cryptohack Elliptic Curves 部分wp by crumbling

目录

ELLIPTIC CURVES

STARTER

Point Negation

ECC上的点取负

```
from sage.all import*
p=9739
a=497
b=1768
F=GF(p)
E=EllipticCurve(F,[a,b])
P=E(8045,6936)
Q=-P
print(Q)
```

Point Addition

ECC上的点加

```python
from sage.all import*
p=9739
a=497
b=1768
F=GF(p)
E=EllipticCurve(F,[a,b])
P = E(493, 5564)
Q = E(1539, 4742)
R = E(4403,5202)
print(P+P+Q+R)
```

Scalar Multiplication

ECC上数*点

```python
from sage.all import*
p=9739
a=497
b=1768
F=GF(p)
E=EllipticCurve(F,[a,b])
P = E(2339, 2213)
Q=7863*P
print("crypto{"+f"{Q[0]},{Q[1]}"+"}")
```

Curves and Logs

一种利用ECDLP的DH

```python
from sage.all import*
import hashlib
p=9739
a=497
b=1768
F=GF(p)
E=EllipticCurve(F,[a,b])
G=E(1804,5368)
QA=E(815,3190)
nB=1829
S=nB*QA
sha1=hashlib.sha1()
sha1.update(str(S[0]).encode('ascii'))
key = sha1.hexdigest()
print(key)
```

Efficient Exchange

本意是利用p mod 4 =3 求解二次同余方程那个东西，但是sage有集成已知椭圆曲线和点的一个坐标求全部坐标的函数。

```
from sage.all import*
p=9739
a=497
b=1768
F=GF(p)
E=EllipticCurve(F,[a,b])
G=E(1804,5368)
q_x = 4726
Q=E.lift_x(F(q_x))
nB = 6534
S=nB*Q
print(S[0])
```

PARAMETER CHOICE

Smooth Criminal

椭圆曲线上的离散对数的求解问题。

这里可以直接用函数求解，但是也看到用对生成元的阶进行质因数分解后crt的解法。

另外题目给的源代码是直接用python实现的椭圆曲线基本操作，挺有趣。

```
from sage.all import *
import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

def decrypt(shared_secret:int,iv,encrypted_flag):
    # Derive AES key from shared secret
    iv=bytes.fromhex(iv)
    encrypted_flag=bytes.fromhex(encrypted_flag)
    sha1=hashlib.sha1()
    sha1.update(str(shared_secret).encode('ascii'))
    key=sha1.digest()[:16]
    # decrypt flag
    cipher = AES.new(key, AES.MODE_CBC, iv)
    flag = cipher.decrypt(pad(encrypted_flag, 16))
    return flag

# Define the curve
p = 310717010502520989590157367261876774703
a = 2
b = 3
F=GF(p)
E=EllipticCurve(F,[a,b])
# Generator
g_x = 179210853392303317793440285562762725654
g_y = 105268671499942631758568591033409611165
G = E(g_x, g_y)

#public=n*G
public=E(280810182131414898730378982766101210916,291506490768054478159835604632710368904)
```

```python
# Bob's public key
b_x = 272640099140026426377756188075937988094
b_y = 51062462309521034358726608268084433317
B = E(b_x, b_y)

n=discrete_log(public,G,operation='+')
shared_secret=n*B
iv='07e2628b590095a5e332d397b8a59aa7'
encrypted_flag='8220b7c47b36777a737f5ef9caa2814cf20c1c1ef496ec21a9b4833da24a008d
0870d3ac3a6ad80065c138a2ed6136af'
print(decrypt(shared_secret[0],iv,encrypted_flag))
```

Exceptional Curves

阶成了质数，但这个质数刚好是p。

在L佬（[ECC | Lazzaro (lazzzaro.github.io)]）那里看到了Smart's attack

```python
from sage.all import*
import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

p =
0xa15c4fb663a578d8b2496d3151a946119ee42695e18e13e90600192b1d0abdbb6f787f90c8d102
ff88e284dd4526f5f6b6c980bf88f1d0490714b67e8a2a2b77
a =
0x5e009506fcc7eff573bc960d88638fe25e76a9b6c7caeea072a27dcd1fa46abb15b7b6210cf90c
aba982893ee2779669bac06e267013486b22ff3e24abae2d42
b =
0x2ce7d1ca4493b0977f088f6d30d9241f8048fdea112cc385b793bce953998caae680864a7d3aa4
37ea3ffd1441ca3fb352b0b710bb3f053e980e503be9a7fece
E = EllipticCurve(GF(p), [a, b])
G =
E(303471280937553790810298875011338244400875853944897275058152581090063424339217
2703684905257490982543775233630011707375189041302436945106395617312498769005,498
66450985826164156900740822378176244243333390749693645275481070428761754808941325
76399611027847402879885574130125050842710052291870268101817275410204850)
b_x =
0x7f0489e4efe6905f039476db54f9b6eac654c780342169155344abc5ac90167adc6b8dabacec64
3cbe420abffe9760cbc3e8a2b508d24779461c19b20e242a38
b_y =
0xdd04134e747354e5b9618d8cb3f60e03a74a709d4956641b234daa8a65d43df34e18d00a59c070
801178d198e8905ef670118c15b0906d3a00a662d3a2736bf
B = E(b_x, b_y)
A=E(4748198372895404866752111766626421927481971519483471383813044005699388317650
39531519392226704604937454742608233124831870493636003725200307683939875286865,2
42187330900227984102179136988448330805149721579801750980530204110246831063682206
0707350789776065212606890489706597369526562336256272258544226688832663757)

def SmartAttack(P,Q,p):
    E = P.curve()
    Eqp = EllipticCurve(Qp(p, 2), [ ZZ(t) + randint(0,p)*p for t in
E.a_invariants() ])
```

```
    P_Qps = Eqp.lift_x(ZZ(P.xy()[0]), all=True)
    for P_Qp in P_Qps:
        if GF(p)(P_Qp.xy()[1]) == P.xy()[1]:
            break

    Q_Qps = Eqp.lift_x(ZZ(Q.xy()[0]), all=True)
    for Q_Qp in Q_Qps:
        if GF(p)(Q_Qp.xy()[1]) == Q.xy()[1]:
            break

    p_times_P = p*P_Qp
    p_times_Q = p*Q_Qp

    x_P,y_P = p_times_P.xy()
    x_Q,y_Q = p_times_Q.xy()

    phi_P = -(x_P/y_P)
    phi_Q = -(x_Q/y_Q)
    k = phi_Q/phi_P
    return ZZ(k)

r = SmartAttack(G,A , p)
shared_secret=(r*B)[0]
iv= '719700b2470525781cc844db1febd994'
encrypted_flag:
'335470f413c225b705db2e930b9d460d3947b3836059fb890b044e46cbb343f0'

def decrypt(shared_secret:int,iv,encrypted_flag):
    # Derive AES key from shared secret
    iv=bytes.fromhex(iv)
    encrypted_flag=bytes.fromhex(encrypted_flag)
    sha1=hashlib.sha1()
    sha1.update(str(shared_secret).encode('ascii'))
    key=sha1.digest()[:16]
    # decrypt flag
    cipher = AES.new(key, AES.MODE_CBC, iv)
    flag = cipher.decrypt(pad(encrypted_flag, 16))
    return flag
print(decrypt(shared_secret,iv,encrypted_flag))
```