公钥和私钥的都可以有多种编码格式。一个密钥被不同的格式编码后，虽然结果看起来可能不同，但是密钥所编码数字并没有改变。这些不同的编码格式主要是用来方便人们无误地使用和识别密钥。

私钥的格式

私钥可以以许多不同的格式表示，所有这些都对应于相同的 256 位的数字。。

私钥表示法（编码格式）

种类版本描 述

Hex None64 hexadecimal digits

Base58Check encoding: Base58 with version

WIF 5

prefix of 128 and 32-bit checksum

WIF-compressedK or LAs above, with added suffix 0x01 before encoding 表 4-3 展示了用这三种格式所生成的私钥。

同样的私钥，不同的格式格式私钥

1E99423A4ED27608A15A2616A2B0E9E52CED330AC530ED

Hex

WIF

WIF-compr essed

CC32C8FFC6A526AEDD

5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB 1Jcn
KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGW ZgawvrtJ

这些表示法都是用来表示相同的数字、相同的私钥的不同方法。虽然编码后的字符串看起来不同，但不同的格式彼此之间可以很容易地相互转换。

将 Base58Check 编码解码为十六进制sx工具包（参见"Libbitcoin 和 sx Tools"）可用来编写一些操作比特币密钥、地址及交易的 shell 脚本和命令行"管道"。你也可以使用 sx 工具从命令行对 Base58Check 格式进行解码。

我们使用的命令是 base58check-decode：

$sxbase58check-decode5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd 128

所得结果是十六进制的密钥，紧接着是钱包导入格式（Wallet Import Format,WIF）的版本前缀 128。

将十六进制转换为 Base58Check 编码要转换成 Base58Check 编码（和之前的命令正好相反），我们需提供十六进制的私钥和钱包导入格式（Wallet Import Format，WIF）的版本号前缀 128：

$sx base58check-encode 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd 128

5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn

将十六进制（压缩格式密钥）转换为 Base58Check 编码

要将压缩格式的私钥编码为 Base58Check（参见"压缩格式私钥"一节），我们需在十六进制私钥的后面添加后缀 01，然后使用跟上面一样的方法：

$ sx base58check-encode 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a

526aedd01 128
KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

生成的 WIF 压缩格式的私钥以字母"K"开头，用以表明被编码的私钥有一个后缀"01"，且该私钥只能被用于生成压缩格式的公钥（参见"压缩格式公钥"一节）。

公钥的格式

公钥也可以用多种不同格式来表示，最重要的是它们分为非压缩格式或压缩格式公钥这两种形式。

我们从前文可知，公钥是在椭圆曲线上的一个点，由一对坐标（x，y）组成。公钥通常表示为前缀 04 紧接着两个 256 比特的数字。其中一个 256 比特数字是公钥的 x 坐标，另一个 256 比特数字是 y 坐标。前缀 04 是用来区分非压缩格式公钥，压缩格式公钥是以 02 或者 03 开头。

下面是由前文中的私钥所生成的公钥，其坐标 x 和 y 如下：

x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A

y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB

下面是同样的公钥以 520 比特的数字（130 个十六进制数字）来表达。这个 520

比特的数字以前缀 04 开头，紧接着是 x 及 y 坐标，组成格式为 04 x y：

K = 04F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF

579DC341A07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328

AE52DDFE2E505BDB

压缩格式公钥

引入压缩格式公钥是为了减少比特币交易的字节数，从而可以节省那些运行区块链数据库的节点磁盘空间。大部分比特币交易包含了公钥，用于验证用户的凭据和支

付比特币。每个公钥有520比特（包括前缀，x 坐标，y 坐标）。如果每个区块有数百个交易，每天有成千上万的交易发生，区块链里就会被写入大量的数据。

正如我们在"公钥"一节所见，一个公钥是一个椭圆曲线上的点(x, y)。而椭圆曲线实际是一个数学方程，曲线上的点实际是该方程的一个解。因此，如果我们知道了公钥的 x 坐标，就可以通过解方程y2mod p = (x3 + 7) mod p 得到 y 坐标。这种方案可以让我们只存储公钥的 x 坐标，略去 y 坐标，从而将公钥的大小和存储空间减少了 256 比特。每个交易所需要的字节数减少了近一半，随着时间推移，就大大节省了很多数据传输和存储。

未压缩格式公钥使用 04 作为前缀，而压缩格式公钥是以 02 或 03 作为前缀。需要这两种不同前缀的原因是：因为椭圆曲线加密的公式的左边是 y2，也就是说 y 的解是来自于一个平方根，可能是正值也可能是负值。更形象地说，y 坐标可能在 x 坐标轴的上面或者下面。从图的椭圆曲线图中可以看出，曲线是对称的，从 x 轴看就像对称的镜子两面。因此，如果我们略去 y 坐标，就必须储存 y 的符号（正值或者负值）。换句话说，对于给定的 x 值，我们需要知道 y值 在 x 轴的上面还是下面，因为它们代表椭圆曲线上不同的点，即不同的公钥。当我们在素数 p 阶的有限域上使用二进制算术计算椭圆曲线的时候，y 坐标可能是奇数或者偶数，分别对应前面所讲的 y 值的正负符号。因此，为了区分 y 坐标的两种可能值，我们在生成压缩格式公钥时，如果 y 是偶数，则使用 02 作为前缀；如果 y 是奇数，则使用 03 作为前缀。这样就可以根据公钥中给定的 x 值， 正确推导出对应的 y 坐标，从而将公钥解压缩为在椭圆曲线上的完整的点坐标。下图阐释了公钥 压缩：