ElGamal与离散对数

📌 学号：202200460104

姓名：密语

班级：网安一班

练习1.16

$$
\begin{gathered}
y^2 = x^3 + 6(mod13) \\
y^2 = x^3 + 2x + 8(mod13) \\
y^2 = x^3 + 2x(mod13)
\end{gathered}
$$

1. 判断以上是否是 $\mathbb{Z}_{13}$ 上的椭圆曲线方程?

   使用判别式 $\Delta = -16 \cdot (4a^3 + 27b^2)$，如果 $\Delta \neq 0$，则方程是一个椭圆曲线。

   编写一个python脚本来判断：

```python
def delta(a,b,p):
    if(-16)*(4*pow(a,3)+27*pow(b,2))%p!=0 :
        return True
    else:
        return False


print(delta(0,6,13))
print(delta(2,8,13))
print(delta(2,0,13))
```

   运行结果：

```
PS D:\密码学引论> python -u "d:\密码学引论\delta.py"
True
True
True
```

   得到以上方程均是 $\mathbb{Z}_{13}$ 上的椭圆曲线方程。

2. 求出上述方程在 $\mathbb{Z}_{13}$ 上所有的解。

   使用sagemath的 `points()` 函数来求椭圆曲线上的所有有理点：

```
#sagemath 9.5
F=GF(13)
curve1=EllipticCurve(F,[0,6])
points1=curve1.points()
print("曲线1的所有解为：")
for point in points1:
     print(point)
curve2=EllipticCurve(F,[2,8])
points2=curve2.points()
print("曲线2的所有解为：")
for point in points2:
     print(point)
curve3=EllipticCurve(F,[2,0])
points3=curve3.points()
print("曲线3的所有解为：")
for point in points3:
    print(point)
```

```
➜ 密码学引论 sage EllipticCurve.sage
曲线1的所有解为：
(0 : 1 : 0)
(2 : 1 : 1)
(2 : 12 : 1)
(5 : 1 : 1)
(5 : 12 : 1)
(6 : 1 : 1)
(6 : 12 : 1)
曲线2的所有解为：
(0 : 1 : 0)
(5 : 0 : 1)
(7 : 1 : 1)
(7 : 12 : 1)
(8 : 4 : 1)
(8 : 9 : 1)
(9 : 1 : 1)
(9 : 12 : 1)
(10 : 1 : 1)
(10 : 12 : 1)
(11 : 3 : 1)
(11 : 10 : 1)
曲线3的所有解为：
(0 : 0 : 1)
(0 : 1 : 0)
(1 : 4 : 1)
(1 : 9 : 1)
(2 : 5 : 1)
(2 : 8 : 1)
(11 : 1 : 1)
(11 : 12 : 1)
(12 : 6 : 1)
(12 : 7 : 1)
```

其中（0：1：0）为无穷远点，其他的点都是有理点。

3. 回顾本章例题1.7中 $\mathbb{Z}_{13}$ 上的椭圆曲线方程

$$E: y^2 = x^3 + 7x + 6 \pmod{13}$$

其对应的群E($\mathbb{Z}_{13}$)是含有11个元素的循环群，具体元素如下：

$$\{O,(1,1),(1,12),(5,6),(5,7),(6,2),(6,11),(10,6),(10,7),(11,6),(11,7)\}$$

请基于上述群E($\mathbb{Z}_{13}$),给出椭圆曲线ElGamal公钥加密的实例。

- 循环群的描述：

```
F=GF(13)
E=EllipticCurve(F,[7,6])
q=E.order()
g=E(5,6)
```

- 密钥生成：
  - 首先由于该椭圆曲线有11个元素，所以该椭圆曲线的阶 $q = 11$
  - 由于该椭圆曲线对应的群E是循环群，所以可以任选一个作为群的生成元 $g = (5,6)$
  - 接下来随机的选取 $x \in \mathbb{Z}_p$ ，计算 $X \leftarrow x \cdot g$ 。
  - 于是就得到了公钥 $pk = (\mathbb{G}, q, g, X)$ ，私钥 $sk = (\mathbb{G}, q, g, x)$

```
def key_gen():
    x=F.random_element()
    X=g*int(x)
    pk=(q,g,X)
    sk=(q,g,x)
    return pk,sk
```

- 加密：
  - 输入公钥 $pk = (\mathbb{G}, q, g, X)$ ，消息 $m \in \mathbb{G}$ ，随机选择 $y \in \mathbb{Z}_p$ ，计算：

$$c_1 \leftarrow y \cdot g, c_2 \leftarrow X \cdot y + m$$

  - 输出密文 $c = (c_1, c_2)$

```
def Enc(pk,m):
    y=F.random_element()
    c1=E(g)*int(y)
    c2=E(pk[2])*int(y)+m
    c=(c1,c2)
    return c
```

- 解密：
  - 输入私钥 $sk=(\mathbb{G},q,g,x)$ ，密文 $c=(c_1,c_2)$ ，计算

$$m \leftarrow c_2-(x\cdot c_1)$$

  - 输出明文 m

```
def Dec(sk,c):
    m=c[1]-int(sk[2])*c[0]
    return m
```

- 实例：
  - 设待加密的明文为 $(11,7)$

```
Pk,Sk=key_gen()
print("公钥：",Pk)
print("私钥：",Sk)
m = (11,7)
C=Enc(Pk,m)
print("加密得到的密文：",C)
M=Dec(Sk,C)
print("解密得到的明文：",M)
```

  - 运行效果：

```
➜ 密码学引论 sage test.sage
公钥： (11, (5 : 6 : 1), (5 : 6 : 1))
私钥： (11, (5 : 6 : 1), 12)
加密得到的密文： ((6 : 11 : 1), (6 : 2 : 1))
解密得到的明文： (11 : 7 : 1)
```

4. $Q=(5,0)$ 不是曲线 $E: y^2=x^3+7x+6 \pmod{13}$ 上的点，若直接对该点进行点加运算法则计算 kQ ，请分析 kQ 的计算结果。

   注意到：$Q=-Q\ \Rightarrow\ 2Q=\ \mathcal{O}\ \Rightarrow\ 3Q=Q$

   故：$kQ=\begin{cases}\mathcal{O} & k\text{为偶数}\\ Q & k\text{为奇数}\end{cases}$

练习1.17

利用Pohlig-Hellman算法求解 $log_a b$

1. $p=41,a=6,b=29$
   a. 分解 $p-1$， $p-1=2^3\ \times\ 5$
   b. $a=2^0a_0+2^1a_1+2^2a_2(mod2^3)$
   c. 计算 a
      i. $a^{\frac{p-1}{2}a_0}\equiv b^{\frac{p-1}{2}}\ (mod41)\ \Rightarrow\ 6^{20a_0}\ \equiv 29^{20}(mod41)\Rightarrow\ a_0=1$
      ii. $a^{\frac{p-1}{2^2}(1+2a_1)}\equiv b^{\frac{p-1}{2^2}}\ (mod41)\ \Rightarrow\ a_1=1$
      iii. $a^{\frac{p-1}{2^3}(1+2\times1+2^2a_2)}\equiv b^{\frac{p-1}{2^3}}\ (mod41)\ \Rightarrow\ a_2=1$
      iv. $a=2^0\ \cdot\ 1+2^1\ \cdot\ 1+2^2\ \cdot\ 3=7(mod8)$
   d. $b=b_0(mod5)$
   e. 计算 b
      i. $a^{\frac{p-1}{5}b_0}\equiv b^{\frac{p-1}{5}}\ (mod41)\ \Rightarrow\ 6^{8b_0}\ \equiv 29^8(mod41)\Rightarrow\ b_0=2$
      ii. $b=2(mod5)$
   f. 中国剩余定理：
      i. $\begin{cases} x=7(mod8) \\ x=2(mod5) \end{cases}\ \rightarrow\ x=7(mod40)$
2. $p=37,a=2,b=29$

   同上。

- 算法实现：

```
#sagemath
def Polig_Hellman(g,y,p):
    factors, exponents = zip(*factor(p-1))
    temp=[]
    for i in range(len(factors)):
        q=factors[i]
        a=[]
        for j in range(1,exponents[i]+1):
            gg=pow(g,((p-1)//q^j),p)
            yy=pow(y,((p-1)//q^j),p)
            for k in range(q):
                s=0
                for t in range(len(a)):
                    s+=a[t]*q^t
                s+=k*q^(len(a))
                if pow(gg,s,p)==yy:
                    a.append(k)
```

```
                    break
        x_q=0
        for j in range(len(a)):
            x_q+=a[j]*q^j
        temp.append(x_q)
    f=[]
    for i in range(len(factors)):
        f.append(factors[i]^exponents[i])
    return crt(temp,f)

print(Polig_Hellman(6,29,41))
print(Polig_Hellman(2,29,37))
```

- 运行效果：

```
➜ 密码学引论 sage pohlig-hellman.sage
7
21
```

- 正确性验证：

  可以使用sagemath里求解离散对数的内置函数 `discrete_log()` 来计算离散对数。

```
F=GF(41)
print(discrete_log(F(29),F(6))==Polig_Hellman(6,29,41))
G=GF(37)
print(discrete_log(G(29),G(2))==Polig_Hellman(2,29,37))
```

```
➜ 密码学引论 sage pohlig-hellman.sage
True
True
```