椭圆曲线加密算法

椭圆曲线密码学（英语：Elliptic curve cryptography，缩写为 ECC），一种建立公开密钥加密的算法，基于椭圆曲线数学。椭圆曲线在密码学中的使用是在1985年由Neal Koblitz和Victor Miller分别独立提出的。

ECC的主要优势是在某些情况下它比其他的方法使用更小的密钥——比如RSA加密算法——提供相当的或更高等级的安全。ECC的另一个优势是可以定义群之间的双线性映射，基于Weil对或是Tate对；双线性映射已经在密码学中发现了大量的应用，例如基于身份的加密。不过一个缺点是加密和解密操作的实现比其他机制花费的时间长

1. 椭圆曲线

在数学上，椭圆曲线（英语：Elliptic curve，缩写为 EC）为一代数曲线，被下列式子所定义

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

其是无奇点的；亦即，其图形没有尖点或自相交。

满足此条件的 a b 满足：$4a^3 + 27b^2 \neq 0$

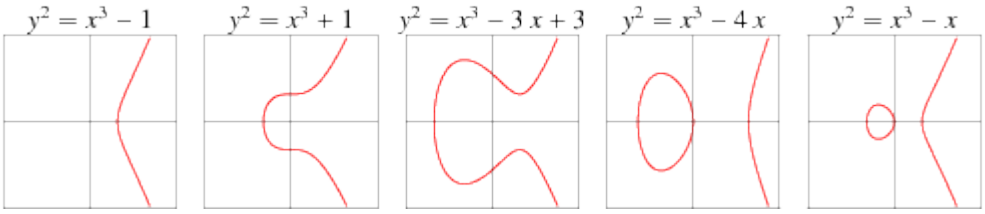

图 1

在基础上需要定义一个无穷远的点，将此点作为零点：此时椭圆曲线定义为：

$$\{(x,y) \in \mathbb{R}^2 | y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0\} \cup \{0\}$$

在椭圆曲线中的群的运算律：

1. 所有的点都在椭圆曲线上
2. 0点作为群上的单元点即

$$P + 0 = P$$

3. P点关于X轴的对称点为P点的逆即

$$P + (-\mathrm{P}) = 0$$

4. 对于位于同一条直线上的三个点P, Q, R. 则有

$$P + Q + R = 0$$

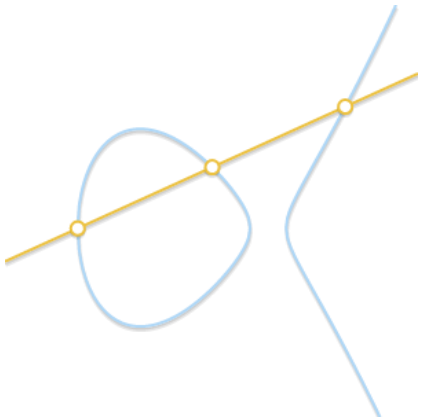

图 2

P+Q+R=0（无限远点

P Q R三个点的位置是任意的，他们满足加法的结合律，因为这个群是一个阿贝尔群。

2. 椭圆曲线加法

当P和Q不相等时($x_P \neq x_Q$)

由于是在阿贝尔群上可以将$P + Q + R = 0$ 改写为 $P + Q = -R$所以在椭圆曲线上的加法定义为P Q 两点加法为P, Q两点连线与曲线的交点R的关于X轴对称点$-R$

图2-3

P+Q=-R

P Q两点的直线的斜率为：

$$m = \frac{y_P - y_Q}{x_P - x_Q}$$

这条线与曲线的交点为：$R = (x_R, y_R)$

$$\begin{aligned} x_R &= m^2 - x_P - x_Q \\ y_R &= y_P + m(x_R - x_P) \end{aligned}$$

因此$(x_P, y_P) + (x_Q, y_Q) = (x_R, -y_R)$ 如果在图上表示即为上述的$P + Q = -R$

当P和Q不相等时$(x_P = x_Q)(y_P = -y_q)$

因为$p + (-p) = 0$

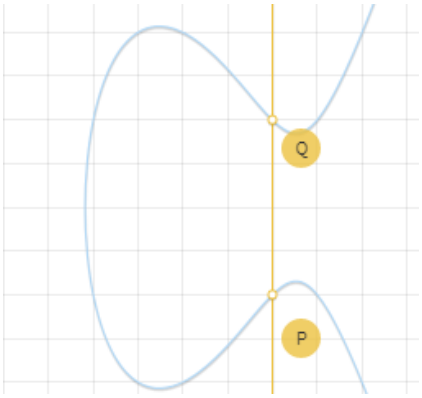

图 3

P Q两点相同时

直线的斜率为

$$m = \frac{3x_P^2 + a}{2y_P}$$

经计算的

$$\begin{aligned} m &= \frac{3x_P^2 + a}{2y_P} \\ x_R &= m^2 - x_P - x_Q \\ y_R &= y_P + m(x_R - x_P) \end{aligned}$$

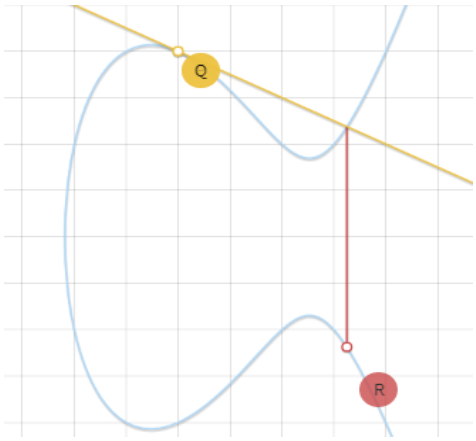

图 4

3. 椭圆曲线标量乘法

通过上面的加法运算我们可以得出其标量乘法运算可以得出

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ times}}$$

从上式可以看出当我们计算nP的时候需要做n次加法，如果n有k位那么的计算时间复杂度变为$O(2^k)$, 这显然不是快捷的方式。

其中一种加快计算的方式是将这些加法运算的次数减少，我们将n写成2进制形式，然后我们按每位计算，每次复用上一次的结果如当$n = 151$时，其二进制位10010111_2 所以

$$\begin{aligned} 151 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2^7 + 2^4 + 2^2 + 2^1 + 2^0 \end{aligned}$$

所以如果运用在上面的标量乘法时，其运算可以简化为：

$$151 \cdot P = 2^7P + 2^4P + 2^2P + 2^1P + 2^0P$$

4. 离散空间椭圆曲线

我们将曲线上的点全部局限于$\mathbb{F}_p$上此时离散形式的椭圆曲线变为：

$$\{(x, y) \in (\mathbb{F}_p)^2 \mid y^2 \equiv x^3 + ax + b (\mathrm{mod} p), 4a^3 + 27b^2 \not\equiv 0 (\mathrm{mod} p)\} \cup \{0\}$$

0仍然为无穷远点，a, b 为整数

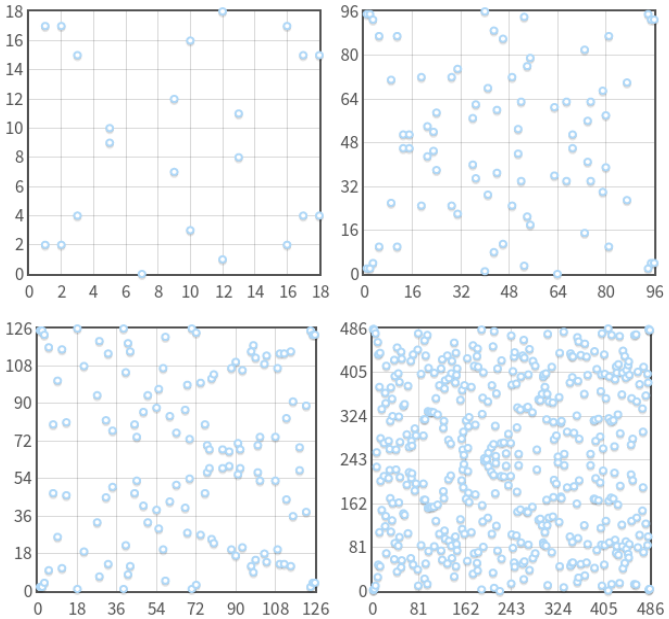

图 5

$y^2 \equiv x^3 - 7x + 10(\mathrm{mod}p)$ p = 19,97,127,487的图示

在离散椭圆图上的加法和曲线上的加法的定义类似，在离散的椭圆曲线中，直线的方程变为$ax + by + c \equiv 0(\mathrm{mod}p)$，点P，Q两点的加法的操作也是基于这个定义上下图展示了$y^2 \equiv x^3 - x + 3(\mathrm{mod}127)$曲线上 $P = (16,20)$ 和$Q = (41,120)$两点之间的加法 直线 $y \equiv 4x + 83(\mathrm{mod}127)$ 在此空间上是一组距离相等的平行线

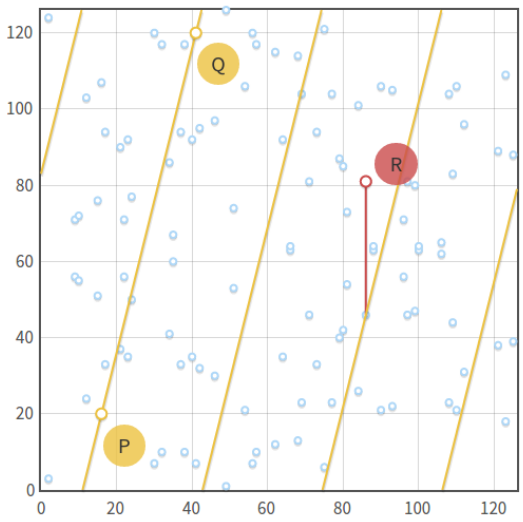

图 6

5. 离散椭圆曲线代数加法

对照曲线上的加法，离散空间椭圆曲线加法公式可以写作

$$\begin{aligned} x_R &= (m^2 - x_P - x_Q)modp \\ y_R &= [y_P + m(x_R - x_P)]modp \\ &= [y_Q + m(x_R - x_Q)]modp \end{aligned}$$

如果$P \neq Q$，此时的m的值的形式为

$$m = (y_P - y_Q)(x_P - x_Q)^{-1}modp$$

如果P = Q时：

$$m = (3x_P^2 + a)(2y_P)^{-1}modp$$

在离散型的椭圆空间里面我们知道点的数目是有限的，而这些点的个数我们称为椭圆曲线群的阶数

同样的在离散空间，对于曲线上的点同样满足标量加法的法则

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ times}}$$

但是在离散空间椭圆曲线加法会有一个有趣的现象，对一个点进行有限次的加法，存在一个是n使得$nP = P$ 例如在曲线$y^2 \equiv x^3 + 2x + 3(\mathrm{mod}97)$的一点$P = (3,6)$，计算其的标量乘法，计算结果如下：

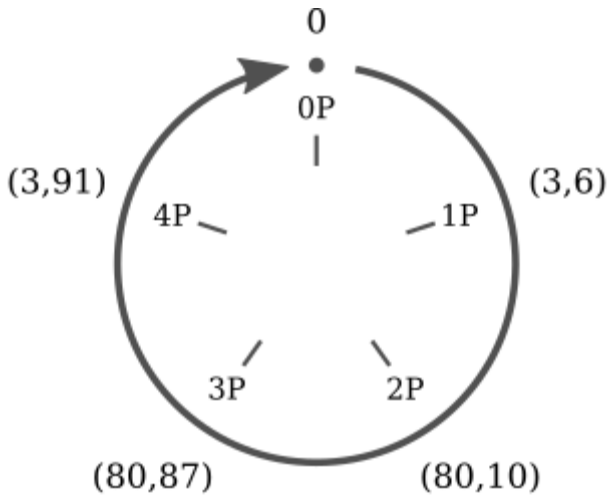

图 7

可以看出$P = (3,6)$经过6次加法后又回到本身，也即是$5P = 0$

所以对于P的加法可以写作$kP = (k \bmod 5)P$

可以证明的是一个点的加法所构成的子集是一个循环的子集

首先一个点的加法所得到的点一定是有限的，因为加法所获得的点全部都在父集上的，父集的大小是有限的，所以经过有限次的加法后所获得的结果必定会重复，那么结果就会重复下去。只需证明第一个重复的值是P即可。

假设第一个重复的值不是P，而是kP, 那么后面的结果也必然是kP的倍数，假设重复的次数为n,

如果$\gcd(n,k)$不为1，那么存在使得$km \bmod n = 0$ 即$kmP = 0$即$(km + 1)P = P$与假设矛盾

如果$\gcd(n,k) = 1$那么$kmP = (m \bmod n)\ kP$ 那么存在一个数使得$mk \bmod n = 1$即$mkP = (mk \bmod n)P = P$与假设矛盾。

在一个循环子集中将P称之为生成点或者基点，其子集的点的大小n称之为子集的阶即$nP = 0$

由拉格朗日定理可知子群的阶n是全集阶N的因子

例如$y^2 = x^3 - x + 3$在$\mathbb{F}_{37}$上$\mathrm{N} = 42$ 它的子集大小只能$\mathrm{n} = 1,2,3,6,7,14,21,42$为基点$\mathrm{P} = (2,3)$, 因为$P \neq 0$，$2P \neq 0 \cdots 7P = 0$ 所以构成的子集的大小为7

如果在$\mathbb{F}_{29}$上，其阶为N=37 所以其子集大小只能为$\mathrm{n} = 1$或者 37

对于ECC（椭圆加密算法），需要找到一个具有高阶子集，一般情况下选择一个椭圆曲线并计算他的阶，然后找到一个数字大的因子n作为子集的阶，如果这个子集的基点为P则$\mathrm{nP} = \mathrm{Np} = 0$

给定n和P，那么需要计算n次计算出$\mathrm{Q} = \mathrm{nP}$, 如果只给出P, Q如何计算出n这就是所谓的对数问题。这个问题也就是椭圆曲线的离散对数问题而求解这个对数问题是非常困难的，而正是由于这种困难使得我们使用椭圆加密算法变得更加可靠

与其他类似的加密算法相比，破解椭圆加密算法更加困难，这也意味着更少位数的秘钥k可以达到同类算法同样的安全强度

6. 椭圆加密算法 ECDH 和 ECSDA

椭圆加密算法是基于有限域的离散型的椭圆曲线的循环子集上，定义一个这样的子集需要以下参数，

- 有限域的大小素数P
- 椭圆曲线方程a和b
- 用于生成子集的基点G
- 子集的阶n
- 子集的辅因子 h

总的来说一个椭圆加密算法基于(p, a, b, G, n, h)参数上

(1) ECDH

同其他非对称加密算法一样，需要计算私钥和公钥，

在ECC中私钥是一个随机数d来自于$\{1, \ldots, n-1\}$(n是子集的阶)

公钥是曲线上的一个点$\mathrm{H} = \mathrm{dG}$（G是子集的基点）

如果我们知道d和G，找到公钥H是很容易的，但是如果只知道H和G找到私钥d是及其困难的，这等同有在做离散对数问题，上文有提到这个问题在现在来说还没有有效的算法，所以是很困难的。

ECDH是迪夫赫尔曼算法的一个变种，它实际上是一个秘钥共识协议，ECDH协议定

义了秘钥的如何产生以及如何交换秘钥。

如果Alice和Bob需要交换秘钥，中间人Hack可以在两者之间监听

1. Alice和Bob使用同样参数的椭圆曲线，并且随机产生各自的秘钥d_A和d_B,并且计算出各自的公钥$H_A = d_A G$和$H_B = d_B G$,
2. 他们彼此交换各自的公钥H_A和H_B,而中间人也将他们的公钥截获，但是他们都不能从这些信息中计算出他们的私钥
3. Alice计算出$S = d_A H_B$，Bob计算出$S = d_A H_B$，明显的是他们各自计算出的结果是一致的。

$$S = d_A H_B = d_A(d_B G) = d_B(d_A G) = d_B H_A$$

而中间人不能从公钥中计算出S，所以Alice和Bob共享了密码S,这也正是迪夫赫尔曼所要解决的问题。

当Alice和Bob获得了共同的秘钥S，他们就可以使用对称加密算法如AES，3DES来加密他们的信息了。

(2) ECSDA签名算法

在比特币的交易中，一笔交易会在整个区块链网络中传播，每个节点需要验证一笔交易的有效性，通过交易中解锁脚本去验证交易中的输入是否是有效的。这其中正是使用了ECDSA算法来验证。

假如Alice需要对一个信息进行签名,Bob需要使用Alice的公钥来验证这个信息。Alice和Bob都使用的是同一个参数相同的椭圆曲线函数。ECDSA作用在的是信息的hash值上的，且这个值是被截取后的使得其大小与这个曲线的子集的阶一致。将这个截取后的hash值定义为z

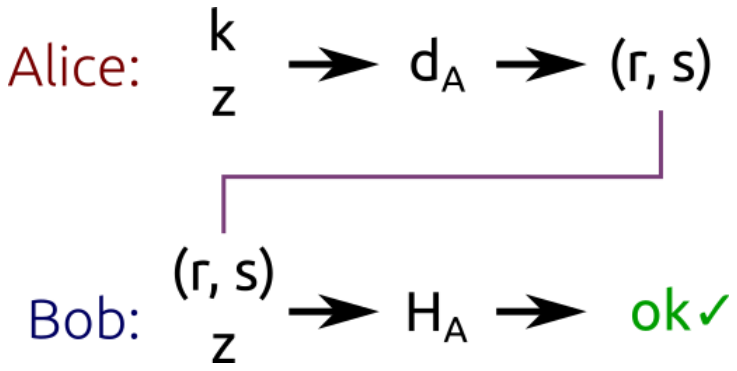

图 8

算法的流程是

1. Alice随机选择一个随机数k (k < n),作为本次加密的临时私钥，并对需要签

名的信息作hash处理得到z。计算P = kG并将$r = x_P mod n$ 作为临时公钥，若r = 0则重新选择。

2. 计算$s = k^{-1}(z + rd_A) mod n$，$d_A$是Alice使用的私钥，$k^{-1}$是k关于n的模逆，如果s为0则重新选择k再次计算。将(r,s)作为签名信息

3. Bob此时有了Alice的公钥H_A和签名信息(r,s)，首先计算

$$u_1 = s^{-1} z mod n$$

$$u_2 = s^{-1} r mod n$$

4. 然后在计算

$$P = u_1 G + u_2 H_A$$

如果$r = x_P mod n$，则可以判定这个信息是来自于Alice的

（3）算法的正确性证明

从验证的步骤往前回溯有

$$\begin{aligned} P &= u_1 G + u_2 H_A \\ &= u_1 G + u_2 d_A G \\ &= (u_1 + u_2 d_A) G \end{aligned}$$

将u_1和u_2的等式带入得到

$$\begin{aligned} P &= (u_1 + u_2 d_A) G \\ &= (s^{-1} z + s^{-1} r d_A) G \\ &= s^{-1}(z + r d_A) G \end{aligned}$$

因为对于基点G来说它的阶为n,因此为了简洁，可以省去mod n

由模逆运算的性质可以将$s = k^{-1}(z + rd_A) mod n$ 改写为$k = s^{-1}(z + rd_A) mod n$ 于是有

$$\begin{aligned} P &= s^{-1}(z + r d_A) G \\ &= kG \end{aligned}$$

可以看出此式正是通过随机数k生成点P的运算，回溯到了最开始的计算，可以验证算法的正确性

（4）算法的漏洞

在Console Hacking 2010大会上，黑客披露了一个Sony PS3的关于ECSDA的一个破解方式。造成这个漏洞的原因在于PS3 在ECSDA算法中未随机的选择数字k用来生成点P,从而造成私钥泄露。

在此处键入公式。

假设我们有两笔使用相同k值签名的信息，这两笔信息的hash为(z_1, z_2)，他们的签名信息为(r_1, s_1)和(r_2, s_2)

- 首先他们的$r_1 = r_2$，因为k值相同
- 考虑$(s_1 - s_2)modn = k^{-1}(z_1 - z_2)modn$
- 将两边同时乘以k，$k(s_1 - s_2)modn = (z_1 - z_2)modn$
- 两边同时除以$(s_1 - s_2)$得到$k = (z_1 - z_2)(s_1 - s_2)^{-1}modn$

上面的最后一个式子我们可以计算得到k，从这个确定的k可以计算得出私钥

$$s = k^{-1}(z + rd_S)modn \Rightarrow d_S = r^{-1}(sk - z)modn$$

从上述分析可以看出相同k是可以导致私钥泄露的

所以随机数k在ECSDA算法中扮有重要角色，如果是在比特币的交易中如果出现私钥泄露那么可能会导致你的财产被盗取，这是很严重的安全问题。