VICTORLAMP

区块链技术核心篇
之二：比特币区块链密钥与地址

e休，爱编程的葫芦娃：exiu@victorlamp.com

煤油灯科技公司 http://www.victorlamp.com

多媒体课程：《深入浅出区块链技术核心篇》

深入浅出区块链技术

课程介绍

《区块链技术基础篇之一：白话非对称加解密》
《区块链技术基础篇之二：白话哈希算法》
《区块链技术基础篇之三：白话P2P网络》
《区块链技术基础篇之四：白话拜占庭将军问题》
《区块链技术核心篇之一：比特币区块链起源及原理》
《区块链技术核心篇之二：比特币区块链密钥与地址》
《区块链技术核心篇之三：比特币区块链交易共识》
《区块链技术核心篇之四：比特币区块链核心架构》

讲师介绍

网名：一休

2015 年开始，从事区块链技术开发，先后成功的研发出：区块链技术的数字版权管理（DRM）系统、基于区块链IPFS的CDN产品开发。

是多项区块链技术专利发明人。

对于比特币区块链、IPFS项目源码非常熟悉。

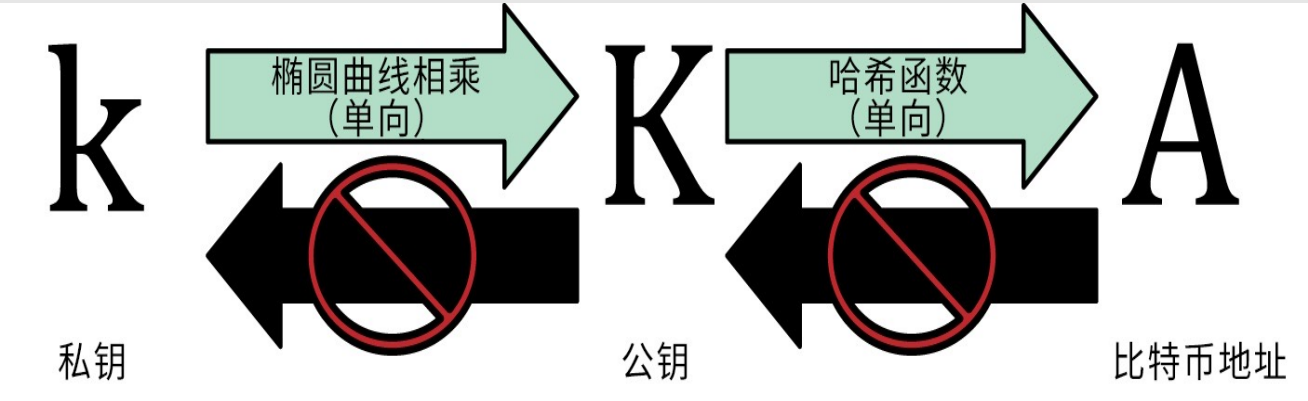
本人负责的基于区块链的创新技术方案，获得了当年华为公司年度十大发明奖。

在《比特币：一种点对点的电子现金系统》一文中，中本聪提到了用椭圆加密算法（ECDSA）来产生比特币的私钥和公钥。

基于椭圆加密的原理，由私钥是可以计算出公钥的，再由公钥经过一系列数字签名运算就会得到比特币钱包地址。

因为由公钥可以算出比特币地址，所以我们经常把公钥和比特币地址的说法相混淆，他们都是指的同一个概念，比特币钱包地址只是另一种格式的公钥，但是两者的外在表现形式是不一样的。

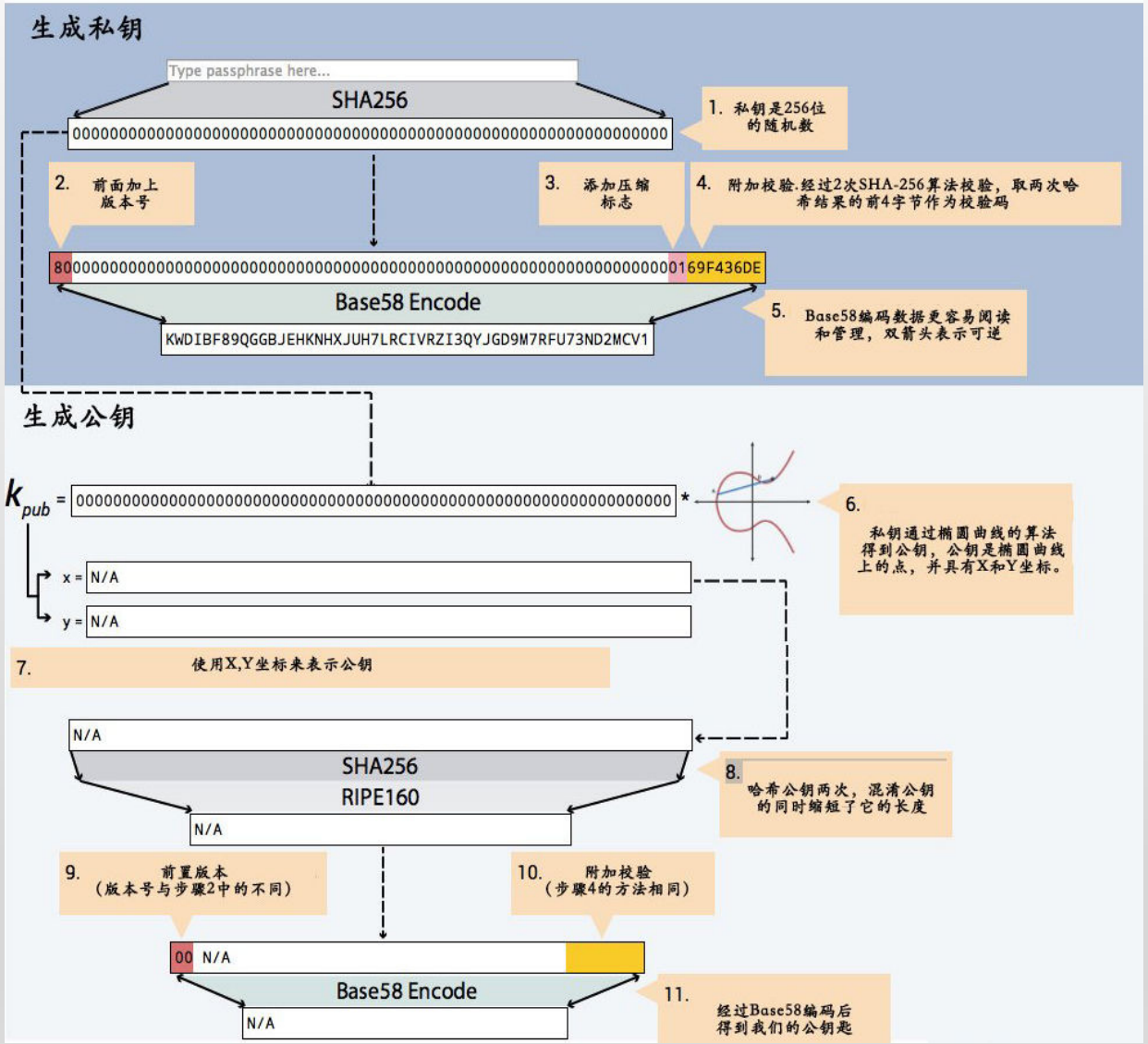
私钥 —— >>公钥 ——>> 比特币钱包地址



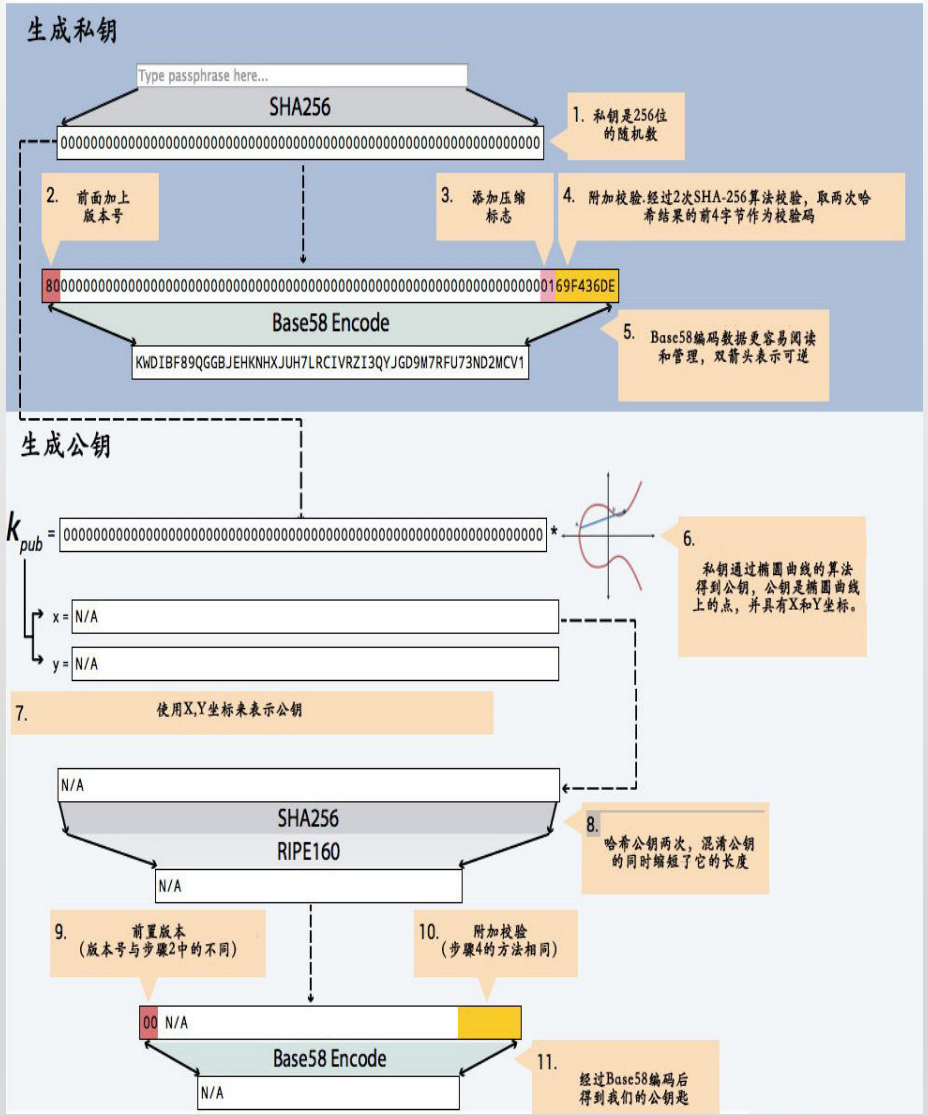
比特币区块链密钥

从比特币私钥得到我们日常转账所用的比特币钱包地址总共需要九个步骤。

中间用到了SHA256加密、RIPEMD160加密和BASE58编码。



比特币区块链私钥生成



第一步：生成随机私钥

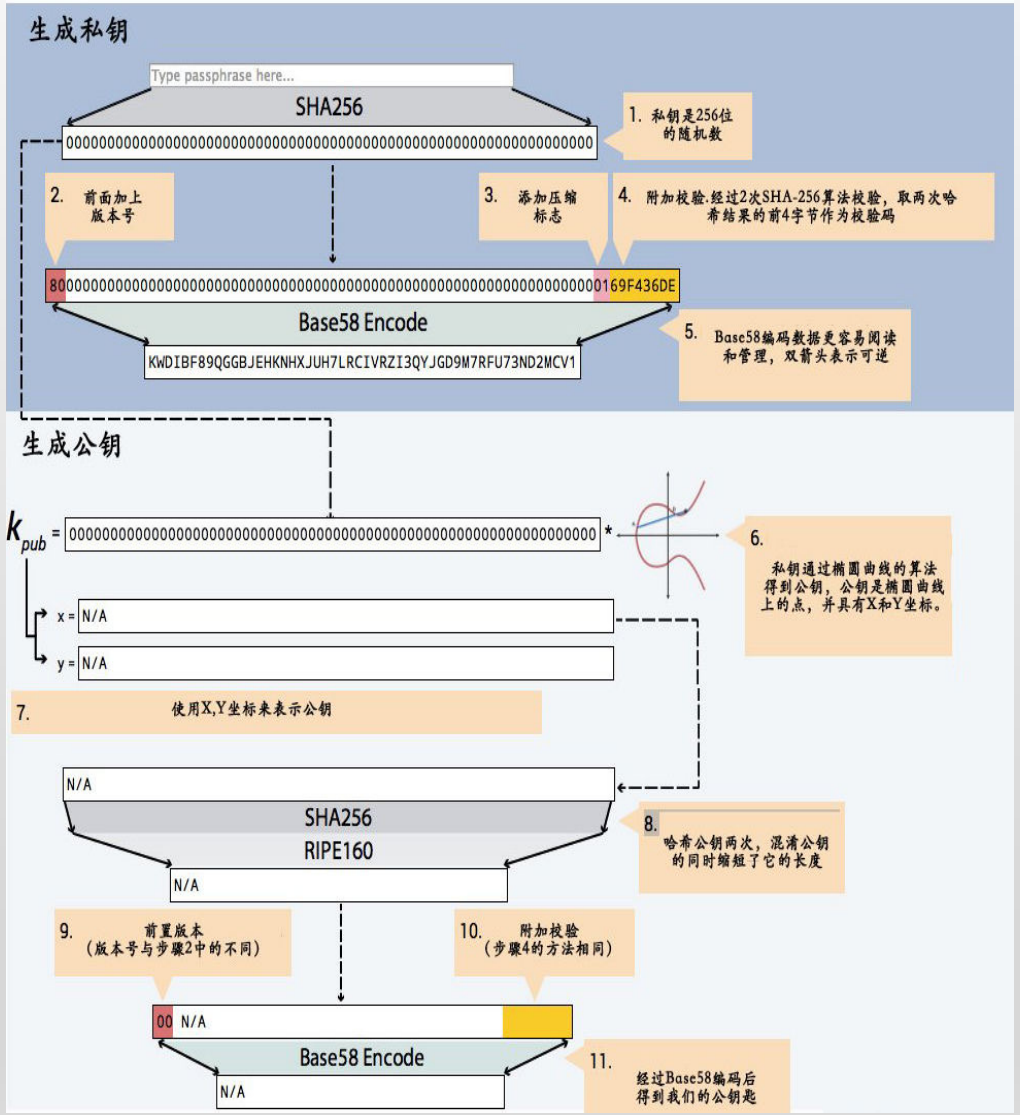
私钥是一个随机数，随机选取一个32字节的数，这个数的范围大小是介于1 ~ 0xFFFF FFFF FFFF FFFF FFFF FFFF FFFE BAAE DCE6 AF48 A03B BFD2 5E8C D036 4141之间的一个数，为了方便后面的计算，我们随机生成一个合法的私钥：
8F72F6B29E6E225A36B68DFE333C7CE5E55D83249D3D2CD6332671FA445C4DD3

私钥就是一个随机选出的数字而已。一个比特币地址中的所有资金的控制取决于相应私钥的所有权和控制权。在比特币交易 中，私钥用于生成支付比特币所必需的签名以证明资金的所有权。私钥必须始终保持机密，因为一旦被泄露给第三方，相当 于该私钥保护之下的比特币也拱手相让了。私钥还必须进行备份，以防意外丢失，因为私钥一旦丢失就难以复原，其所保护 的比特币也将永远丢失。

从一个随机数生成私钥
生成密钥的第一步也是最重要的一步，是要找到足够安全的熵源，即随机性来源。生成一个比特币私钥在本质上与"在1到 2256之间选一个数字"无异。只要选取的结果是不可预测或不可重复的，那么选取数字的具体方法并不重要。比特币软件使用 操作系统底层的随机数生成器来产生256位的熵(随机性)。通常情况下，操作系统随机数生成器由人工的随机源进行初始 化，也可能需要通过几秒钟内不停晃动鼠标等方式进行初始化。对于真正的偏执狂，可以使用掷骰子的方法，并用铅笔和纸 记录。
更准确地说，私钥可以是1和n-1之间的任何数字，其中n是一个常数(n=1.158*1077，略小于2256)，并由比特币所使用的椭圆曲线的阶所定义。

要生成这样的一个私钥，我们随机选择一个256位的数字，并检查它是否小于n-1。从编程的角度来看，一般是通过在一个密码学安全的随机源中取出一长串随机字节，对其使用SHA256哈希算 法进行运算，这样就可以方便地产生一个256位的数字。如果运算结果小于n-1，我们就有了一个合适的私钥。否则，我们就 用另一个随机数再重复一次。

比特币区块链公钥和地址生成



第二步：椭圆曲线算公钥

生成了私钥之后，我们使用椭圆曲线加密算法（ECDSA-secp256k1）计算私钥所对应的非压缩公钥，生成的公钥共65字节， 其中一个字节是0x04，其中32个字节是X坐标，另外32个字节是Y坐标：

公钥P.X：
06CCAE7536386DA2C5ADD428B099C7658814CA837F94FADE365D0EC6B1519385
公钥P.Y：
FF83EC5F2C0C8F016A32134589F7B9E97ACBFEFD2EF12A91FA622B38A1449EEB

第三步：计算公钥的SHA-256哈希值

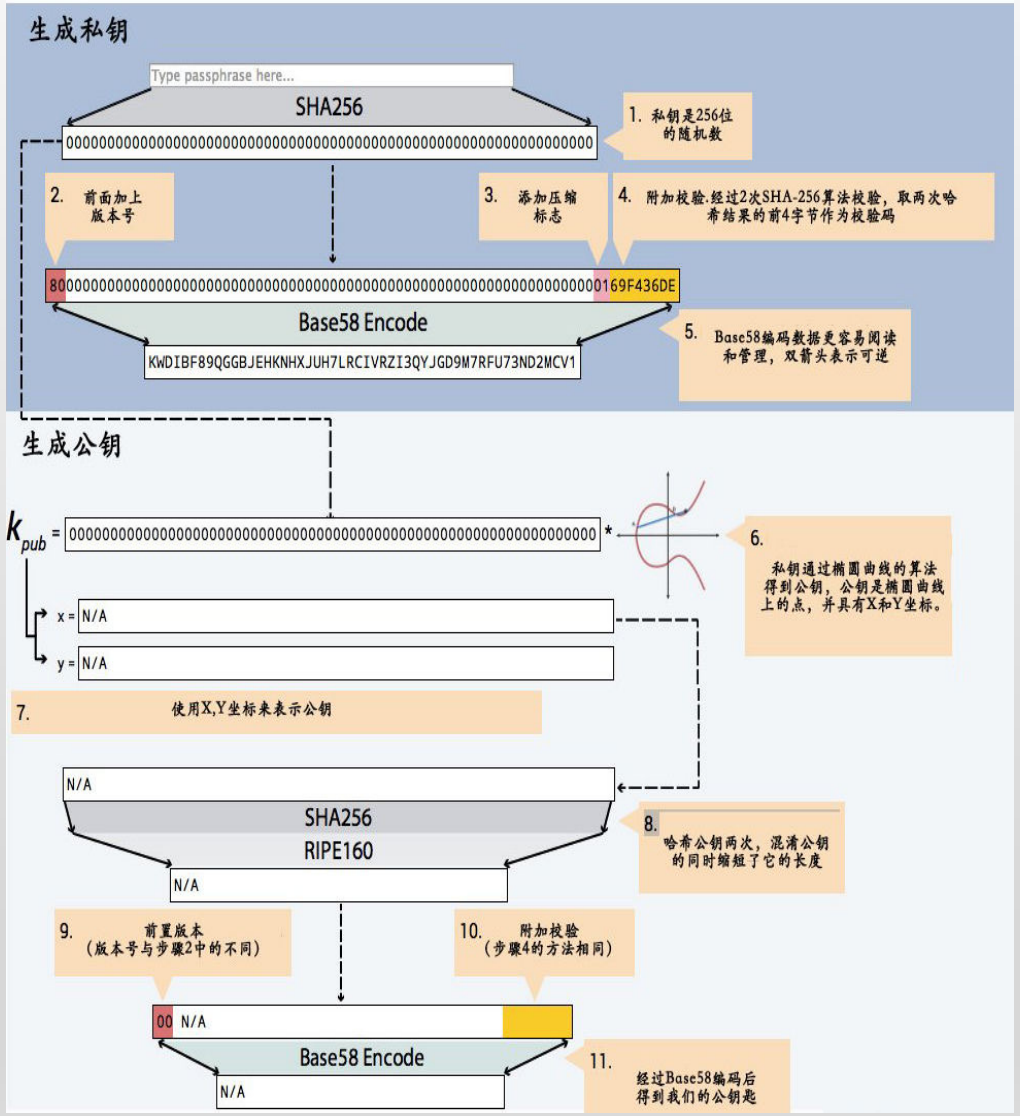
将上述公钥地址拼合，得到标准地址：
0406CCAE7536386DA2C5ADD428B099C7658814CA837F94FADE365D0EC6B1519
385FF83EC5F2C0C8F016A32134589F7B9E97ACBFEFD2EF12A91FA622B38A1449EEB

对齐进行SHA-256哈希计算，得到结果：
2572e5f4a8e77ddf5bb35b9e61c61f66455a4a24bcfd6cb190a8e8ff48fc097d

第四步：计算 RIPEMD-160哈希值

取上一步结果，进行RIPEMD-160计算，得到结果：
0b14f003d63ab31aef5fedde2b504699547dd1f6

比特币区块链公钥和地址生成



第五步：加入地址版本号（比特币主网版本号“0x00”）

取上一步结果，在前面加上16进制的00，即：

000b14f003d63ab31aef5fedde2b504699547dd1f6

第六步：计算 SHA-256 哈希值

取上一步结果，进行SHA-256计算，可得：

ddc2270f93cc84cc6869dd373f3c340bbf5cb9a8f5559297cc9e5d947aab2536

然后，对以上结果再次计算 SHA-256 哈希值，得到：

869ac57b83ccf75ca9da8895823562fffb611e3c297d9c2d4612aeeb32850078

第七步：取上一步结果的前4个字节（8位十六进制）

869ac57b

第八步：把这4个字节加在第五步的结果后面

作为校验位，将这4个字节加载第五步的结果后面，这就是比特币地址的16进制形态了：

000b14f003d63ab31aef5fedde2b504699547dd1f6869ac57b

第九步：用Base58编码变换一下地址

对上一步的结果进行Base58编码，得到：

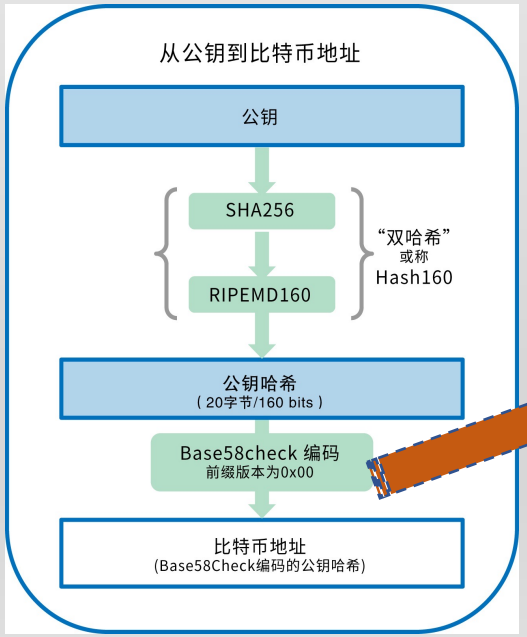
1QCXRuoxWo5Bya9NxHaVBArBQYhatHJrU7

这就是我们经常看到的传统意义上的比特币钱包地址了。

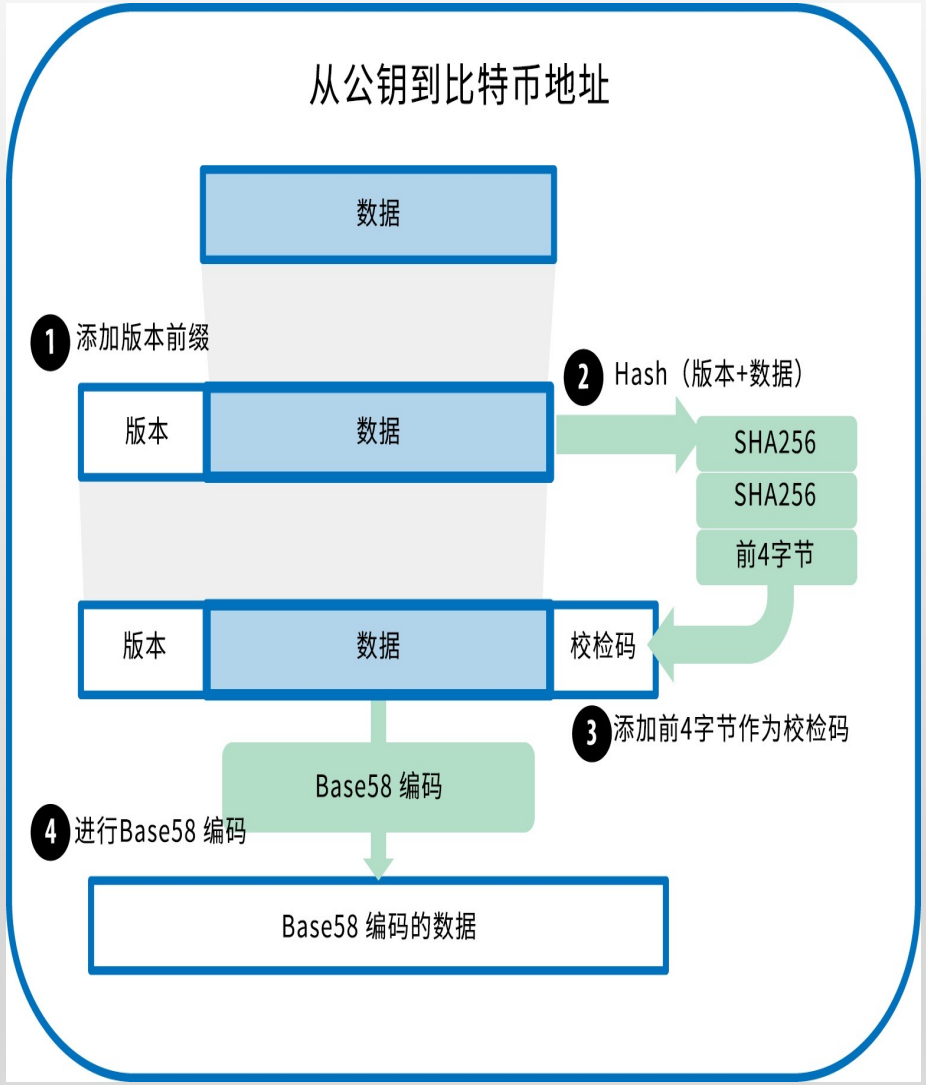
生成比特币区块链地址：用Base58Check编码变换

为了使用Base58Check编码格式对数据(数字)进行编码，首先我们要对数据添加一个称作"版本字节"的前缀，这个前缀用 来明确需要编码的数据的类型。例如，比特币地址的前缀是0(十六进制是0x00)，而对私钥编码时前缀是128(十六进制是 0x80)。

接下来，我们计算"双哈希"校验码，意味着要对之前的结果(前缀和数据)运行两次SHA256哈希算法: checksum = SHA256(SHA256(prefix+data))
在产生的长32个字节的哈希值(两次哈希运算)中，我们只取前4个字节。这4个字节就作为校验码。校验码会添加到数据之 后。
结果由三部分组成:前缀、数据和校验码。这个结果采用之前描述的Base58字母表编码。下图描述了Base58Check编码的 过程。



Base58Check编码



Base58和Base58Check编码

为了更简洁方便地表示长串的数字，许多计算机系统会使用一种以数字和字母组成的大于十进制的表示法。

例如，传统的十 进制计数系统使用0-9十个数字，而十六进制系统使用了额外的 A-F 六个字母。一个同样的数字，它的十六进制表示就会比十 进制表示更短。

更进一步，Base64使用了26个小写字母、26个大写字母、10个数字以及两个符号(例如“+”和“/”)，用于在 电子邮件这样的基于文本的媒介中传输二进制数据。Base64通常用于编码邮件中的附件。

Base58是一种基于文本的二进制 编码格式，用在比特币和其它的加密货币中。这种编码格式不仅实现了数据压缩，保持了易读性，还具有错误诊断功 能。Base58是Base64编码格式的子集，同样使用大小写字母和10个数字，但舍弃了一些容易错读和在特定字体中容易混淆 的字符。具体地，Base58不含Base64中的0(数字0)、O(大写字母o)、l(小写字母L)、I(大写字母i)，以 及“+”和“/”两个字符。

简而言之，Base58就是由不包括(0，O，l，I)的大小写字母和数字组成。

比特币的Base58字母表： 123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

Base58Check编码：一种Base58格式的、有版本的、经过校验的格式，可以明确的对比特币数据编码的编码格式。

比特币地址，Base58Check版本前缀和编码后的结果

在比特币中，大多数需要向用戶展示的数据都使用Base58Check编码，可以实现数据压缩，易读而且有错误检 验。Base58Check编码中的版本前缀是数据的格式易于辨别，编码之后的数据头包含了明确的属性。这些属性使用戶可以轻 松明确被编码的数据的类型以及如何使用它们。例如我们可以看到他们的不同，Base58Check编码的比特币地址是以1开头 的，而Base58Check编码的私钥WIF是以5开头的。下表展示了一些版本前缀和他们对应的Base58格式。

Base58Check版本前缀和编码后的结果

种类	版本前缀 (hex)	Base58格式
Bitcoin Address	0x00	1
Pay-to-Script-Hash Address	0x05	3
Bitcoin Testnet Address	0x6F	m or n
Private Key WIF	0x80	5, K or L
BIP38 Encrypted Private Key	0x0142	6P
BIP32 Extended Public Key	0x0488B21E	xpub

比特币区块链公钥--椭圆曲线方程

椭圆曲线方程（来源于黎曼几何），一条椭圆曲线在射影平面满足一齐次方程——威尔斯特拉斯方程：

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

的所有点的集合，且曲线上的每个点都是非奇异（光滑）的。

椭圆曲线并不是椭圆，是由椭圆曲线的描述方程类似于计算椭圆周长的方程而得名。“非奇异”或“光滑”，可以理解为，满足方程的任意一点都存在切线。

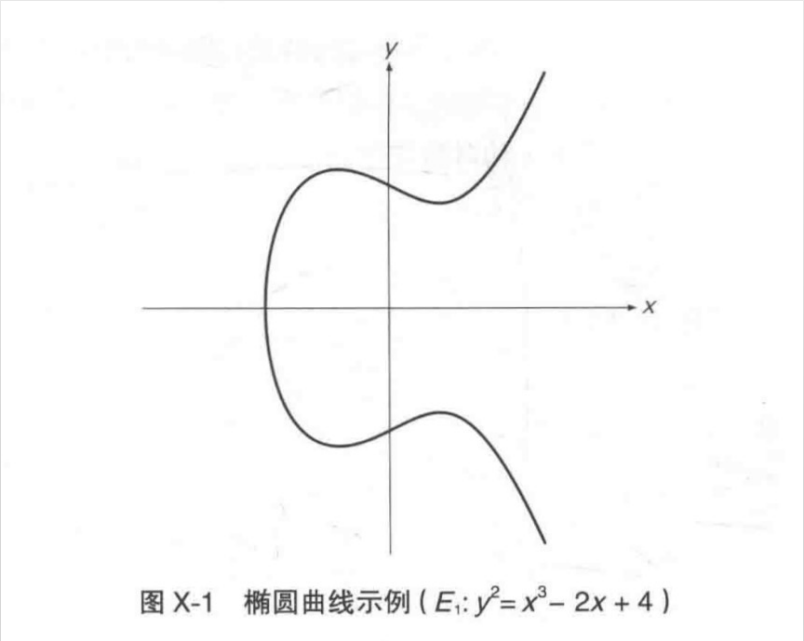
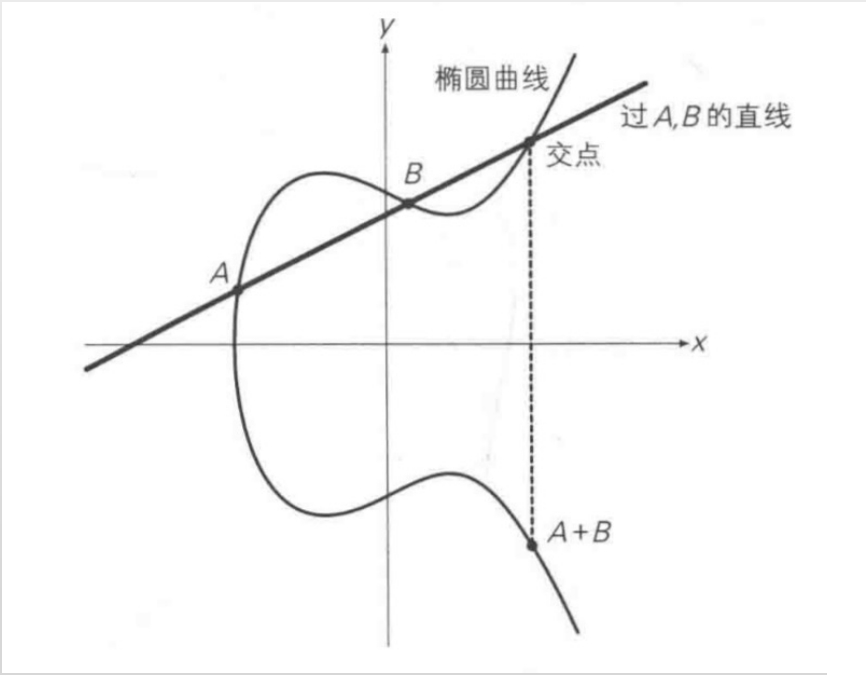


图 X-1　椭圆曲线示例（E_1: $y^2 = x^3 - 2x + 4$）

比特币区块链公钥--椭圆曲线算法操作

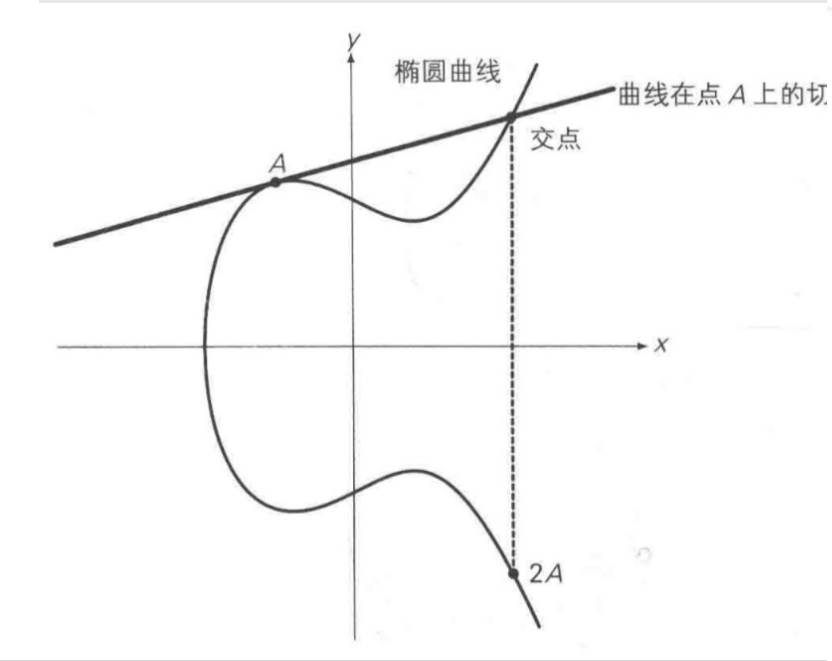
椭圆曲线加法

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



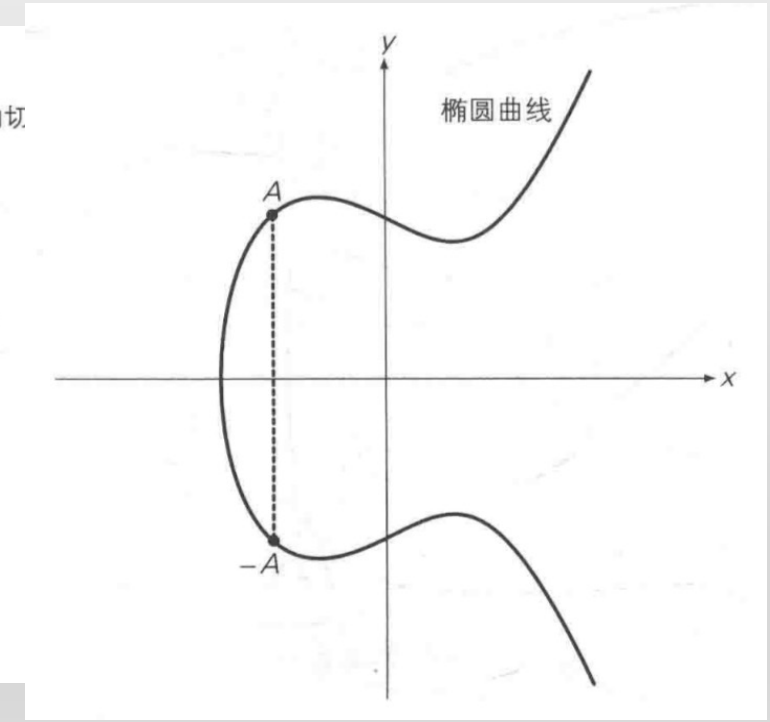
椭圆曲线二倍运算

上述方法无法解释A + A，即两点重合的情况。因此在这种情况下，将椭圆曲线在A点的切线，与椭圆曲线的交点，交点关于x轴对称位置的点，定义为A + A，即2A，即为二倍运算。



椭圆曲线正负取反

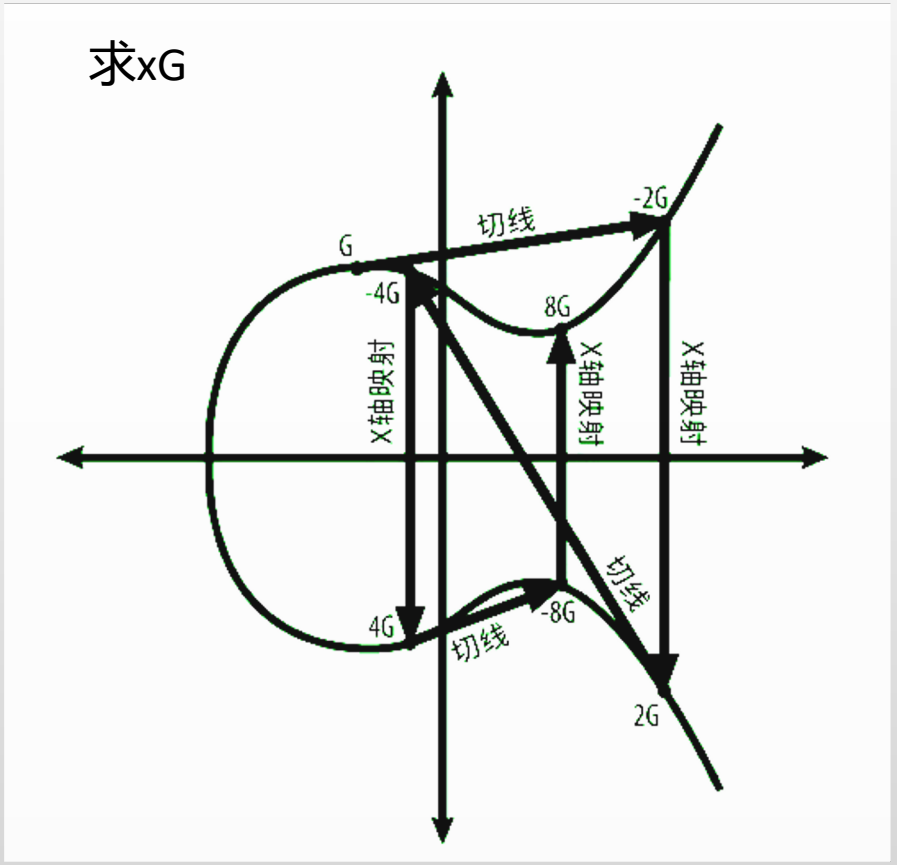
将A关于x轴对称位置的点定义为-A，即椭圆曲线的正负取反运算。如下图所示：



给定椭圆曲线上的两个点P1和P2，则椭圆曲线上必定有第三点 P3 = P1 + P2

比特币区块链公钥--椭圆曲线算法操作

椭圆曲线无穷远点
如果将A与-A相加，过A与-A的直线平行于y轴，可以认为直线与椭圆曲线相交于无穷远点。



综上，定义了A+B、2A运算，因此给定椭圆曲线的某一点G，可以求出2G、3G（即G + 2G）、4G……。即：当给定G点时，已知x，求xG点并不困难。反之，已知xG点，求x则非常困难。此即为椭圆曲线加密算法背后的数学原理。

比特币区块链公钥--椭圆曲线加密

比特币区块链使用了 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 mod p = (x ^ 3 + 7) mod 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，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。

私钥通过椭圆曲线算法可以从私钥计算得到公钥，这是不可逆转的过程: K= k* G 。其中 k是私钥， G 是被称为生成点的常数 点，而 K 是所得公钥。其反向运算，被称为"寻找离散对数"——已知公钥K来求出私钥k——是非常困难的，就像去试验所有可能的k值，即暴力搜索。

比特币区块链公钥

生成公钥

以一个随机生成的私钥k为起点，我们将其与曲线上已定义的 生成点G相乘以获得曲线上的另一点，也就是相应的公钥K。生成点是secp256k1标准的一部分，比特币密钥的生成点都是相同的:

K = k*G

其中k是私钥，G是生成点，在该曲线上所得的点K是公钥。因为所有比特币用戶的生成点是相同的，一个私钥k乘以G将得到 相同的公钥K。k和K之间的关系是固定的，但只能单向运算，即从k得到K。这就是可以把比特币地址(K的衍生)与任何人 共享而不会泄露私钥(k)的原因。

为实现椭圆曲线乘法，我们以之前产生的私钥k和与生成点G相乘得到公钥K:

K = 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD * G

公钥K 被定义为一个点 K = (x, y)，其中:

x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A

y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB

比特币区块链公钥：椭圆曲线加密ECC（elliptic curve cryptography）

椭圆曲线加密ECC

考虑K = k G，其中K、G为椭圆曲线Ep(a,b)上的点，n为G的阶（n G = O ∞），k为小于n的整数。则给定k和G，根据加法法则，计算K很容易但反过来，给定K和G，求k就非常困难。因为实际使用中的ECC原则上把p取得相当大，n也相当大，要把n个解点逐一算出来是不可能的。这就是椭圆曲线加密算法的数学依据 。

- 点G称为基点（base point）
- k (k < n) 为私有密钥（private key）
- K为公开密钥（public key）

下面是利用椭圆曲线进行加密通信的过程：

1、用户A选定一条椭圆曲线Ep(a,b)，并取椭圆曲线上一点，作为基点G。
2、用户A选择一个私有密钥k，并生成公开密钥K=kG。
3、用户A将Ep(a,b)和点K，G传给用户B。
4、用户B接到信息后，将待传输的明文编码到Ep(a,b)上一点M（编码方法很多，这里不作讨论），并产生一个随机整数r（r<n）。
5、用户B计算点C 1 = M + r K 和C 2 = r G 。
6、用户B将C 1 、 C 2 传给用户A。
7、用户A接到信息后，计算C 1 – k C 2 ，结果就是点M。再对点M进行解码就可以得到明文。
因为C 1 – k C 2 = M + r K – k (r G) = M + r k G – k r G = M

回顾：比特币区块链地址

比特币地址是一个由数字和字母组成的字符串，可以与任何想给你比特币的人分享。由公钥(一个同样由数字和字母组成的 字符串)生成的比特币地址以数字“1”开头。下面是一个比特币地址的例子:
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy

在交易中，比特币地址通常以收款方出现。如果把比特币交易比作一张支票，比特币地址就是收款人，也就是我们要写入收款人一栏的内容。一张支票的收款人可能是某个银行账戶，也可能是某个公司、机构，甚至是现金支票。支票不需要指定一 个特定的账戶，而是用一个普通的名字作为收款人，这使它成为一种相当灵活的支付工具。与此类似，比特币地址的使用也使比特币交易变得很灵活。比特币地址可以代表一对公钥和私钥的所有者，也可以代表其它东西，比如在 “P2SH (Pay-to-Script-Hash)”一节讲到的付款脚本。

比特币地址可由公钥经过单向的加密哈希算法得到。哈希算法是一种单向函数，接收任意长度的输入产生指纹摘要。加密哈 希函数在比特币中被广泛使用:比特币地址、脚本地址以及在挖矿中的工作量证明算法。由公钥生成比特币地址时使用的算法是Secure Hash Algorithm (SHA)和the RACE Integrity Primitives Evaluation Message Digest (RIPEMD)，特别是 SHA256和RIPEMD160。

以公钥 K 为输入，计算其SHA256哈希值，并以此结果计算RIPEMD160 哈希值，得到一个长度为160比特(20字节)的数 字:
A = RIPEMD160(SHA256(K))
公式中，K是公钥，A是生成的比特币地址。

通常用户见到的比特币地址是经过“Base58Check”编码的，这种编码使用 了58个字符(一种Base58数字系统)和校验码，提高了可读性、避免歧义并有效防止了在地址转录和输入中产生的错 误。Base58Check编码也被用于比特币的其它地方，例如比特币地址、私钥、加密的密钥和脚本哈希中，用来提高可读性和 录入的正确性。

谢 谢!

VICTORLAMP

Shenzhen VictorLamp Technologies CO. Ltd.
http://www.victorlamp.com

多媒体课程：《深入浅出区块链技术核心篇》

深入浅出区块链技术