# 数学基础（椭圆曲线）

# 椭圆曲线

在数学上，任何满足以下方程的点所形成的曲线称为随机椭圆曲线：

$$y^2 = x^3 + ax + b$$

并且 $4a^3 + 27b^2 \neq 0$，a和b可以为任意值。下面展示几个随机椭圆函数的示例：

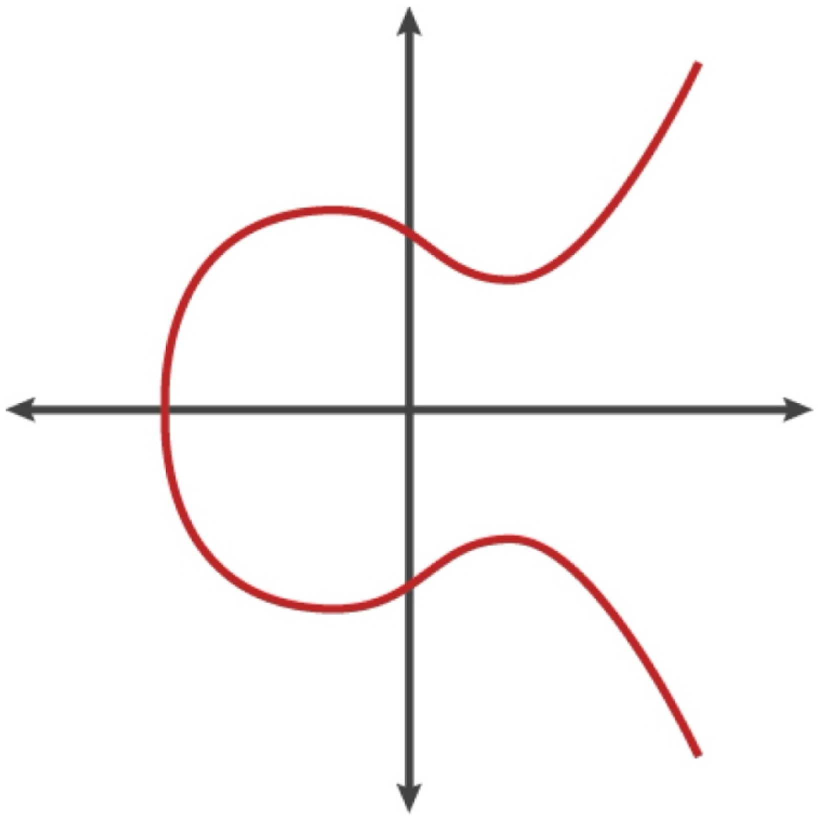
Weierstrass 方程，比特币使用了 secp256k1 标准所定义的一条特殊的椭圆曲线和一系列数学常数。该标准由美国国家标准与技术研究院(NIST)设立。 secp256k1 曲线由下述函数定义，该函数可产生一条椭圆曲线: $y_2 = (x_3 + 7)$} over ($F_p$)

或

$y_2 \bmod p = (x_3 + 7) \bmod p$

上述*mod p*(素数p取模)表明该曲线是在素数阶*p*的有限域内，也写作$F_p$，其中p $= 2_{256} - 2_{32} - 2_9 - 2_8 - 2_7 - 2_6 - 2_4 - 1$， 这是一个非常大的素数。

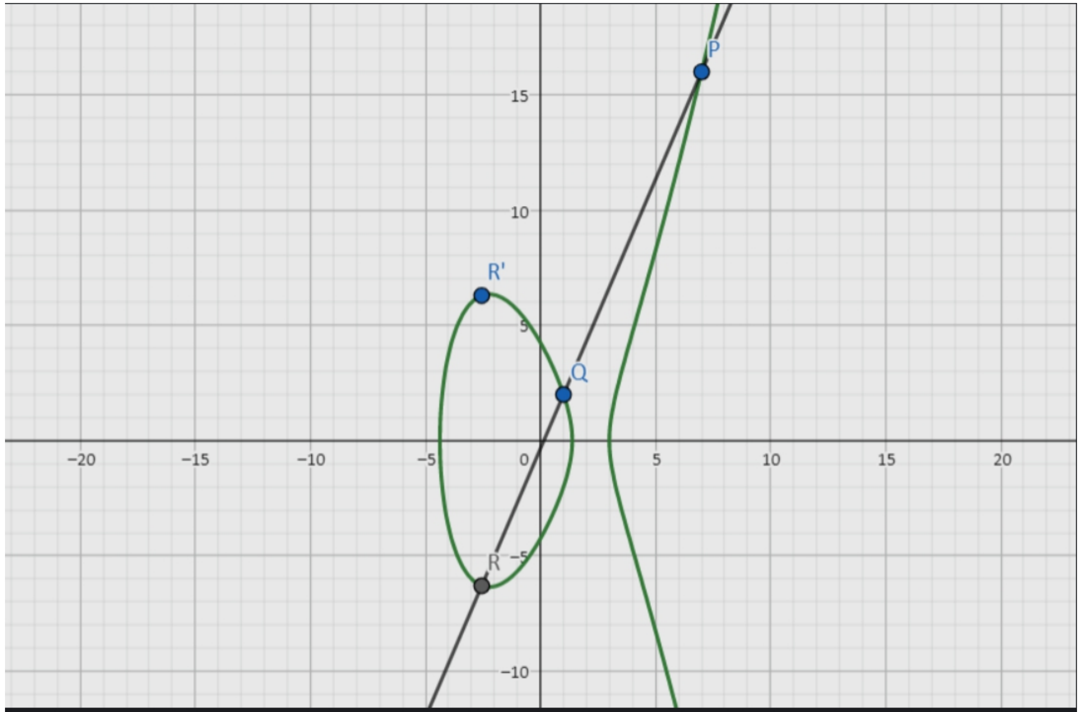
因为这条曲线被定义在一个素数阶的有限域内，而不是定义在实数范围，它的函数图像看起来像分散在两个维度上的散点 图，因此很难画图表示。不过，其中的数学原理与实数范围的椭圆曲线相似。作为一个例子，下图显示了在一个小了很多的 素数阶17的有限域内的椭圆曲线，其形式为网格上的一系列散点。而 secp256k1 的比特币椭圆曲线可以被想象成一个极大的 网格上一系列更为复杂的散点。



# 椭圆曲线加法

我们可以在椭圆曲线 E 上定义“加”的运算：假设有两个点 P,Q，则 P 与 Q 的和 P⊕Q 由如下步骤得到：

连接 PQ 形成一条直线，这条直线交 E 于 R 点。
R 相对于 x 轴的对称点，就是 P⊕Q 的结果




▲在 $E: Y^2 = X^3 - 15X + 8$ 上计算 $(7,16)+(1,2)$

首先，需要获取 PQ 这条直线的表达式。显然斜率

$$\lambda = \frac{x_P - x_Q}{y_P - y_Q} = \frac{7}{3}$$

于是得到了直线的表达式：

$$PQ: Y = \frac{7}{3}X - \frac{1}{3}$$

得到如下方程：

$$X^3 - \frac{49}{9}x^2 - \frac{121}{9}X + \frac{161}{9} = 0$$

我们懒得直接解这个一元三次方程。注意到直线与 E 交于三个点，那么上面这条方程等价于

$$(x-7)(x-1)(x-e_3) = 0$$

观察两个等价方程的零次项，有

$$-7e_3 = \frac{161}{9} \quad \Rightarrow \quad e_3 = \frac{23}{9}$$

x轴坐标知道了，y轴左边代进直线就知道。于是我们得到 $R = (-\frac{23}{9}, -\frac{170}{27})$，于是

$$P \oplus Q = R' = (-\frac{23}{9}, \frac{170}{27})$$

# 椭圆曲线加法的性质

在椭圆曲线上，$\oplus$ 满足加法的性质，以下简写为 “+”。我们有：

- 有单位元：$P+O=O+P=P$.
- 有逆元：$P+(-P)=\mathcal{O}$，其中 $-P$ 是 $P$ 关于 x 轴的对称点。
- 结合律：$(P+Q)+R=P+(Q+R)$
- 交换律：$P+Q=Q+P$

以上每一条性质都可以手工验证。从而 $(E,+)$ 是一个阿贝尔群。于是立刻可以定义数乘运算：

> 对于正整数 $n$ 和椭圆曲线上的点 $P$，定义
>
> $$nP=P+P+P+\cdots+P$$
>
> 显然，$nP$ 可以通过类似于快速幂的算法，以 $O(\log n)$ 次加法的代价求出来。
> 给定 $P$ 和 $Q=nP$，求解 $n$，就是椭圆曲线上的离散对数问题。

# 椭圆曲线加法P+P

举个例子。还是用刚刚的椭圆曲线，$P=(7,16)$。为了计算 $P\oplus P$，需要先求出 $P$ 点的切线方程。把方程 $E$ 左右都取微分，有

$$2ydy=(3x^2-15)dx$$

从而在 $P$ 点，有

$$\frac{dy}{dx}=\frac{3x^2-15}{2y}=\frac{33}{8}$$

于是得到直线方程：

$$L:Y=\frac{33}{8}X-\frac{103}{8}$$

回代到曲线 $E$:

$$X^3-\frac{1089}{64}X^2+\frac{2919}{32}X-\frac{9457}{64}=0$$

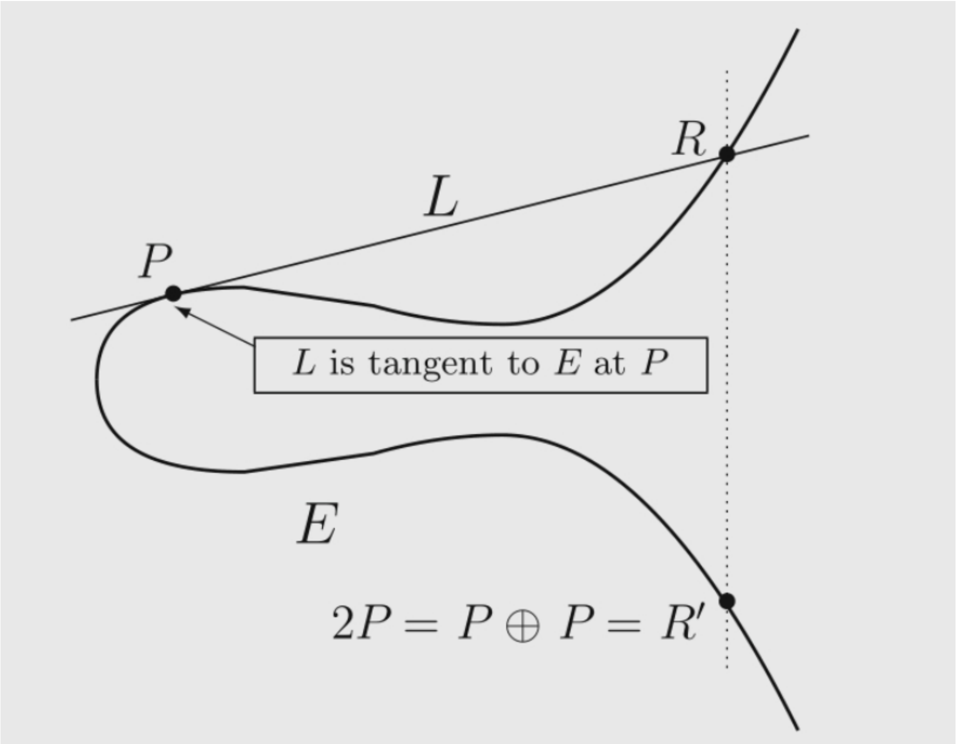
这个方程等价于交点方程：

$$(x-7)(x-7)(x-e_3)=0$$

考察零次项，有

$$-49e_3=-\frac{9457}{64}\quad\Rightarrow\quad e_3=\frac{193}{64}$$

于是终于得到 $P\oplus P=(\frac{193}{64},\frac{223}{512})$



# 离散对数椭圆曲线加法群

最后，我们给出离散对数加法算法的精确描述。设 $E: Y^2 = X^3 + AX + B$ 是椭圆曲线，则可以使用如下算法计算 $P_1 + P_2$:

1. 若 $P_1 = \mathcal{O}$，则 $P_1 + P_2 = P_2$.
2. 否则，若 $P_2 = \mathcal{O}$，则 $P_1 + P_2 = P_1$.
3. 否则，记 $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$.
4. 若 $x_1 = x_2$ 且 $y_1 = -y_2$，则 $P_1 + P_2 = \mathcal{O}$.
5. 否则，记

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P_1 \neq P_2 \\ \frac{3x_1^2 + A}{2y_1}, & P_1 = P_2 \end{cases}$$

令

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1$$

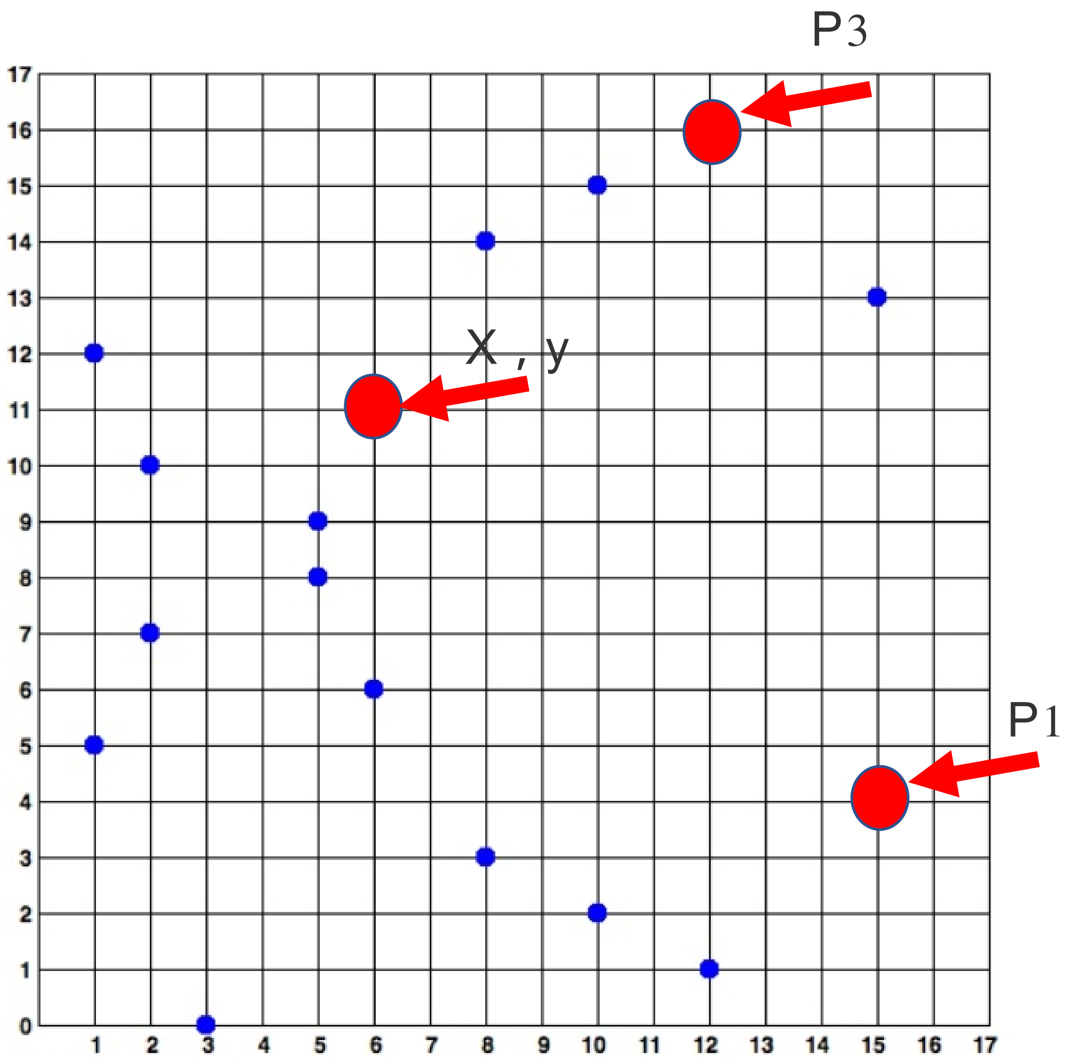
则 $P_1 + P_2 = (x_3, y_3)$.

# 椭圆曲线有限域

以p=17为例，可以在图上画出这些点：
P1(15,4)，P2(15,13)，P3(12,16)， P3(12,2)
可用python来验证其满足
$y^2 \bmod p = (x^3 + 7) \bmod p$

*例如：*

```
>>> 15**3+7-4**2
3366
>>> 3366%17
0
```



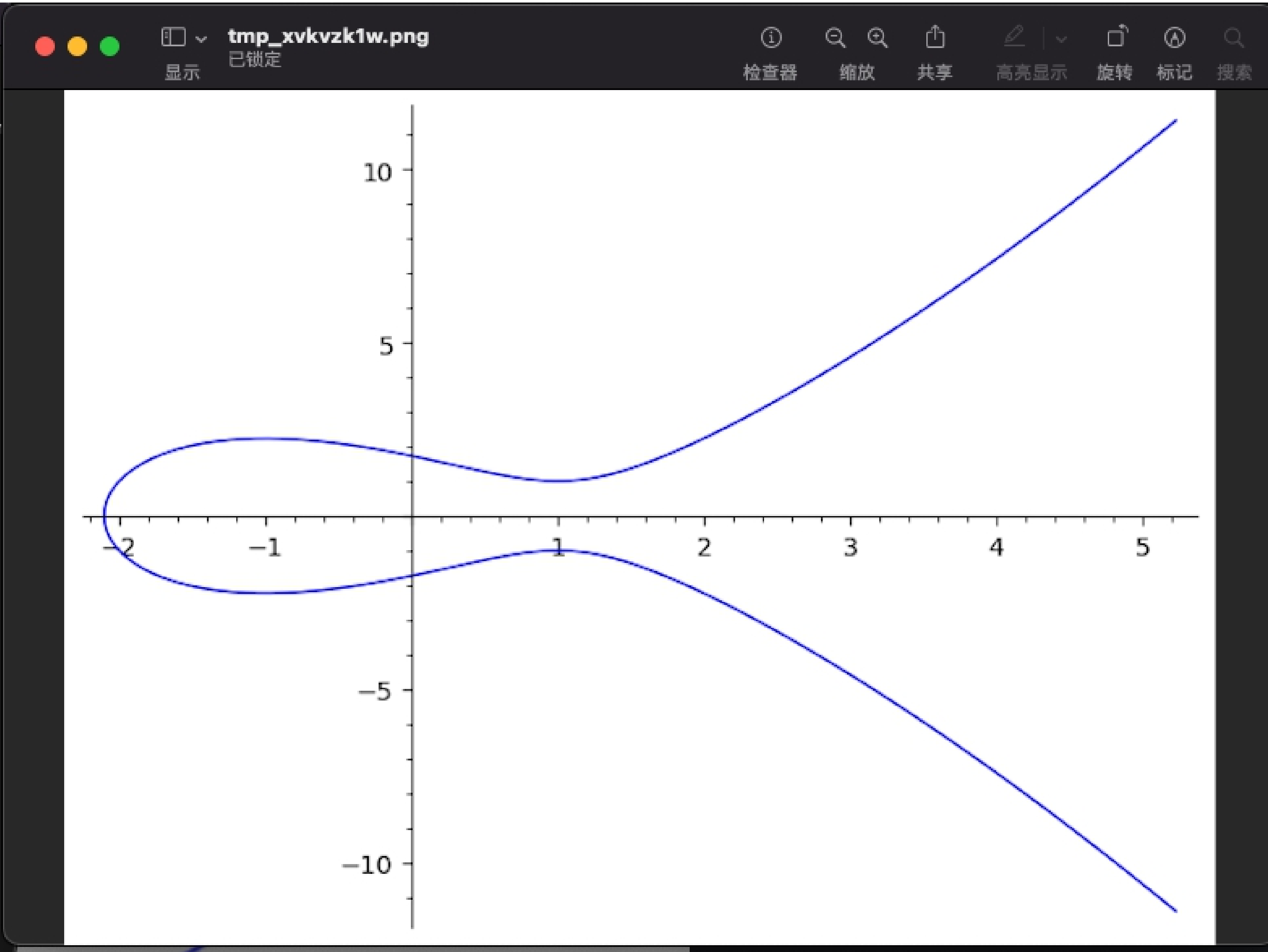

如何求P1+P3？
Step1: 求P1, P3两点决定的斜率k=-4, 在有限域里k=13 mod p
Step2:
$X = k^2 – P1.x - P2.x = 13*13-15-12 \bmod 17 = 6$
$Y = k(P1.x-X)-P1.y \bmod 17 = 13(15-6)-4=113=11 \bmod 17$
验证(X, Y)在椭圆曲线上：(直观看图也能得出结果)
$x^3 + 7 - y^2=102=0 \bmod 17$

# 用sagemath来加深对ECC的理解

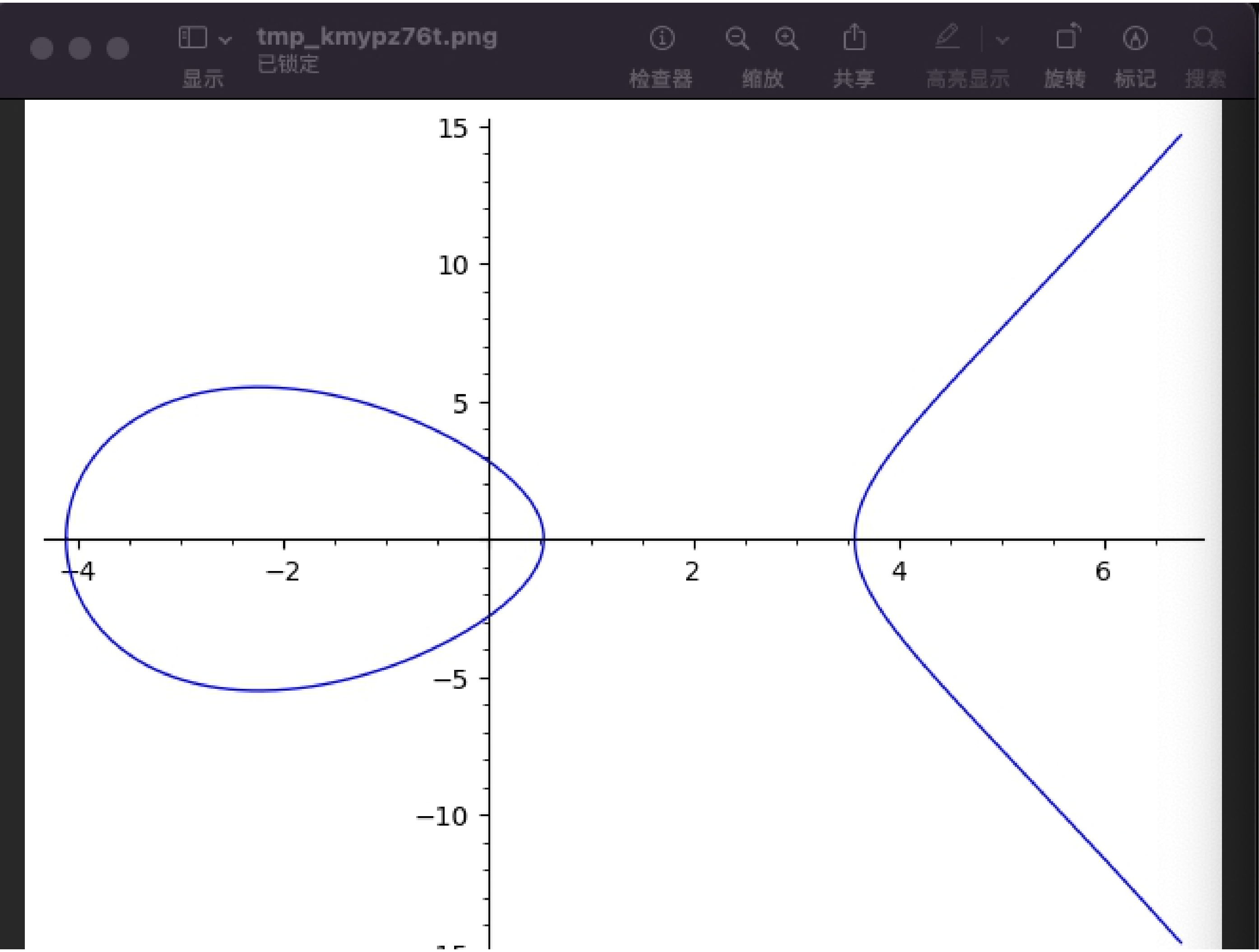
SageMath — IPython: Downloads/SageMath — python3 ‹ sage-python ~/...

```
s/config/loader.py:804: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if len(key) is 1:

/Users/caiqingfeng/Downloads/SageMath/local/lib/python3.9/site-packages/sage/co
binat/posets/poset_examples.py:158: DeprecationWarning: invalid escape sequence
\{
  """
sage:
sage: quit()
Exiting Sage (CPU time 0m0.07s, Wall time 1m38.81s).
caiqingfeng@MacBook-Pro-2 SageMath % ./sage
┌────────────────────────────────────────────────────────────────────┐
│ SageMath version 9.4, Release Date: 2021-08-22                     │
│ Using Python 3.9.5. Type "help()" for help.                        │
└────────────────────────────────────────────────────────────────────┘
sage: E=EllipticCurve([-3,3])
sage: show(E)
Elliptic Curve defined by y^2 = x^3 - 3*x + 3 over Rational Field
sage: E.plot()
Unable to revert mtime: /Library/Fonts
Launched png viewer for Graphics object consisting of 1 graphics primitive
sage: 
```



# 用sagemath来加深对ECC的理解




SageMath — IPython: Downloads/SageMath — python3 ◂ sage-python...

```
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-7-1895bea54695> in <module>
----> 1 E(Integer(0)).xy()

~/Downloads/SageMath/local/lib/python3.9/site-packages/sage/schemes/elliptic_c
urves/ell_point.py in xy(self)
    753                 return self[0], self[1]
    754             else:
--> 755                 return self[0]/self[2], self[1]/self[2]
    756
    757     def is_divisible_by(self, m):

~/Downloads/SageMath/local/lib/python3.9/site-packages/sage/rings/rational.pyx
 in sage.rings.rational.Rational.__truediv__ (build/cythonized/sage/rings/rati
onal.cpp:21281)()
   2448             if type(left) is type(right):
   2449                 if mpq_cmp_si((<Rational> right).value, 0, 1) == 0:
-> 2450                     raise ZeroDivisionError('rational division by zero')
   2451                 x = <Rational> Rational.__new__(Rational)
   2452                 mpq_div(x.value, (<Rational>left).value, (<Rational>right)
.value)

ZeroDivisionError: rational division by zero
sage: E=EllipticCurve(RealField(), [-15,8])

sage: show(E)

Elliptic Curve defined by y^2 = x^3 - 15.0000000000000*x + 8.00000000000000 ov
er Real Field with 53 bits of precision
sage: E.plot()

Launched png viewer for Graphics object consisting of 2 graphics primitives
sage: 
```

# 椭圆曲线离散对数问题

经典的椭圆曲线上的离散对数问题，是给定点 $P$ 和 $nP$，要推出 $n$ 的值。有限域椭圆曲线上的离散对数问题亦然。

> 设 $E$ 是有限域 $\mathbb{F}_p$ 上的椭圆曲线，$P, Q \in E(\mathbb{F}_p)$. 椭圆曲线离散对数问题(Elliptic Curve Discrete Logarithm Problem) 是指：找到一个整数 $n$ 使得 $Q = nP$. 我们记
>
> $$n = \log_P(Q)$$
>
> 并称 $n$ 是以 $P$ 为底 $Q$ 的椭圆曲线离散对数。

举一个离散对数的例子。考虑 $\mathbb{F}_{73}$ 上的椭圆曲线

$$E : Y^2 = X^3 + 8X + 7$$

其上有点 $P = (32, 53)$ 和点 $Q = (39, 17)$. 计算得知 $Q = 11P$，于是 $\log_P(Q) = 11$.

```
E = EllipticCurve(GF(73), [8, 7])
P = E(32, 53)
Q = E(39, 17)

P.discrete_log(Q)
# 11
```

# 椭圆曲线离散对数难题（ECDLP）

## 4. ECDLP

**椭圆曲线离散对数问题**(Elliptic Curve Discrete Logarithm Problem, **ECDLP**)

椭圆曲线上的两个点$P$和$Q$，$k$为整数。

$$Q = kP.$$

**椭圆曲线加密的数学原理：**

点$P$称为基点（base point）；$k$为私有密钥（private key）；$Q$为公开密钥（public key

- 则给定$k$和$P$，根据加法法则，计算$Q$很容易。
- 但给定$P$和$Q$，求$k$非常困难（实际应用ECC，质数$p$取得非常大，穷举出$k$非常困难）.



**举个椭圆曲线离散对数问题（ ECDLP ）的例子**

椭圆曲线上的点集$E_{23}(1,1)$：

$$y^2 = x^3 + x + 1 \mod 23$$

设$P = (3,10)$, **则求得$10P$，$38P$，$66P$，⋯，$nP$如下：**

| $10P$ | $38P$ | $66P$ | $94P$ | $122P$ | ⋯ | $nP$ |
|---|---|---|---|---|---|---|
| (6, 4) | (6, 4) | (6, 4) | (6, 4) | (6, 4) | ⋯ | (6, 4) |

**椭圆曲线加密的数学原理：**

点$P$称为基点（base point）；$k$为私有密钥（private key）；$Q$为公开密钥（public key)

- 则给定$k = 66$和$P = (3,10)$，根据加法法则，计算$Q = (6, 4)$很容易。
- 但给定$P = (3,10)$和$Q = (6, 4)$，求得$k = 10$、38、64、⋯。（即求$k$非常困难）