[1], v[2]]
    otp = struct.pack("<QQQQQQ", *rs)
    l=len(otp)
    plain = []
    for i in range(len(output)):
        plain.append(output[i]^^otp[i%l])
    return bytes(plain)


output = bytes.fromhex("66def695b20eeae3141ea80240e9bc7138c8fc5aef20532282944ebbbad76a6e17446e92de5512091fe81255eb34a0e22a86a090e25dbbe3141aff0542f5")
known_input = b"a"*20+b"flag"
u = get_u_from_out(output, known_input)
vs = get_v_from_u(u)
for v in vs:
    print(try_decode(output,u,v))
```

Flag

```
flag{1b82f60a-43ab-4f18-8ccc-97d120aae6fc}
```


 # Challenge # 2021 # Crypto # 第四届红帽杯网络安全大赛

ezRev

PicPic