目 录

[1. 椭圆曲线概述 1](#_Toc42331988)

[2. 群论 3](#_Toc42331989)

[2.1 密码学与有限循环群 3](#_Toc42331990)

[3. 椭圆曲线群定义 4](#_Toc42331991)

[4. 椭圆曲线有限循环群 6](#_Toc42331992)

[5. 椭圆曲线代数加法 9](#_Toc42331993)

[5.1 数乘和循环子群 10](#_Toc42331994)

[5.2 子群的阶 11](#_Toc42331995)

[5.3 找基点 11](#_Toc42331996)

[6. 离散对数问题 13](#_Toc42331997)

[7. 椭圆曲线密钥交换 14](#_Toc42331998)

[8. ECDSA 17](#_Toc42331999)

[8.1 签名 17](#_Toc42332000)

[8.2 验证签名 18](#_Toc42332001)

[8.3 算法的正确性 18](#_Toc42332002)

[8.4 k的重要性 19](#_Toc42332003)

[9. 随机曲线 20](#_Toc42332004)

[10. 聚焦secp256k1椭圆曲线 21](#_Toc42332005)

[10.1 参数解释 21](#_Toc42332006)

[11. FAQ 22](#_Toc42332007)

[12. 其他 23](#_Toc42332008)

[12.1 片段 23](#_Toc42332009)

[12.2 参考资料和工具 23](#_Toc42332010)

# 椭圆曲线概述

[椭圆曲线定义](https://mathworld.wolfram.com/EllipticCurve.html)是这样的一个二元三次方程

但是这个表达太繁琐了，光看到这种表达式就已经让人很绝望了。好在有数学家发现，大部分的椭圆曲线可以简化成下面这种形式

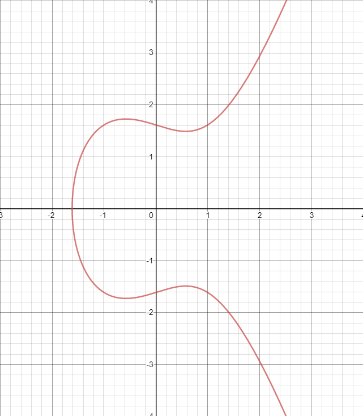
其中a和b系数满足条件，添加这个限制条件是为了保证椭圆曲线上没有非奇异点，进而保证曲线是处处可导的。

这种形式的简化方程叫做Weierstrass Normal Form（WNF）方程，是大数学家Weierstrass（维尔斯特拉斯）的发明。为了脑细胞死得少，这里我们就研究WNF方程式的椭圆曲线。

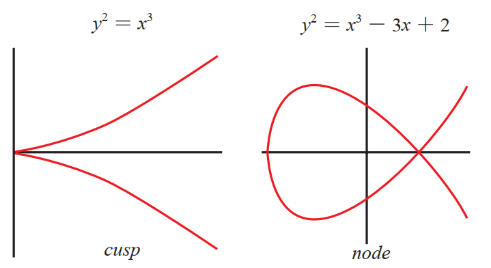
那椭圆曲线长什么样呢？有一个网站可以让我们设置a和b的值看到椭圆曲线的样子，网址：

<https://www.desmos.com/calculator/ialhd71we3>

不同的a和b的值导致绘制出来的椭圆曲线形状是不同的，例如比特币采用的椭圆曲线secp256k1（方程式里a=0，b=7）就长下面这个样子



可以看到该曲线是**光滑处处可导**的曲线，并且是**沿X轴对称**的。下面看看两个反面教材



大家可以看到，左边这条曲线有一个尖锐的点，称作尖点 (cusp)。顾名思义，这条曲线就好比是曲线上捏出一个尖点。而右边的曲线上有一个自交点。在这个交点附近看曲线类似于一个十字架，因此我们称之为结点 (node)，就像是把曲线打了一个结。这两种曲线都不能用来做加密，不在我们研究的范畴。

简化版的椭圆曲线个重要的特性：**沿X轴对称**。这是个重要的特性，后面内容会提到。

从上面的三条椭圆曲线来看，好像椭圆曲线和椭圆一点都不像，那这种曲线的名字里为什么会有椭圆二字？

原来， 当初人们想用微积分计算椭圆的周长（ 圆的周长大家都会求）。 通过一定的积分技巧， 最终要求出以下类型的积分 ：

其中分母的函数项两边平方一下就得到了椭圆曲线方程。

有了这种感性的认识以后，我们先暂停一下椭圆曲线的研究，先补充一下必要的数学知识。

# 群论

## 密码学与有限循环群

现代密码学算法和协议中，消息是作为有限空间中的数字或元素来处理的。加密和解密的各种操作必须在消息之间进行变换，以使变换非常有限消息空间内部的封闭性。然后，数的一般运算注入加减乘除并不满足有限空间内部的封闭性。所以密码算法通常运行于具有某些保持封闭性的代数结构的空间中，这种代数结构就是**有限循环群**。在数学中，群是一种代数结构，由一个集合以及一个二元运算组成。群必须满足四个条件：封闭性，结合律，存在单位元和存在逆元。

最常见的群之一是整数集Z以及加法操作。

**集合**：Z={..., -2,-1,0,1,2,...}

**操作**：加法

**封闭性**：整数 + 整数 = 整数

**结合律**：a+b+c = a+(b+c) = (a+b)+c

**单位元**：对于任意a，有a+0 = a

**逆元**：对于任意a，再存b使a+b=0

有限循环群在群的基础上满足额外的两个条件：群元素个数有限和交换律。循环群由单个元素（这叫产生元，也叫基点）的叠加操作生成，最常见的有限循环群为模拟时钟。



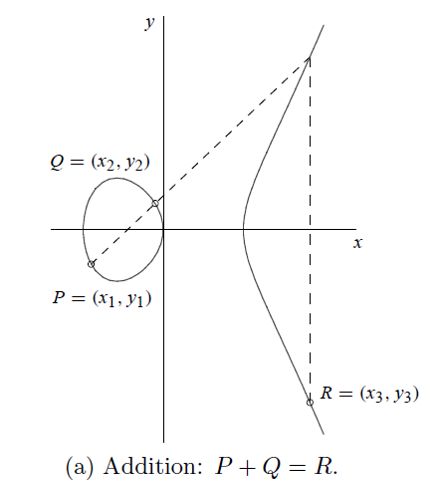
满足交换律的群又叫阿贝尔群。

群的奇妙之处在于如果你能够上述四个性质，那么你可以获得一些其它有趣的性质。

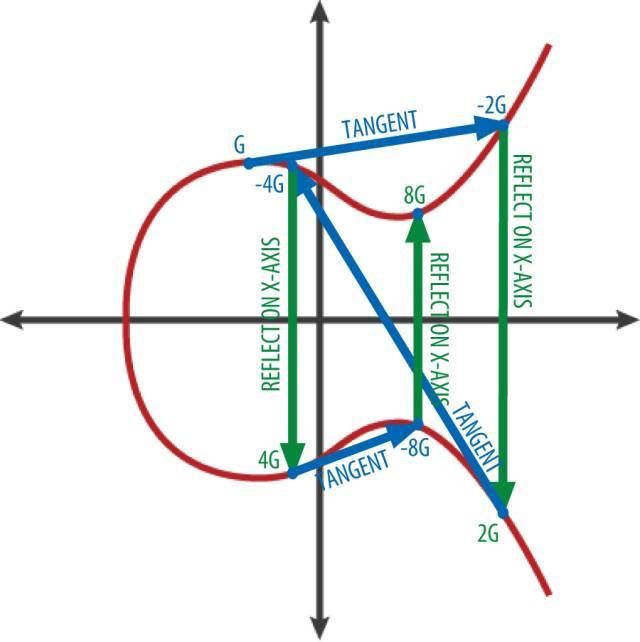
比如，单位元是独一无二的，逆元也是独一无二的。对于任何一个a，有且仅有一个b满足a+b=0。

# 椭圆曲线群定义

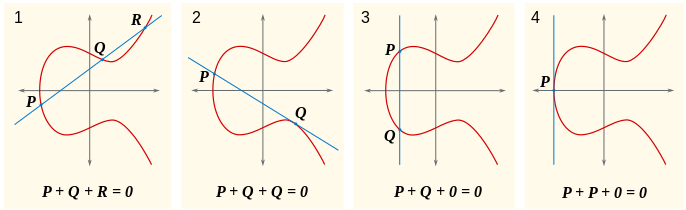
1985年,Neal Koblitz和Victor S.Miller分别独立提出利用椭圆曲线产生椭圆曲线循环群用于密码学。在数学上，椭圆曲线群的元素为椭圆曲线上的点，群操作为“+”，它的几何学定义为：给定曲线两点P和Q，P+Q等于P和Q两点的连线与椭圆曲线交点沿着X轴的对称点，如果P = Q，则P+P等于P在曲线上的切线与曲线交点沿着X轴的对称点。该群的单位元为无穷远点，记作O=(0,0)，有P+O=P，点P的逆元为其沿着X轴的对称点，记作-P。下图是演示P+Q=R的情景。



下图是从基点G出发，不停地做切线，实现2G,4G,8G的过程图，希望你品，你细品。



下面四张图显示了四种“+”操作的情景。



这里我想要说的是，注意看第一张图，P，Q，R三点在同一条直线上。看到这张图，可以理解为P+Q+R = 0，同时也可以理解为P+R+Q=0或者是Q+R+P=0。你品，你细品。因为这里的“+”满足结合律。

注意：密码学中有限域上的椭圆曲线一般有两种，一种是定义在以素数p为模的有限域GF(p)，也就是上面介绍的；另一种则是定义在特征为2的有限域GF(2^m)上，

# 椭圆曲线有限循环群

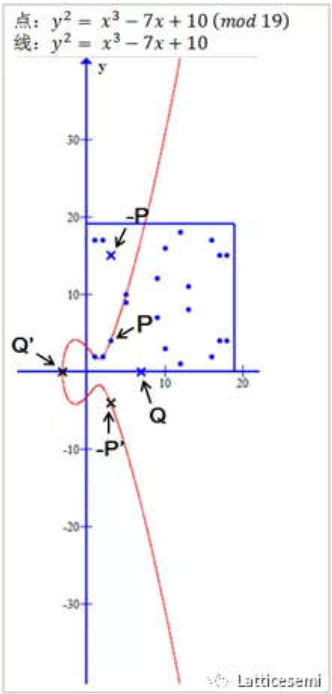
前面介绍的椭圆曲线是基于有理数的，但是计算机运算浮点数的速度慢，更重要的是无法准确表示每一个小数，这将导致加密解密操作后原始消息不能被还原。所以考虑到加密算法的可实现性，密码学上使用**基于整数的模加运算产生椭圆曲线的有限循环群**。

也就是说椭圆曲线有限循环群的公式要由原来的无限域上的方程编程如下形式：

这里不仅添加了mod p操作，还规定了p必须是素数。为什么p必须是素数，后面会讲到。

例如考虑构建下面椭圆曲线的有限循环群

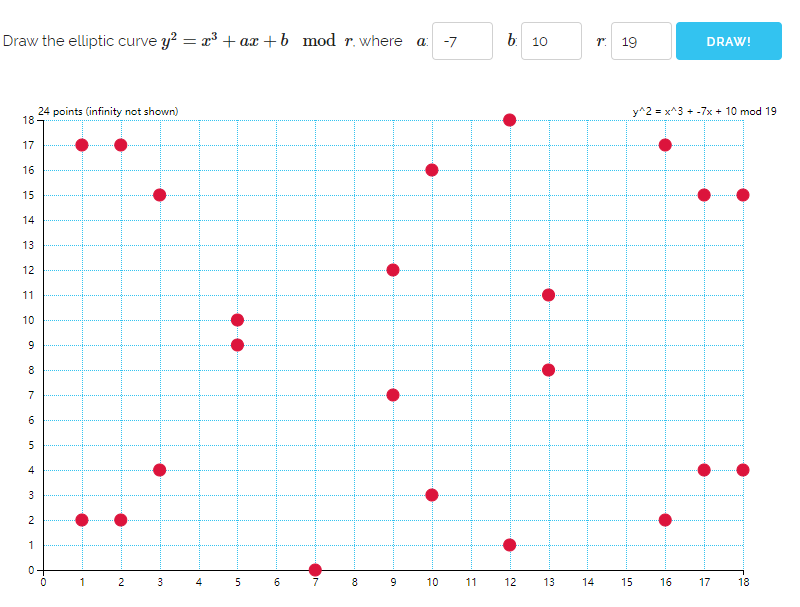
如果要绘制这个有限循环群的图形，我们必须这个有限循环群里的所有元素，即所有合法的坐标点。确认这些点的计算过程是很无聊枯燥的，也没有别的办法，只能暴力尝试。例如这里是mod 19，模运算会把发散的椭圆曲线上的点映射到一个19×19的正方形中。这里要尽量去理解mod运算在几何上的操作，你品，你细品。



这里我们可以使用网上的工具帮助我们绘制椭圆曲线的有限循环群的点，网址如下：

<https://graui.de/code/elliptic2/>

绘制效果如下图所示。

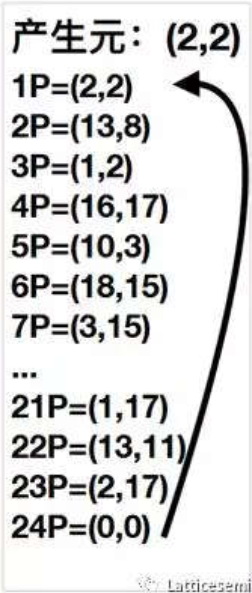


从上图可以看到，正方形里面有很多的“空洞”，这些空洞都是无效的点，只有红色的点才是合法有效的点。并且可以很明显的看出这些点都沿着x = 9.5这条直线对称。当然，随着把19改成更大的素数，有效的点会逐渐增多。

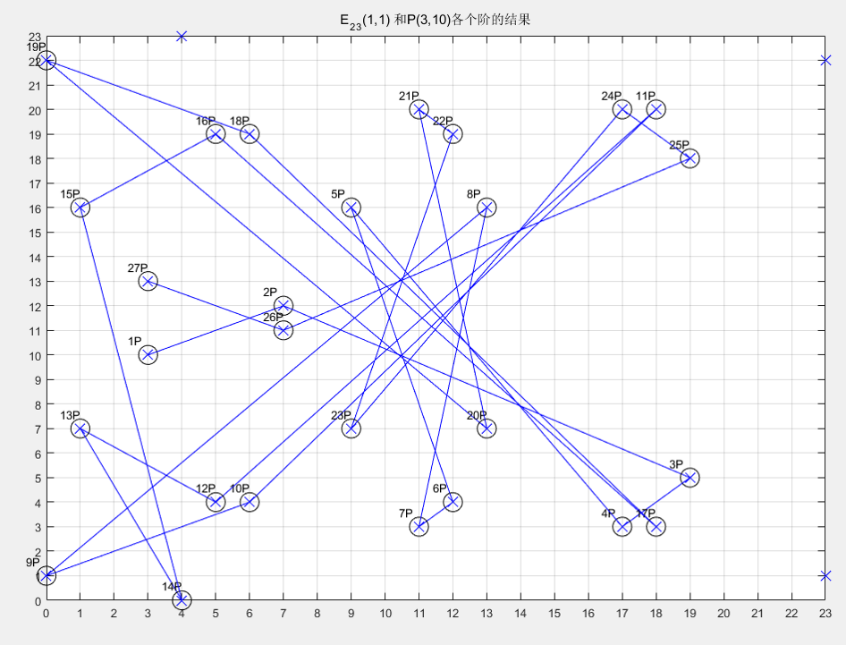
现在，我们基于，利用产生元G(2,2)来生成椭圆曲线有限循环群。

计算1P,2P,3P的过程非常无聊繁琐，网上已经有工具帮助做这种工作：

<http://www.christelbach.com/ECCalculator.aspx>



如果你把产生元生成椭圆曲线有限循环群过程算出来的点按照顺序连起来的话，你会发现像下面这张图。点的生成过程杂乱无章，毫无规律，也就是说，你无法根据当前这个点所在的位置这个点到底是第几个P，哪怕你已经知道了前面若干个点的具体位置。而这正是密码学想要的特性。



这里正好说到椭圆曲线离散对数问题。就是我们这种情形下，已知一个点的具体位置点Q，并且知道产生元位置点P，并且知道点Q和点P存在Q = xP的关系，x是一个正整数。没有快速有效的手段确定x的值。当然，你会说直接查表。查表是在模数很小的情况下可以生效，如果模数非常非常大，如长度为256bit的时候，基本就目前的科技水平来说是无能为力了。据本人的了解，目前由椭圆曲线公钥求解私钥的最有效算法复杂度为O(),其中p是阶数n的最大素数因子。当参数选的足够好让p > 时，以目前的计算能力，攻破椭圆曲线是不现实的。

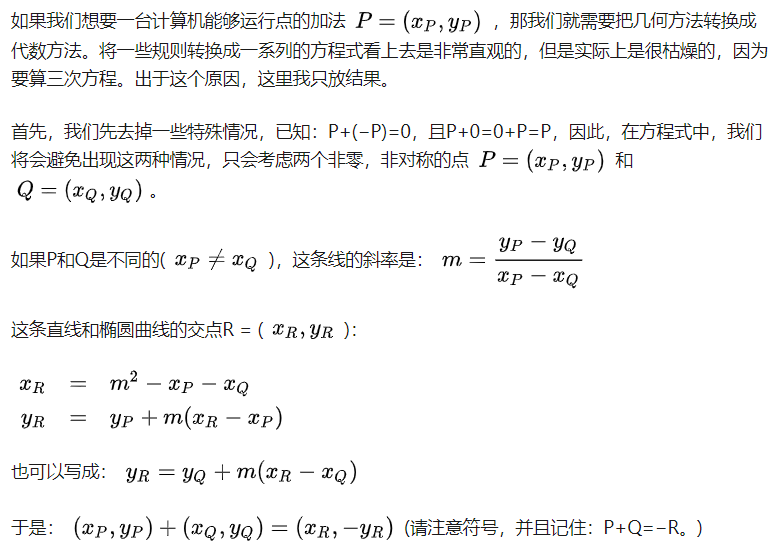
用密码学术语描述为：椭圆曲线有限循环域构成了一个单向函数Q = xP，已知x和P，很容易计算Q，但已知P和Q却很难计算x。**于是（x, Q）就构成了一对公私钥，其中x是私钥，而坐标点Q则是公钥。**

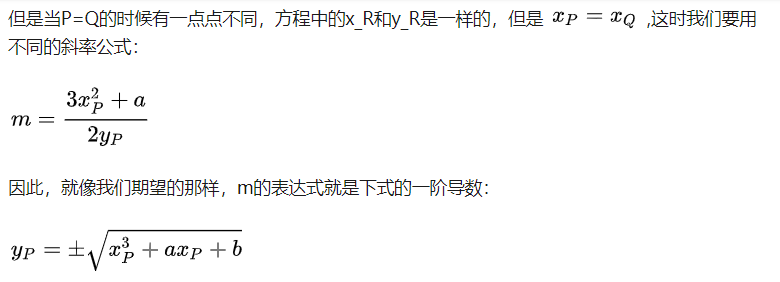
举个例子，利用square and multiply算法，只需要9步就完成Q = 137P的计算。



忘了补充一点，在域中，我们有2种二元运算，加法和乘法，两者运算都是封闭的、满足结合律、分配率。两个运算都是唯一的单位元。并且对于每一个元素，都有唯一的逆元。最后，乘法对加法满足**分配率**：x \*( y + z) = x\*y + x\*z

# 椭圆曲线代数加法



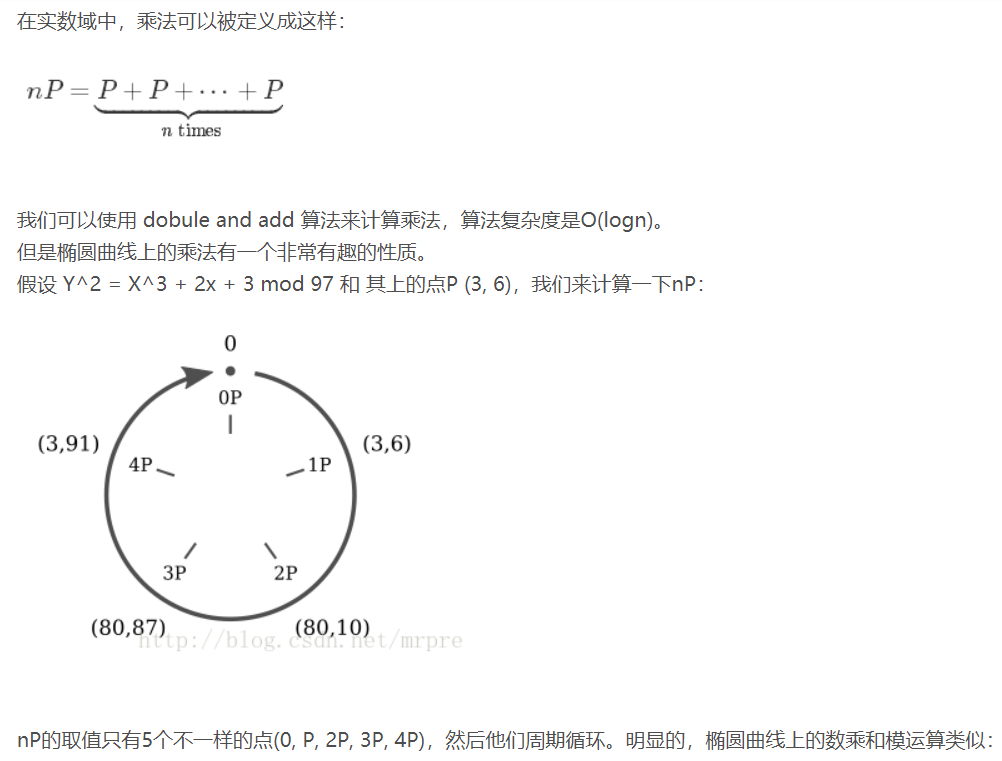


（上面求直线的切线的过程我演算过是没问题的，但是求交点R的坐标公式我没做出来。不知道怎么算。卡在了求一个一元三次方程上）

从这两公式可以看到，要求点相加，只需要进行一些有限域的加减乘除即可。事实上，加减乘除复杂度的关系如下：除法>乘法>加减。为了提高运算效率，往往在运算中把一个点的坐标保留分数的形式如：（）。运算过程中一直保留分数，直到运算结束再做一次除法，回到有限域，这种算法叫做投影坐标。有了投影坐标以后，一次正向计算（私钥求公钥）其实就是一系列的乘法、加法、减法。当然，此处的乘法、加法、减法都是素数域上的运算。为了快速计算素数域乘法(实际上就是模乘)，又诞生了一系列算法：蒙哥马利模乘算法，Barrett模乘算法等，都是通过预计算加快求模速度的算法。

数论中，x/y表示 x\* y^-1，即 x 乘上y的逆元。所以除法运算分为两步：第一步求逆元，第二步计算乘法。乘法逆元可以使用欧几里得拓展算法轻松的计算。

## 数乘和循环子群





循环子群是ECC的基础。

## 子群的阶

我们可以问自己，由P生成的子群的阶是什么？

1：首先，我们已经定义了阶就是群中点的个数。在子群中也是这样的，但是我们可以换一种表达方式：子群的阶是最小能够使得nP=0的n。就像上文中给出的例子，n是5。

2：子群的阶和群的阶是有关系的。拉格朗日定理说明了，子群的阶是群的阶的因子。即如果N是群的阶，则其子群的阶n，则n|N。

上述向我们给出了一个找到子群的阶的方法

（1）计算群的阶N

（2）找出所有N的因子（因式分解）

（3）每个N的因子n，然后乘以P

（4）在3中，找出最小的n，使得满足nP = 0。则这个n是子群的阶。

例如，假设在F37上定义椭圆曲线 y^2 = X^3 - x + 3，显然群的阶可以轻松算出为N = 42，子群的阶是n 可能是 1,2,3,6,7,14,21,42。

设曲线上的点P(2, 3)，由于P 不等于0、2P不等于0……7P等于0，故P的阶是7。即由P生成的子群的阶是7。

注意，“最小的n”是非常重要的。如果随机的遍历0 ~ 42，则很有可能遍历到14,14P也是0，但是14不是P的阶。

另外一个例子，在F29上定义一个椭圆曲线：y^2 = x^3 - x + 1 ，则椭圆曲线的阶为N=37，由于37是素数，所以其因子只有1和37。如果子群的阶为1，则显然，该子群包含一个点，该点就是0.如果子群的阶是37，则该子群就是parent群。

## 找基点

在ECC算法种，我们希望找到一个阶数较大的子群。

通常我们会选择一个椭圆曲线，然后计算它的阶N，选择一个较大的因子n，然后找一个合适的基点。也就是说，我们不是首先找一个基点，然后计算它的阶，而是相反，我们先找到一个合适的阶，然后找以这个数为阶的子群的生成元。

怎么找呢？

首先，拉格朗日揭示，h = N/n是一个整数（当然，n是N的因子），h有一个自己的名字：cofactor of the subgroup。

首先，每个椭圆曲线上的点P，NP = 0，因为N是P的阶n的倍数。

我们可以写成这样 n(hP) = 0。

假设n是一个素数（下篇文章会讲到为什么），我们令G= hP，则G就是子群的生成元。

我们总结一下：

1：计算椭圆曲线的阶N。

2：选择一个数n当成子群的阶。n应该是N的素因数

3：计算h = N/n

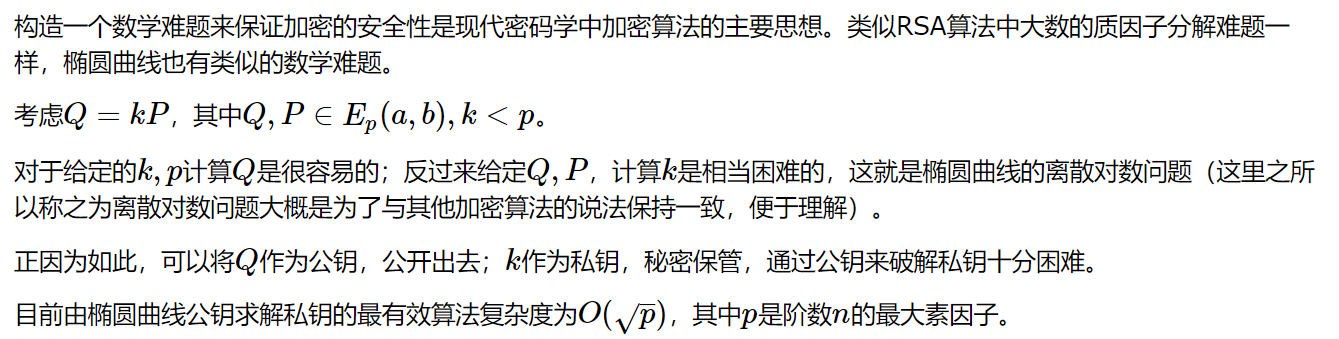
4：随机选择一个点P

5：计算G = hP

6：如果G是0，到第4步。否则，我们找到了这个基点。

n必须是素数，若非如此，则nP = 0不一定表示n是P的阶，因为P的阶可能是n的一个因子。

# 离散对数问题



接下来我们需要考虑一个问题

如果我们知道了P和Q，Q是P的倍数，我们计算这个倍数k？

这个问题就是基于椭圆曲线的离散对数问题，他被认为是很难解的问题。

目前为止没有找到在多项式时间(Polynomial time)内能够解决这个的方法，同样的，这个难题也没有数学上的严格证明。

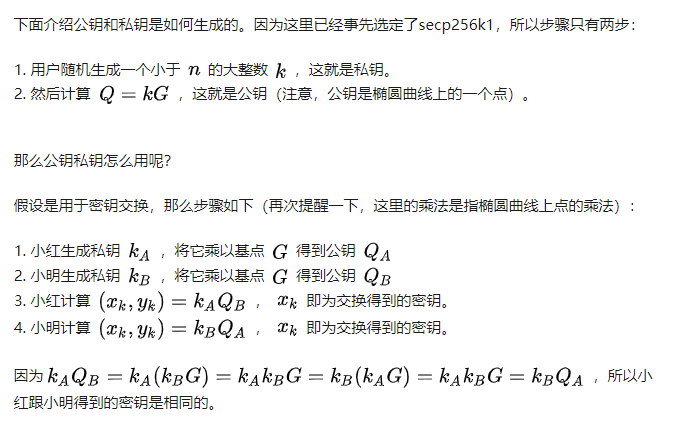
这个难题也和其他密码学中的离散对数问题类似，例如DSA算法，DH密钥交换算法，ElGamal 算法，他们名字基本类似不是没有原因的，因为都是基于离散对数问题。只是上述例子中，使用了模幂运算，而不是我们椭圆曲线的数乘运算。

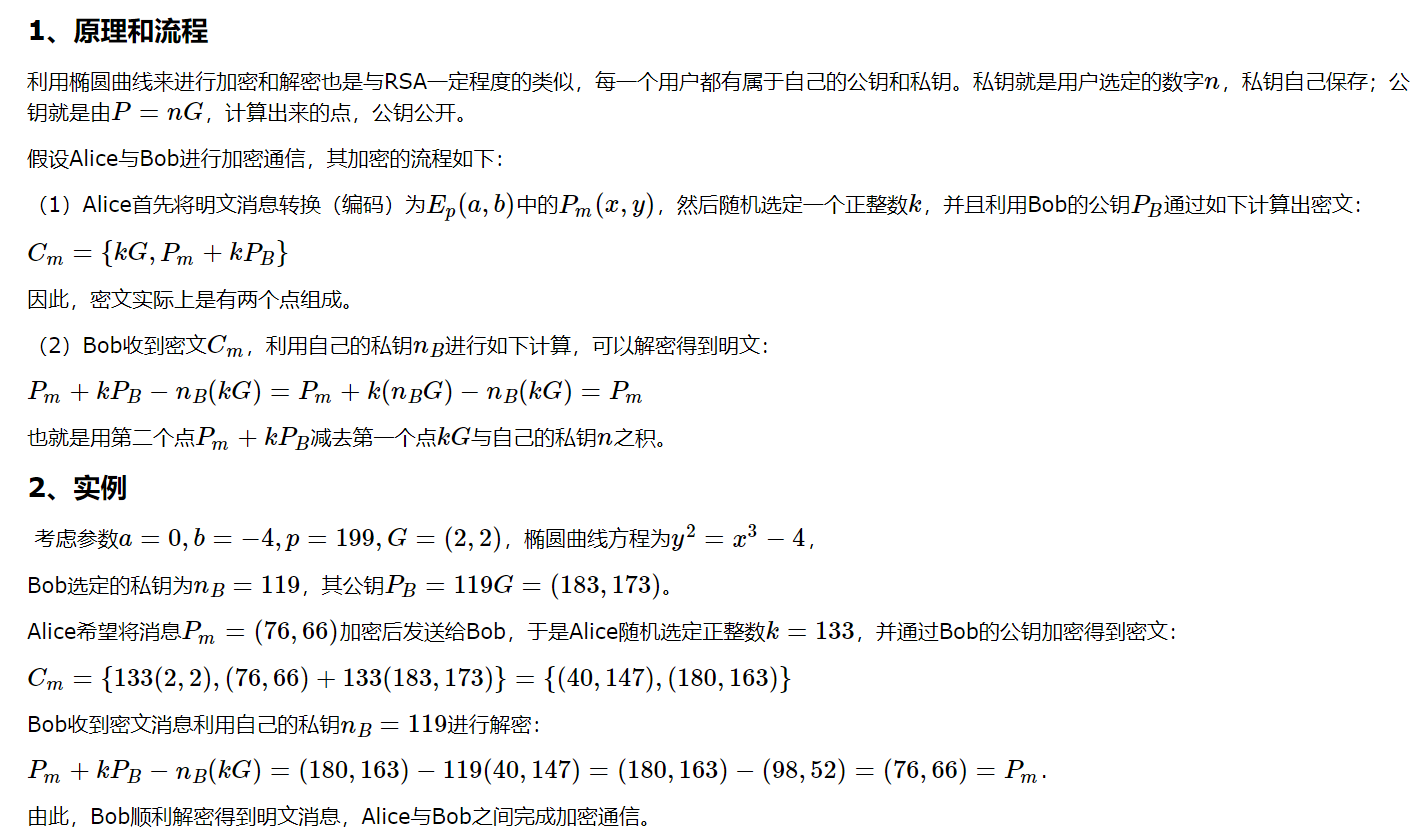
模幂运算的离散对数问题可以这么描述：

我们知道a 和 b是有这么一个等式关系 b = a^k mod p，给你一个a,b,p，让你求k。当然不管是模幂运算的DH还是椭圆曲线数乘的DH，他们的值都是离散的。因为值都取自于有限集合（子群），称之为对数，因为你要计算这个难题，需要对数运算。

ECC有趣的地方就是，它的离散对数问题，看起来比其他的离散对数问题难多了，这也意味着，在椭圆曲线算法中，我们可以使用更小的值k，来达到其他离散对数难题中同样的安全效果。

# 椭圆曲线密钥交换





实际应用中，我们并不需要关心椭圆曲线的众多参数如何选取（要选对参数对于普通使用者来说并不现实），只要从密码学家们精心挑选的一堆曲线中选择一个就行了。一般来说曲线Curve25519，prime256v1是比较常用的，比特币选择secp256k1则是因为它效率较高，并且其参数是可预测的，降低了包含后门的可能性。

ECC保密通信算法

1.Alice选定一条椭圆曲线E，并取椭圆曲线上一点作为基点G 假设选定E29(4,20)，基点G(13,23) , 基点G的阶数n=37

2.Alice选择一个私有密钥k（k<n），并生成公开密钥K=kG 比如25, K= kG = 25G = (14,6）

3.Alice将E和点K、G传给Bob

4.Bob收到信息后，将待传输的明文编码到上的一点M（编码方法略），并产生一个随机整数r（r<n,n为G的阶数） 假设r=6 要加密的信息为3,因为M也要在E29(4,20) 所以M=(3,28)

5.Bob计算点C1=M+rK和C2=rG C1= M+6K= M+6\*25\*G=M+2G=(3,28)+(27,27)=(6,12) C2=6G=(5,7)

6.Bob将C1、C2传给Alice

7.Alice收到信息后，计算C1-kC2，结果就应该是点M C1-kC2 =(6,12)-25C2 =(6,12)-25\*6G =(6,12)-2G =(6,12)-(27,27) =(6,12)+(27,2) =(3,28)

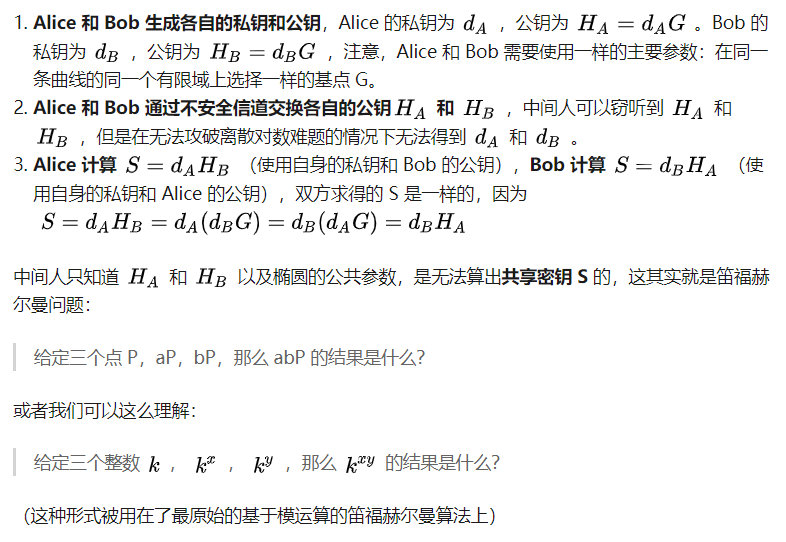
数学原来上能解密是因为:C1-kC2=M+rK-krG=M+rkG-krG-M

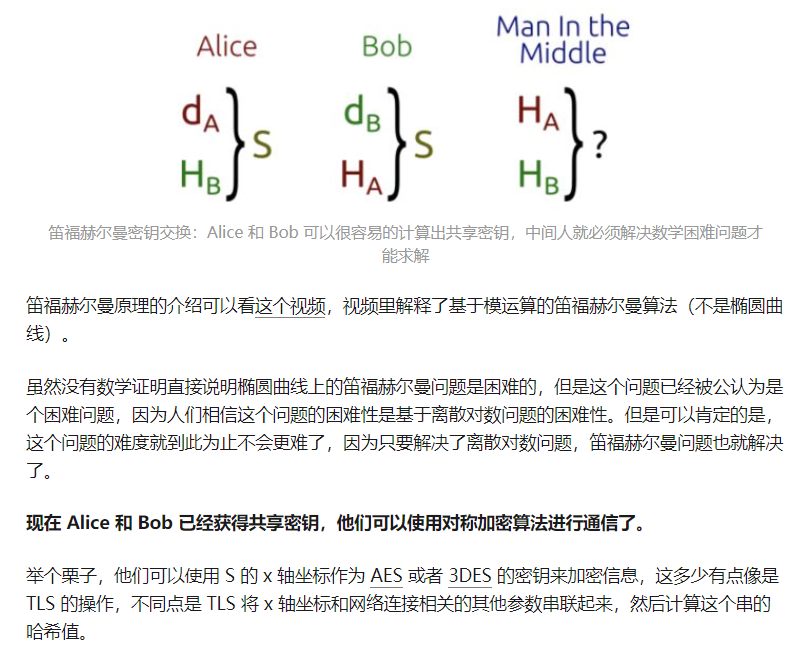
目前Openssl里面的ECC算法的套件支持是ECDSA/ECDH。在国密的SSL套件中，可以使用ECDSA/ECC(密钥加密传输)，ECDSA/ECDH(密钥磋商)两种套件  
  
ECDH

ECDH 是椭圆曲线的笛福赫尔曼算法的变种，它其实不单单是一种加密算法，而是一种密钥协商协议，也就是说 ECDH 定义了（在某种程度上）密钥怎么样在通信双方之间生成和交换，至于使用这些密钥怎么样来进行加密完全取决通信双方。

我们需要解决的问题通常是这样的：Alice 和 Bob 想要安全通信，中间人可能会窃听消息，但是没办法解密消息。

那么 ECDH 是这样的：





ECDHE

你可能听过 ECDH 而没听过 ECDHE，ECDHE 中的 E 代表着「短暂的」，是指交换的密钥是暂时的动态的，而不是固定的静态的。

举个栗子，在 TLS 中就使用了 ECDHE，连接建立时，服务器和客户端都动态生成公私钥，这些密钥在之后会用于 TLS 认证和通信双方之间的信息交换。

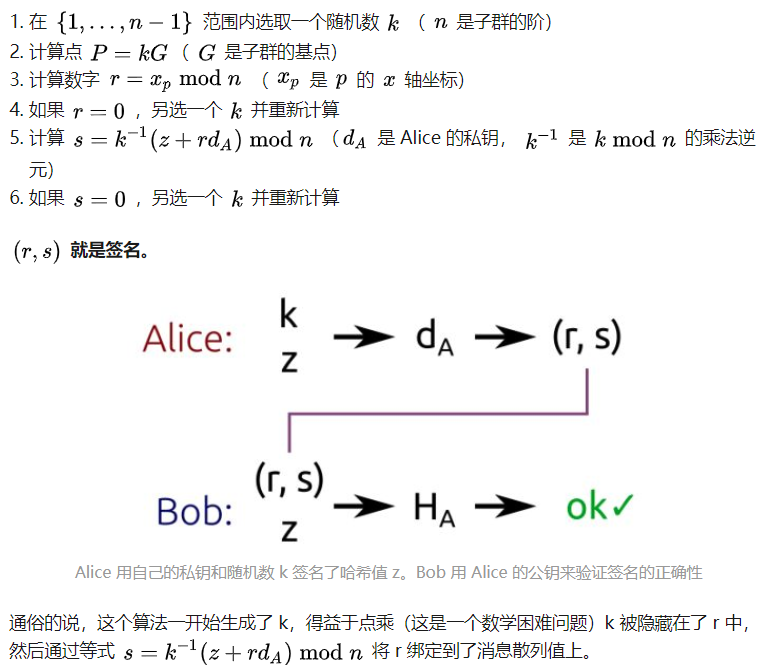
# ECDSA

假设这样一个场景：Alice 想要使用她的私钥 [公式] 来签名，Bob 想用 Alice 的公钥 [公式] 要验证签名，只有 Alice 才能提供正确的签名，而每个人都可以验证签名。

ECDSA 是 DSA 作用于椭圆曲线的一个变种算法。Alice 和 Bob 仍然使用同样的曲线，ECDSA 需要使用明文的哈希结果，而不是明文本身。哈希函数的选择取决于使用者，但是需要明确的是必须选择加密安全的哈希函数，为了使哈希结果的比特长度和 n （子群的阶）的比特长度一致，消息的哈希结果需要被截断，被截断后的哈希值会是一个整数，我们用 Z 来表示。

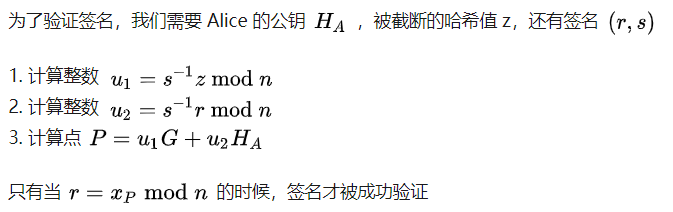
## 签名

Alice 使用算法来签名的步骤如下：



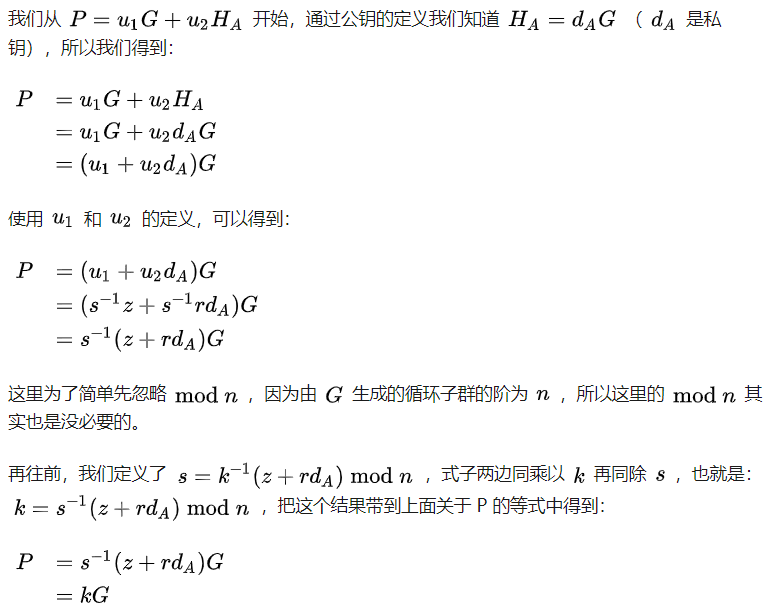
为了计算 s，我们必须计算 k 的逆 mod n，在之前的文章中说过只有在 n 是素数的情况下才能保证这一过程，如果子群的阶不是一个素数，ECDSA 将不起作用。几乎所有标准的曲线都是素数阶的，这肯定不是巧合，非素数阶的那些曲线是不能被 ECDSA 使用的。

## 验证签名



## 算法的正确性

算法的逻辑一开始看不是很容易理解，如果我们把前面用到的公式整合联立一下，就变得清晰了。



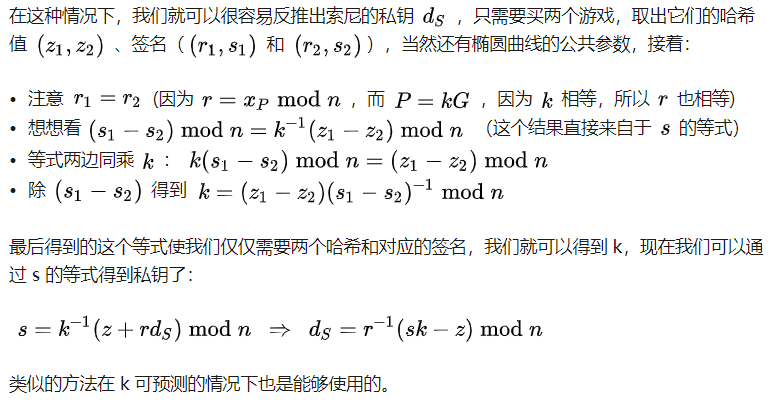
这不就是我们在签名时候的第二个步骤得到的等式吗！在生成签名和验证签名的时候，我们使用了不同的等式计算了同样的点 P，这就是这个算法能够使用的原因。

实践代码：<https://github.com/andreacorbellini/ecc/blob/master/scripts/ecdsa.py>

## k的重要性

在生成 ECDSA 签名的过程中，保证 k 的绝对私密非常重要。如果所有的签名都使用一样的 k，或者使用的随机数生成器不够随机（可预测），那么攻击者就能够找出私钥！

索尼在几年前就犯过这样的错误，正常来说， PlayStation 3 只能运行被索尼的 ECDSA 算法签名过的游戏，如果我想创建一个 PS3 的新游戏，我并不能在没有索尼签名的情况下向市场推广我的游戏。问题来了，索尼在 PS3 中的所有签名都是用固定的 k 生成的。



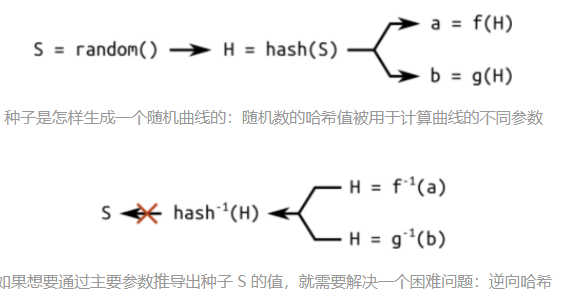
# 随机曲线

随机曲线

「离散对数问题很困难」这种说法其实不完全正确，有一类椭圆曲线特别的弱以至于一些不怀好意的算法可以有效率的求解离散对数问题。例如，具有 p = hn（这意味着有限域的阶等于椭圆曲线的阶） 性质的所有曲线对于 smart 攻击是脆弱的，这就可以被用来在经典计算机上，多项式时间内解决离散对数问题。

现在，假设我给你一个曲线的主要参数，有可能我发现了一种新的没人知道的弱曲线，而且我已经在我给你的曲线上构建了一个快速算法，可以用来求解离散对数问题，我怎么样能让你确认我给你的曲线是安全的（换句话说，它不能被我用来做一些特殊攻击）？

为了解决这个问题，有时候我们需要另一个参数：种子 S，这是一个用来生成参数 a， b 或者基点 G，或者三个参数都生成的随机数，这些参数是通过计算种子 S 的哈希值得到的。哈希值，我们知道的，是正向计算容易，反向计算困难的。



通过种子生成的曲线是可验证随机性的，使用哈希来生成参数的原则是众所周知的「Nothing-up-my-sleeve number」，这个原则也被运用在密码学中。

种子 S 可以提供一种保证，使提供曲线的人不会知道一些特殊的攻击漏洞。如果我将种子 S 和曲线一起提供给你，这就意味着我不会任意地选择参数 a 和 b，你也就可以相对确认我不能够发起一些特数目的的攻击，为什么是「相对」的呢，这个稍后解释。

生成和检查随机曲线的标准算法在 ANSI X9.62 中有描述，这是一个基于 SHA-1 的算法。如果你感兴趣，可以了解用于在 SECG 规范上生成可验证随机曲线的算法（找到 "Verifiably Random Curves and Base Point Generators"）

# 聚焦secp256k1椭圆曲线

## 参数解释

p，a，b ，G，n，h

a和b是椭圆曲线的两个参数，这没什么好说的

p是椭圆曲线所在的有限域Fp。密码学所使用的椭圆曲线都是定义在有限域上的。就叫它模素数p，因为p一定是素数。

G是产生元，或者叫基点。

n是椭圆曲线的子群的阶，表示这个子群的元素数量。同时它是使得nG = O的最小正整数。

h是椭圆曲线群的阶跟由G生成的子群的阶的比值。如果是1表示这个子群就是该椭圆曲线群。

通常将Fp上的一条椭圆曲线描述为T=(p,a,b,G,n,h)p、a、b确定一条椭圆曲线（p为质数，(mod p)运算）G为基点，n为点G的阶，h是椭圆曲线上所有点的个数m与n相除的商的整数部分参量选择要求：

p越大安全性越好，但会导致计算速度变慢

200-bit左右可满足一般安全要求

n应为质数

h≤4；

p≠n×h ；

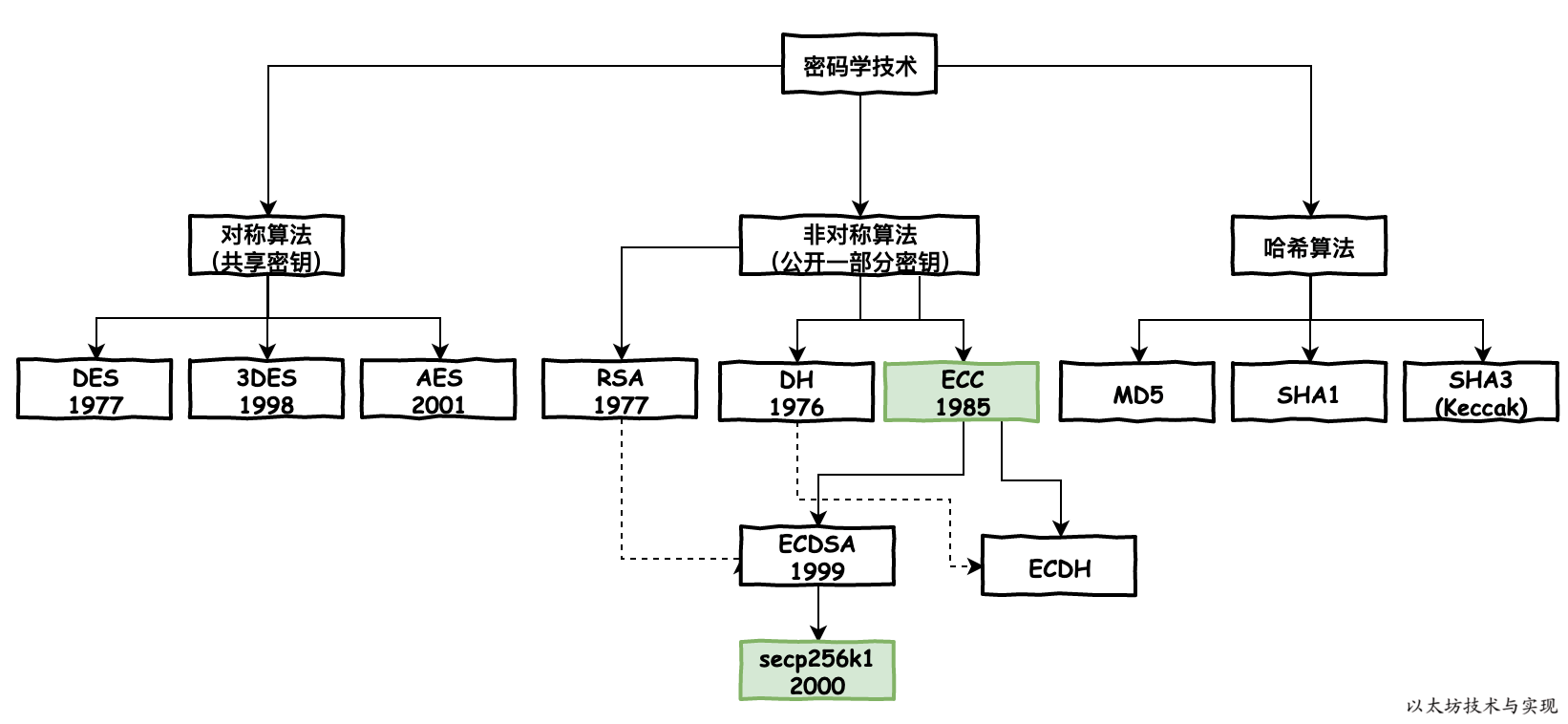
pt≠1(mod n) (1≤t＜20)

4a^3＋27b^2≠0 (mod p)

secp256k1 是[高效密码组标准(SECG)](https://www.secg.org/) 协会开发的一套高效的椭圆曲线签名算法标准。 在比特币流行之前，secp256k1并未真正使用过。secp256k1 命名由几部分组成：sec来自SECG标准，p表示曲线坐标是素数域，256表示素数是256位长，k 表示它是 Koblitz 曲线的变体，1表示它是第一个标准中该类型的曲线。

SECG(Standards for Efficient Cryptography Group) 成立于1998年，一个从事密码标准通用性潜力研究的组织。旨在促进在各种计算平台上采用高效加密和提高互操作性。

但因具有几个不错的特性，现在它越来越受欢迎。大多数常用的椭圆曲线是随机结构，但 secp256k1是为了更有效率的计算而构造了一个非随机结构。因此经过充分地优化算法代码实现，其计算效率可以比其他椭圆曲线算法快30%以上。此外，与常用的NIST曲线不同，secp256k1 的常量是以可预测的方式挑选的，这可以有效降低曲线设计者安置后门的可能性。



从图中看到，secp256k1 是 ECDSA 算法中的一个标准，出现的也比较晚。为何中本聪为比特币secp256k1作为交易验证的签名算法？比特币开发者社区曾讨论过 [secp256k1](https://bitcointalk.org/?topic=2699.0) 是否安全。中本聪没有明确解释，只是说道”有根据的推测”。社区的讨论不外乎是在安全和效率上做权衡，选择一个不受任何政府控制、无后门的签名算法是比特币的首要考虑因素，其次，也需要提供计算速度，毕竟在比特币中加密、签名、校验签名是不断在处理的事情（60%左右的CPU时间几乎全用在这上面），而具有可预测性、高计算效率特性的Koblitz曲线是不错的选择。基于安全第一，效率第二原则，secp256k1 就是一个最优解。

# FAQ

椭圆曲线有限循环群，有限是由什么参数决定的，而循环又是由什么决定的？

答： 有限，是由mod p后面那个素数p决定的，它把有限群里的所有元素都框在了一个p×p的正方形里，这就说明这个群里元素上限就是p×p个。而实际上正方形里有很多非法的点，实际上要用schoof算法计算有限群元素的个数。

而循环是由基点和mod运算决定的。取模就限定了这就是个循环，而基点只不过决定了循环的值是多少。因为一旦选择了一个基点，那么就决定了Q = xP，这样一直算下去，一定有一个整数x会让xP等于O。

椭圆曲线为什么要X轴对称？

由于椭圆曲线有X轴对称这个特性，所以在椭圆曲线群上做“+”的时候，可以很方便的实现逆元操作。

椭圆曲线为什么要光滑可导？

因为椭圆曲线“+”运算过程中需要做椭圆曲线上点的切线。如果不可导，则无法做切线计算，也就无法实现“+”运算。

量子计算机能攻破椭圆曲线加密算法吗？

量子计算机下有攻破椭圆曲线密码的多项式算法(shor算法）

请问下生成以后公钥压缩编码前缀的时候，为什么y的正负决定了y的奇偶呢？

因为 + = 模素数p

而模素数p一定是奇数，所以 和 二者必有一个是奇数，另一个是偶数。所以说y的正负决定了y的奇偶。

有限域上的椭圆曲线的模素数p为什么一定要是素数？

如果p是素数，则模p剩余类环是域。P是素数很重要，如果p不是素数，比如p=4，则集合Zp={0, 1, 2, 3}，显然2没有逆元。

私钥为什么要分压缩和非压缩？

为了区分其对应的公钥是否压缩，进而对应地址是不同的，也就是说压缩私钥和非压缩私钥对应的地址是不同的。

# 其他

## 片段

密码学中使用的通常不是实数域内的曲线，而是有限域内的曲线。最常用的有限域就是素数域GF(p),其中所有的算术运算需要针对素数p执行模运算。

不管是RSA、离散对数加密还是椭圆曲线加密，公钥加密算法都是依赖于某个正向计算很简单（比如多项式时间复杂度），而逆向计算很难（比如指数时间复杂度）的数学难题。对于RSA，这个问题是大整数因子分解问题；对于离散对数加密，是离散对数问题；对于椭圆曲线加密，则为椭圆曲线上的离散对数问题。

## 参考资料和工具

绘制光滑椭圆曲线

<https://www.desmos.com/calculator/ialhd71we3>

绘制有限域椭圆曲线

<https://graui.de/code/elliptic2/>

另一个简单的椭圆曲线绘制工具，有两个点的加法

<https://andrea.corbellini.name/ecc/interactive/reals-add.html>

ECC教程（英文）

<https://www.certicom.com/content/certicom/en/ecc-tutorial.html>

ECC的另一个教程

<https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/>

椭圆曲线加密算法入门

<https://zhuanlan.zhihu.com/p/36326221>

安全曲线

<https://safecurves.cr.yp.to/field.html>

只能计算一些比较小的数据

<http://www.christelbach.com/ECCalculator.aspx>