# 椭圆曲线算法介绍

椭圆曲线加密算法是基于椭圆曲线上的离散对数难解问题的非对称密码算法，该算法利用有限域上椭圆曲线的点构成的Abel群离散对数难解性，实现加密、解密和数字签名。将椭圆曲线中的加法运算与离散对数中的模乘运算相对应，就可以建立基于椭圆曲线的对应密码体制。目前常用的椭圆曲线包括secp384r1和secp521r1等。我国基于椭圆曲线制定了SM2安全算法，其中包括加密解密算法以及数字签名算法。

传统的RSA算法，为保证算法安全性，需要一再增加其密钥的长度，这要求设备需要有更大的存储空间来存储密钥，密钥长度的增加也使得算法的计算开销越来越大。而椭圆曲线加密体制在同等安全性下，所需的密钥长度更小，一般认为163bit的椭圆曲线算法即可与密钥长度为1024位的RSA算法达到条相同的加密安全强度，因此椭圆曲线加密算法在存储空间较小的嵌入式环境下更有优势。此外，椭圆曲线加密算法相对于RSA算法，其具有更好的扩展空间，加密效率也更高，带宽的消耗也更低。RSA算法数学原理简单，在工程中容易实现，但是其单位安全强度相对较低。用一般数域筛（NFS）方法去破译和攻击RSA算法，其破解难度是亚指数级的，而椭圆曲线算法相对而言，数学原理更加复杂，在工程中实现的难度较高，但于此同时，其单位安全强度高于RSA算法，使用Pollard rho方法去攻击椭圆曲线加密算法，其攻击的难度为指数级。

# 椭圆曲线加密算法基本原理

## 椭圆曲线

一般而言，椭圆曲线的曲线方程是满足形式的三次方程。其中参数a,b,c,d,e是满足一定条件的实数，定义中还包括一个称为无穷远点的的元素O。说是椭圆曲线，但是实际上该曲线的图形并非是椭圆，而只是方程的形式类似椭圆方程。椭圆曲线是关于x轴对称的。

## 有限域上的椭圆曲线

椭圆曲线是一条连续的曲线，并不适合用于数据加密领，因此，在实际的应用中，我们需要将椭圆曲线变成离散的点，出于此目的，一般将椭圆曲线定义在有限域上。有限域上的椭圆曲线所有的系数都是某一个有限域中的元素（其中p是一个大素数）。在密码学中，最常用的椭圆曲线是由如下方程定义的曲线：

当时，方程有重根，设为,则有点是方程的重根。，在点，曲线的切线无定义，因此在点的倍点运算无意义。

与椭圆曲线公钥密码学有关的有限域有三类，,和。

### 椭圆曲线的加法

椭圆曲线上，对于,其加法定义为：

若有,则

1. 如果,那么有,即是P的加法逆元，表示为。

已知椭圆曲线上的两个不同的点和，且有,令点,则的计算表达式如下所示：

其中为经过P和Q两点的且相交与椭圆曲线上点的直线的斜率，而该斜率的计算方式为：

### 椭圆曲线的乘法

椭圆曲线上，点的乘法被定义为重复的加法，如。

# 对椭圆曲线算法的优化

## 对椭圆曲线加密算法的时间攻击

时间攻击（Timming Attack）是一种攻击者试图通过私有密钥计算时间的变化量来猜测私有密钥的攻击方式。时间攻击无需专用设备，但是却可以很有效的获取用户的私有密钥。例如，攻击者可以通过简单的测量智能卡对用户输入的反应时间和记录用户的输入来破解只能卡的密钥。攻击者通过演唱被动窃听时间就可能破译私有密钥，从而获取卡上的重要信息。时间攻击同其他侧信道攻击一样，依赖于特定的密码系统的实现。不同的实现方式，时间攻击方式可能并不会成功。

在算法层面，椭圆曲线上点的加法和乘法所需用时不同，且因为每次进行操作时，加法和乘法的操作可能会产生不同，为时间攻击创造了可能性。在椭圆曲线密码体系中，攻击者能够通过乘法和求逆元的时间来推测出密钥的位串，这两个运算是椭圆曲线点的加法和倍加操作中耗费时间的。而通过标量乘法的移位-加方法进行计算时，很依赖于密钥位串，因此使用这种乘法算法的椭圆曲线密码体制很容易受到时间攻击。

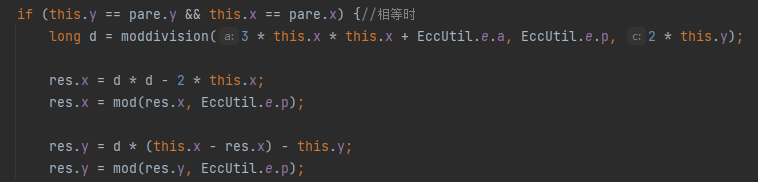
## 椭圆曲线上加法的定义

加法是椭圆曲线的基础，其他的所有运算都是基于椭圆曲线上点的加法实现的。有限域上，点的加法依据参与运算的两个点的不同而不同。

若参与运算的两个点和为相同点，则切线斜率定义为：

long d = moddivision(3 \* this.x \* this.x + EccUtil.e.a, EccUtil.e.p, 2 \* this.y);

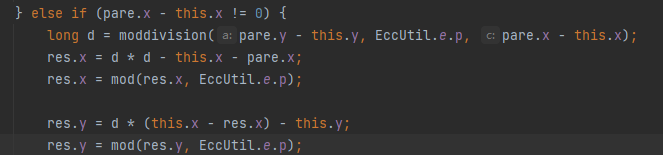
此时定义,则点的坐标为：



而如果两个点P和Q为不同点，且P与Q点的横坐标并不相等，即P和Q两点并非同一点，且P和Q两点并不互逆，则有过P，Q两点并交椭圆曲线于点的直线的斜率为：

long d = moddivision(pare.y - this.y, EccUtil.e.p, pare.x - this.x);

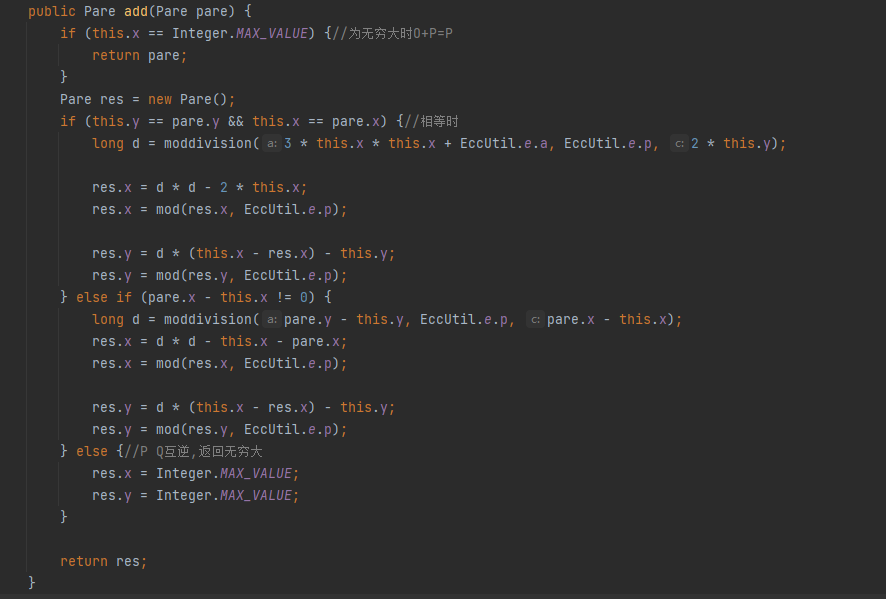
此时，P和Q两点的和，其坐标计算如下：



当P和Q点互逆，即两点横坐标相等，而纵坐标为相反数，即两点关于轴对称时，此时应返回曲线上定义的，一个理论上的无穷远点O。而在计算机的表示上，则可以将无穷点的横纵坐标都定义为无穷大。即当P和Q点互逆时，定义两点和的横纵坐标定义为整数类型可以表示的最大值。

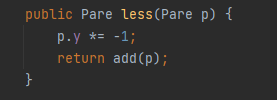
res.x = Integer.MAX\_VALUE;  
res.y = Integer.MAX\_VALUE;

则有对于P和Q两点，使用如下方法进行表示：



## 椭圆曲线上减法的定义

椭圆曲线上，点的减法通过加上点的逆元来实现。即将被减数的纵坐标取导数，然后再进行



## 对乘法的优化

### 二进制乘法

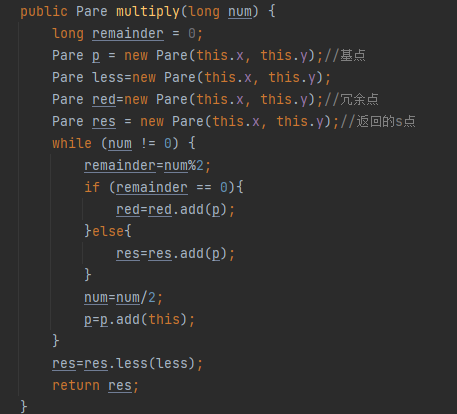
在椭圆曲线中，点的乘法运算是基于点的加法运算。对于椭圆曲线上点的乘法，可以使用二进制算法进行直接计算，其原理较为简单，直观。对于的计算，该二进制直接计算算法分为从左往右和从右往左两种方向，按照乘数的二进制位进行乘法。对于从左往右的加法，对乘数k从高位到低位进行展开，算法的描述为：

输入 ：正整数 k=(kl\_1，…，k1，ko)2，P ∈E为椭圆曲线群上的点   
输出：kP  
 (1)Q←0；  
 (2)对于i从l一1到 0，重复执行  
 1)Q←2Q；  
 2)若 k =1，则 Q← Q+P；  
 (3)返 回(Q)。

对于此算法，用m表示乘数k的二进制长度，D表示点乘，A表示点加运算，则该算法的期望时间复杂度为：

同时，为了抵抗时间攻击，采用一个冗余的点，无论乘数的第i位是0还是1，都进行点加操作。因此该种算法的期望运行时间总计为：

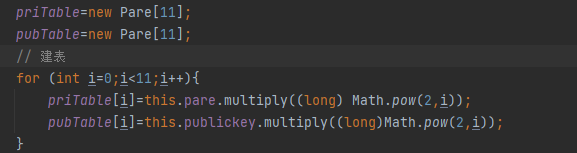
而对于从右到左的直接算法，则从乘数的低位到高位逐位进行计算。算法实现如下：



### 基于造表的乘法

可以通过空间换时间的方法，事先将公钥和私钥的点倍运算计算出结果，并将计算结果存储下来，后续再进行公钥和私钥的点倍计算时，则需要从相应的表中查询对应位置的值进行加法即可，通过这种方式，可以减少在实际的运算过程中，由于点倍运算的时间开销。

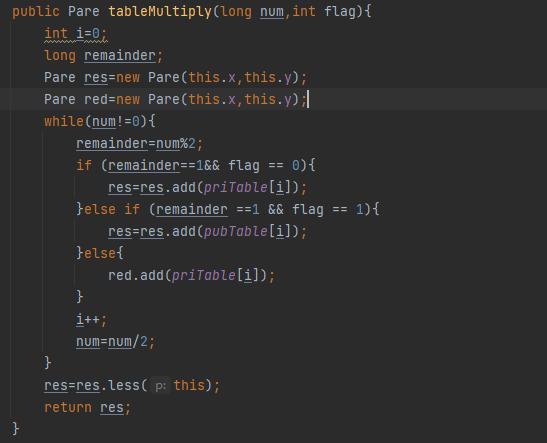
首先需要在初始化加密算法时进行造表。一般而言，以密钥取值为0-1024范围内的椭圆曲线加密算法为例，只需要存储10个点即可保存公钥的点乘造表。造表的实现方式如下：



造表完成之后，则需要利用已经造好的表进行点的乘法。对于,在椭圆曲线进行初始化时进行造表，得到关于公钥和私钥的表Table，基于造表的乘法如下算法所示：

输入 ：正整数 k=(kl\_1，…，k1，ko)2，P ∈E为椭圆曲线群上的点   
输出：kP  
 (1)Q←0；  
 (2)对于i从l一1到 0，重复执行  
 1)若K=0，则不加Table[i]；  
 2)若k =1，则 Q← Q+Table[i]；  
 (3)返回(Q)。

算法的实现方式如下所示：使用标志变量flag表示是公钥还是私钥的乘法，通过flag标志量可以减少代码量，提高函数的使用范围。



通过造表的方式，在实际计算点乘的时候，只需要进行加法即可，这使得计算时间与用户输入不同，从而在一定层面可以避免因为密钥位数不同而造成的计算时间差异。考虑到需要从算法层面考虑到对时间攻击的防御，定义该椭圆曲线的密钥为m位，每次点加用时为A，因此该简单的基于造表的快速点乘方法其时间的期望值为：

而存储造表所需的空间复杂度为：

### 基于窗口的算法

基于二进制乘法的造表方法一次只能通过一个比特位进行比较，这导致密钥有多少个比特位，就需要计算多少次，这毫无疑问的会使得椭圆曲线的计算时间延长。

基于此，可以通过一次进行多位的比较来减少需要比较的次数，从而达到加快点的标量乘法的速度。但是相应的，一次比较多位，这也导致需要更大的存储空间来进行表的存储。假设密钥的长度为n位，而一次进行m位的比较，每次加法用时为A，则基于窗口的造表标量乘法所需时间的期望值为：

于此同时，椭圆曲线密钥体制所需的存储空间约为：

即通过使用窗口，可以在一定范围，通过增加所需的存储空间来较快提升运算的速率。

此外，窗口的值m除了为固定的值长度以外，还可以使用变长的滑动窗口，更加灵活，同时也可以减少运算的次数。

## 不同乘法的比较

基于二进制的交替使用倍点和点加算法得到标量乘积kP的方法是标量乘法的基础算法，其要执行的倍点运算和点加运算与密钥的位数成线性关系。在不考虑时间均衡的前提下，密钥的汉明重量越小，则计算标量乘法的时间越少。

而实现事先造表的方法实现快速点乘的方法，可以以增加所需的存储空间为代价，提高实现点乘的速度。还可以通过一次进行多位的比较来进一步加快标量乘法，但是值得注意的是，该种方法以指数级的空间消耗为代价，换取乘法运算时间的减少，因此每次比较的值并不适合取较大值。