CTF 学习笔记二——ECC 加密

椭圆加密算法（ECC）是一种公钥加密体制，最初由 Koblitz 和 Miller 两人于 1985 年提出，其数学基础是利用椭圆曲线上的有理点构成 Abel 加法群上椭圆离散对数的计算困难性。公钥密码体制根据其所依据的难题一般分为三类：大素数分解问题类、离散对数问题类、椭圆曲线类。有时也把椭圆曲线类归为离散对数类。

之所以称其为椭圆曲线加密，是因为这种加密方式是在椭圆曲线方程上进行操作。即形如$y^2 = x^3 + ax + b$，$4a^3 + 27b^2 \neq 0$的方程。

由于在标准的椭圆曲线方程上进行运算会出现小数或者无理数，从而导致精度的问题，所以我们通过取模，将椭圆曲线上的点变成整数点，并构成一个封闭群，便于后续运算。即变形成$y^2 \equiv x^3 + ax + b(modp)$，$4a^3 + 27b^2 \neq 0$的形式，其中 p 一般取一个大素数。

椭圆曲线也可以有运算，像实数的加减乘除一样，这就需要使用到加群。19 世纪挪威的尼尔斯•阿贝尔抽象出了加群（又叫阿贝尔群或交换群）。数学

中的群是一个集合，我们为它定义了一个“加法”，并用符号+表示，遵循以下四个特性：

- 封闭性：如果 a 和 b 都是封闭群的成员，那么 a+b 也是封闭群的成员；
- 结合律：(a + b) + c = a + (b + c)；
- 单位元：a+0=0+a=a，0 就是封闭群的单位元；
- 逆元：对于任意值 a 必定存在 b，使得 a+b=0。

如果再增加一个条件，交换律：a + b = b + a，则称这个群为阿贝尔群，根据这个定义整数集是个阿贝尔群。

在椭圆曲线上，加法的运算定义如图 1 所示：过曲线上的两点 A、B 画一条直线，找到直线与椭圆曲线的交点，交点关于 x 轴对称位置的点，定义为 A+B，即为加法。如下图所示：A + B = C

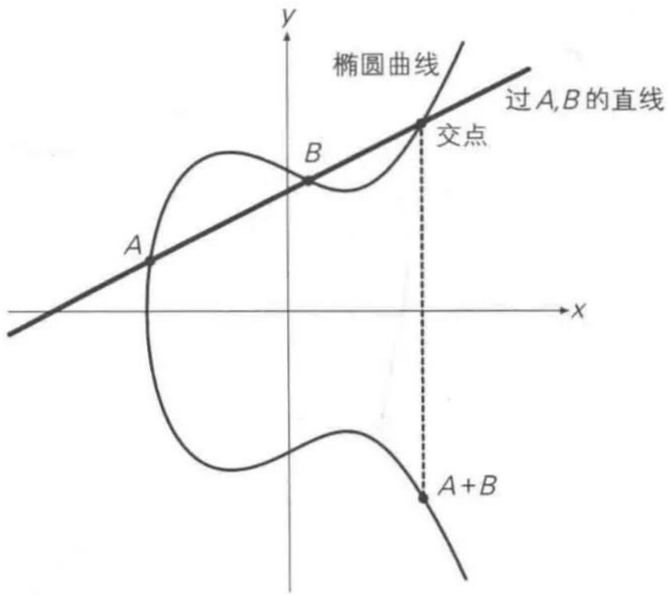

图 1 椭圆曲线加法运算

不妨举一个例子，假设椭圆曲线的方程为$y^2=x^3+1$，设点 A 的坐标为(2,3)，点 B 的坐标为(0,1)，那么直线 AB 的方程为 y=x+1，代入方程可知$x^3-x^2-2x=0$，于是可很快求出第三个交点 C 的坐标(-1,0)，也就是 C=A+B。当然，这里 C 也满足 C=B+A，所以椭圆曲线群是一个阿贝尔群。

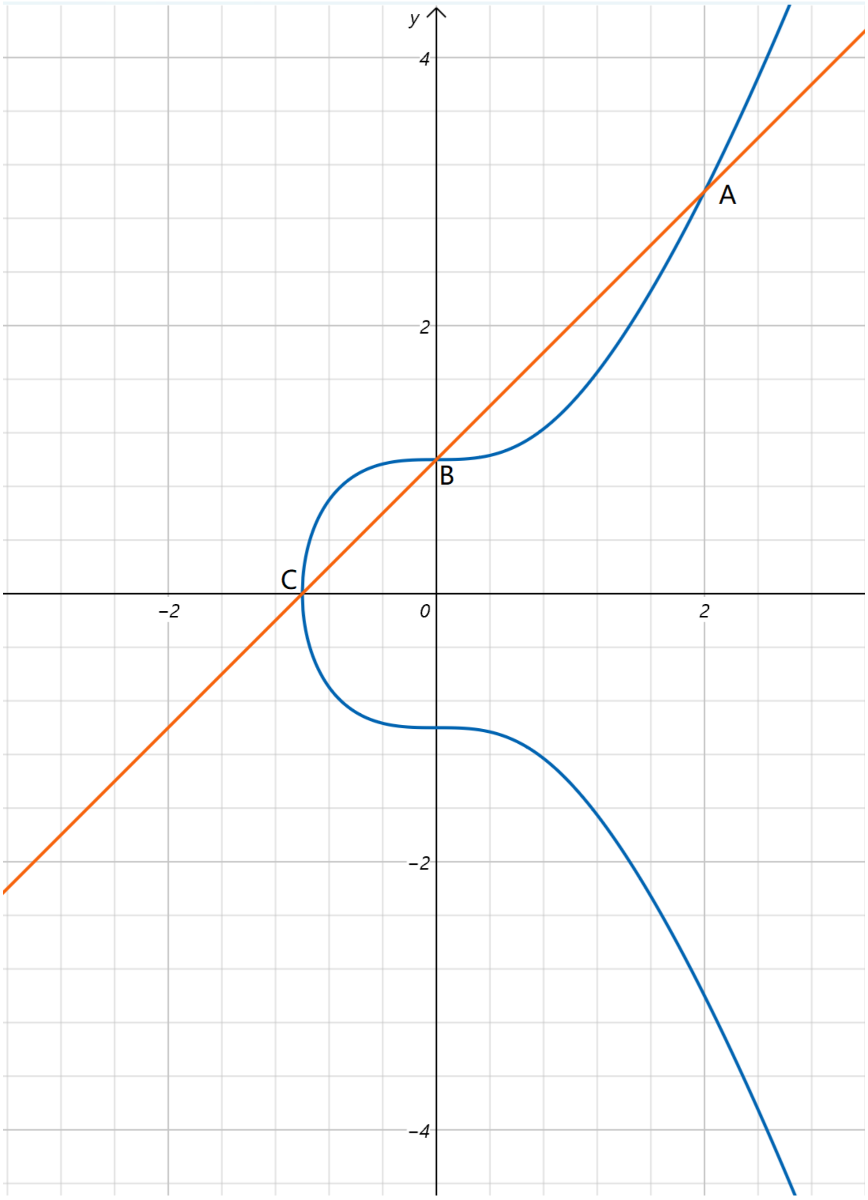

图 2 椭圆曲线加法运算

根据这个定义，我们来讨论一下一般情形下椭圆曲线的加法运算公式，假设椭圆曲线的方程为$\boldsymbol{y^2 \equiv x^3 + ax + b(modp)}$，$\boldsymbol{4a^3 + 27b^2 \neq 0}$，设 A 的坐标为$(\boldsymbol{x_1}，\boldsymbol{y_1})$，B 的坐标为$(\boldsymbol{x_2}，\boldsymbol{y_2})$，那么直线 AB 的方程为$\boldsymbol{y = kx - kx_1 + y_1}$，$\boldsymbol{k = \frac{y_2 - y_1}{x_2 - x_1}}$。带入椭圆曲线的方程，有$\boldsymbol{x^3 - k^2x^2 - 2k(-kx_1 + y_1)x + ax +}$

$b-(-kx_1+y_1)^2=0$，根据韦达定理，如果设直线 AB 与椭圆曲线的第三个交点为（x_3，y_3），那么就有$x_1+x_2+x_3=k^2$，所以$x_3=k^2-x_1-x_2$，$y_3=k(x_3-x_1)+y_1$，由于最后还要将第三个交点的坐标关于 x 轴翻折一下，所以如果设 C=A+B，那么 C 的坐标为：

$$C=((k^2-x_1-x_2)modp，(k(x_1-x_3)-y_1)modp)$$

特别的，如果计算出来的 C 的坐标是一个分数，也就是形如$\frac{a}{b}$，$a\in Z$，$b\in Z$，$gcd(a,b)=1$的形式。那么$\frac{a}{b}modp$的求法如下：

由于$\frac{a}{b}modp=\left((amodp)\left(\frac{1}{b}modp\right)\right)modp$，设$\frac{1}{b}\equiv n(modp)$，那么$1\equiv nb(modp)$

由费马小定理：如果 p 是一个质数，而整数 a 不是 p 的倍数，则有$a^{p-1}\equiv 1(modp)$。带入上述式子可知，$n=b^{p-2}$。

所以$\frac{a}{b}modp=\left((amodp)(b^{p-2}modp)\right)modp=(ab^{p-2})modp$

这边再举一个例子，设椭圆曲线方程为$y^2\equiv x^3+x+1(mod53)$，A 坐标为(4, 4)，B 坐标为(0, 1)，那么根据公式$k=\frac{3}{4}$，$k^2-x_1-x_2=-\frac{55}{16}$，$k(x_1-x_3)-y_1=\frac{101}{64}$，再根据上述分数取模的算法得到 C=(33, 14)。

特别的，当 A 和 B 的坐标相等时，也就是 A=B 时，C=A+A，这个时候为了书写方便，可写为 C=2A，乘法的计算就诞生了，如果要计算 3A，4A 或者更高的数 nA 时，只需要转化成加法，将 n 个 A 依次按照加法运算即可。这时候直线的斜率 k 就是该点的切线斜率，也就是$\boldsymbol{k=\frac{3x^2+a}{2y}}$。举个例子：假设椭圆曲线的方程为$\boldsymbol{y^2\equiv x^3+1(mod 11)}$，设点 A 的坐标是(2, 3)，那么 k=2，切线方程为 y=2x-1，带入公式求出 2A 的坐标(0, 1)。

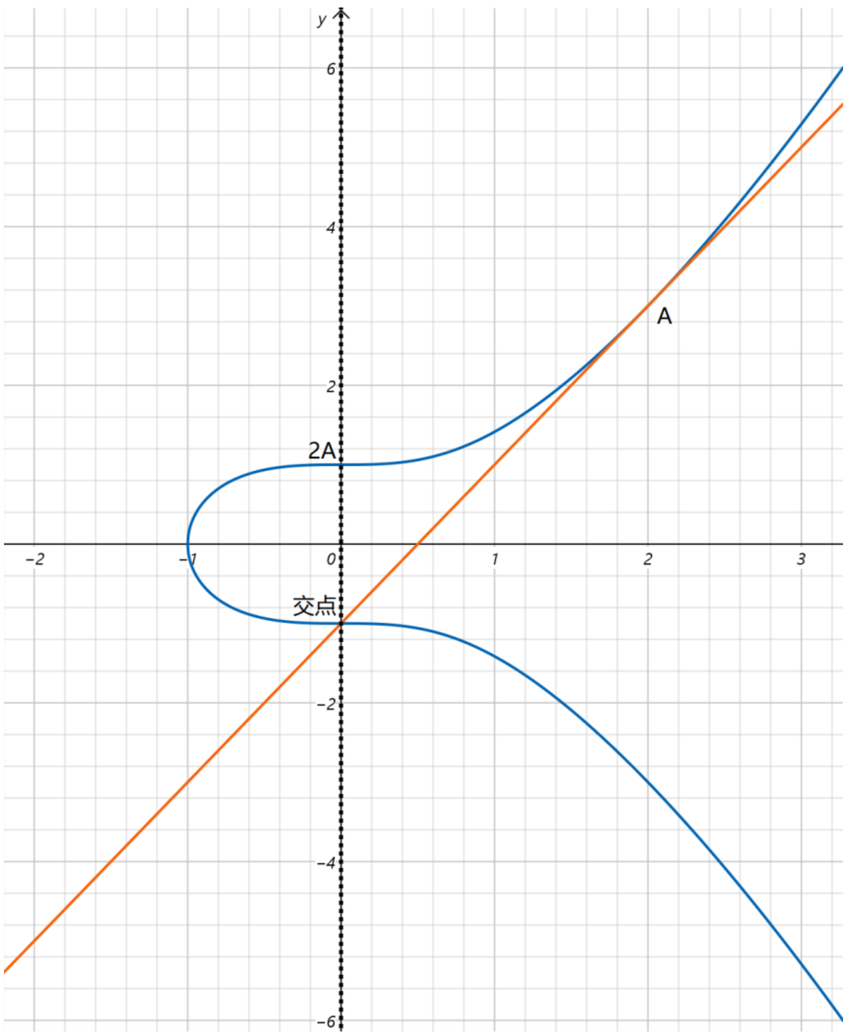

图 3 椭圆曲线乘法运算

椭圆曲线加密的原理就是利用 C=nA 的乘法运算时，知道 n 和 A 可以很快求出 C，但是知道 C 和 A 很难求出 n 的值，下面来看一下椭圆曲线加密的具体流程。

首先定义如下概念：

基点 P：指的是椭圆曲线上满足方程的任意一点，该点会公开。

私钥 k：私钥 k 是一个大整数，该数由发送方和接收方私自保存，不公开。

公钥 Q：公钥的计算法则是将私钥 k 和基点 P 相乘，做为 Q，也就是 Q=kP，公钥也会公开。

明文 M：这里的明文 M 是一个点，假设我们要将一个数 m 进行加密，那么就需要计算出满足$\boldsymbol{y^2 \equiv m^3 + am + b(mod p)}$，$\boldsymbol{4a^3 + 27b^2 \neq 0}$的椭圆方程的一个 y 值构成点(m,y)。那么明文 M 的值就是(m,y)，这个明文不公开。

加密法则 E(M)：加密法则如下，我们先随机选择一个随机数 r，那么就可以得到密文 c，c= E(M)=(rP,M+rQ)，密文 c 是一个点对，密文 c 会公开。

解密法则 D(c)：解密法则如下，M=D(c)= M+rQ-krP。

举个例子：

假设椭圆曲线方程为$y^2 \equiv x^3 + x + 1(mod53)$，发送方和接收方选取的私钥 k=7，基点 P 为(0,1)，假设发送方想要把一个数 4 发送给接收方，那么先将 x=4 带入曲线方程，得到明文 M(4,4)。再根据 k 和 P 计算出公钥 Q=7P=(42, 14)，随机选一个数 r=3，那么密文 c= E(M)=(rP,M+rQ)=((19, 28), (6, 45))。

接收方知道了密文 c 和私钥 k，只需要按照解密法则 M=D(c)= M+rQ-krP 就可以求出明文 M(4,4)。

python 代码如下：

```
#coding:gbk
p = 53
i = lambda x: pow(x, p-2, p)
def add(A, B):#加法运算
    (u, v), (w, x) = A, B
    assert u != w or v == x
    if u == w: m = (3*u*w + 1) * i(v+x)
    else: m = (x-v) * i(w-u)
    y = m*m - u - w
    z = m*(u-y) - v
    return y % p, z % p
def opposite(A):#取 A 的相反数，也就是将 A 变成-A
    return A[0],(-A[1])%p
def mul(t, A, B=0):#乘法运算
```

```python
    if not t:
        return B
    if t%2==0:
        return mul(t//2, add(A,A), B)
    elif B!=0:
        return mul(t//2, add(A,A), add(B,A))
    else:
        return mul(t//2, add(A,A),A)
M=(4,4)
P=(0,1)
r=3
k=7
Q=mul(k,P)
c=(mul(r,P),add(M,mul(r,Q)))
print("公钥 Q 是",Q)
print("密文 c 是",c)
print("明文 M 是",add(c[1],opposite(mul(k,c[0]))))
```

在 CTF 中，ECC 加密是一种较为常见的题型，例如：[攻防世界 (xctf.org.cn)]

已知椭圆曲线加密Ep(a,b)参数为

p = 15424654874903

a = 16546484

b = 4548674875

G(6478678675,5636379357093)

私钥为

k = 546768

求公钥K(x,y)

图 4 题目描述

题目告诉了基点，私钥以及椭圆曲线的方程，让我们求公钥，python 代码如下：

```python
#coding:gbk
p = 15424654874903
a = 16546484
i = lambda x: pow(x, p-2, p)
def add(A, B):#加法运算
    (u, v), (w, x) = A, B
    assert u != w or v == x
    if u == w: m = (3*u*w + a) * i(v+x)
    else: m = (x-v) * i(w-u)
    y = m*m - u - w
    z = m*(u-y) - v
    return y % p, z % p
def opposite(A):#取 A 的相反数，也就是将 A 变成-A
    return A[0],(-A[1])%p
def mul(t, A, B=0):#乘法运算
    if not t:
        return B
    if t%2==0:
        return mul(t//2, add(A,A), B)
    elif B!=0:
        return mul(t//2, add(A,A), add(B,A))
    else:
        return mul(t//2, add(A,A),A)
G=(6478678675,5636379357093)
k=546768
K=mul(k,G)
print("公钥 K 是",K)#公钥 K 是 (13957031351290, 5520194834100)
```