第 3 章

比特币技术原理

比特币作为一种新型的数字货币，开创了全新的数字货币生态系统。不同于日常生活中使用的货币，数字货币在交易过程中并不依赖物理实体，其使用前景将远超过目前看到的支付、担保、保险、博彩、公证等应用场景。随着以比特币为首的数字货币市场引发的一轮投资热潮，越来越多的人将注意力集中在数字货币的应用和背后支撑的技术原理，而比特币作为数字货币的领头羊，更是获得了不同群体的广泛关注。如何拥有属于自己的比特币？如何使用比特币进行交易？比特币如何确保安全性？本章将介绍如何加入比特币网络，如何创建比特币账户，如何进行比特币交易，并重点阐述比特币的共识机制、安全机制，以及目前的扩容方案。

3.1 加入比特币网络

与传统货币不同，比特币是完全虚拟的。用户通过网络进行交易，比特币隐含在发送方和接收方转移价值的交易中，通过交易数据表示，不存在任何实物。网络中的任何参与者都可以作为“矿工”使用计算机的处理能力来验证和记录交易，这些交易信息通过矿工存储在网络的不同节点中，无法篡改。但并非所有节点都强制存储完备的交易记录，比特币网络针对不同需求的用户拥有不同的节点类型。

3.1.1 网络节点

比特币网络采用 P2P 网络结构，每个节点在网络中地位对等，它们为用户提供相同的网络服务。可以按照比特币网络 P2P 协议运行的一系列节点的集合称作比特币网。中本聪在比特币白皮书中说明了如何运行比特币网络，大致包括如下过程：

(1) 新的交易向全网进行广播。

(2) 每一个节点都将收到的交易信息纳入一个区块中。

(3) 每个节点都尝试在自己的区块中找到一个具有足够难度的工作量证明。

(4) 当一个节点找到了一个工作量证明，它就向全网进行广播。

(5) 当且仅当包含在该区块中的所有交易都是有效的且之前未存在过的，其他节点才认同该区块的有效性。

（6）其他节点表示它们认同该区块的方法，就是在该区块的后面添加新的区块以延长该链条，并将被认同区块的随机散列值视为先于新区块的随机散列值。

其中涉及的节点被称为全节点。每个完全节点都是路由、完整的区块链数据库、挖矿和钱包四种功能服务的集合，如图 3-1 所示。它们更新复制最新完整的区块链数据库，能够独立自主校验所有交易并对其进行广播，不需借由任何外部参照，同时可以运行设备的计算能力参与新区块的算力竞争。一般的核心客户端都是能够运行所有功能的全节点，如 Bitcoin core。

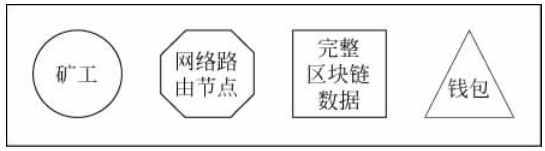

图 3-1　全节点包含功能模块示意图

但事实上，运行全节点对设备提出了较高的要求，需要存储的区块数据会随着交易数量的增加而日益庞大，对于只是想将比特币作为货币使用的用户而言，存储大量的区块数据是多余的。因此在现行的比特币网络中，针对不同的用户群体，存在不同功能集合的节点类型。

对于只想借助比特币充当一种交易方式的用户而言，参与网络后运行的节点可以是全节点或者轻量级节点，只要节点包含钱包功能即可。轻量级节点对比全节点，只保留了区块链数据的一部分，通过简易支付验证的方式完成交易验证，交易数据实时更新。用户只需要下载轻量级钱包，就能够使用比特币进行交易，轻量级钱包包含功能如图 3-2 所示。

而矿工主要依靠挖矿节点参与网络。一般维持挖矿节点运行的设备均配置有特殊硬件设施，节点间通过计算力竞争，破解新区块的工作量证明解。挖矿节点也分为完全节点和轻量级节点，其中轻量级节点依靠矿池服务器的完全节点进行工作，而完全节点一般指的是依靠单一节点进行挖矿的独立矿工节点。节点具有完整区块链副本的挖矿功能，以及比特币网络中的路由功能，具体如图 3-3 所示。

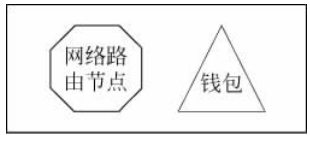

图 3-2　轻量级钱包包含功能示意图

图 3-3　独立矿工节点包含功能示意图

但是随着挖矿难度的日益增加，算力竞争愈发激烈，个体矿工通过独立挖矿获得的收益已经不能覆盖电力和硬件成本了。即使使用消费型 ASIC 进行挖矿，个体矿工也无法与拥有数万芯片、位于低电力成本地区的商业矿池进行竞争。所以，现在的矿工多通过组成矿池方法，汇集众多参与者的算力，凭借算力贡献按比例获取奖励，降低了风险性和不确定性。

矿池通过专用的挖矿协议协调矿工，而矿工会将个人的矿机连接到矿池服务器，通过服务

器和其他矿工同步工作。这种情况下矿工可以选择轻量级的挖矿节点，如图 3-4 所示。轻量级节点保存的只是区块头信息，通过路由功能连接到比特币网络中，就可以通过 P2P 的方式找到中继节点，从而搜索到所需要的交易信息。因此没有足够空间存储区块数据的矿工节点，可以不保存全节点的链数据，按需所取，专注于挖矿即可。

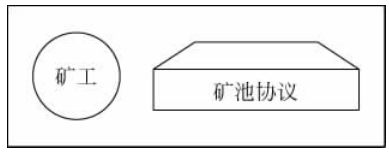

图 3-4　轻量级矿工节点包含功能示意图

3.1.2　比特币客户端

普通用户可以通过在线网站或者下载应用程序加入比特币网络。随着比特币的热潮，出现了很多比特币客户端软件，主要可以分为三种类型：完整客户端、轻量级客户端和在线客户端，其中中本聪客户端被称为标准客户端，标准客户端最开始由中本聪运行维护。

完整客户端存储比特币区块链的全部交易信息，用户可以直接进行交易，不依赖任何第三方服务器确认交易记录；轻量级客户端只存储用户钱包的相关信息，如果想要进行交易，需要访问第三方服务器中存储的交易记录；在线客户端完全依赖第三方服务器，用户通过网页浏览器访问和储存钱包。

用户钱包由系统产生的密钥进行保护，用户需要牢记自己的钱包密钥以确保对钱包中比特币的所有权。客户端会为每一个用户产生一个钱包和对应的比特币地址，一个用户可以有多个钱包，每笔交易也可以有不同的地址。

获得比特币的方式，除了通过挖矿获得开发新区块的比特币奖励和确认交易的交易费以外，用户也可以通过专门的通货交易所进行比特币购买，或者寻找该地区的比特币卖家使用现金进行线下交易，如果本地区存在比特币 ATM，那么可以直接在 ATM 获取。

下面简单介绍几个比特币在线交易平台。

1. Bitstamp

欧洲比特币交易所（Bitstamp）的网址是 https://www.bitstamp.net/，该平台支持多币种的交易和电汇方式，网站首页如图 3-5 所示。

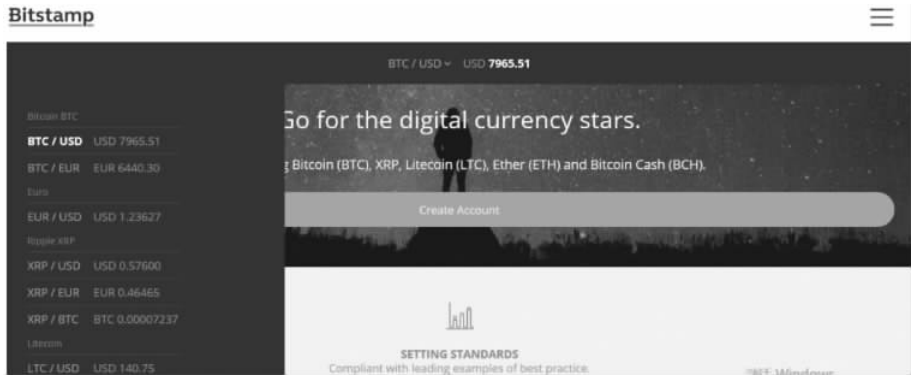

图 3-5　Bitstamp 网站首页截图

2. Coinbase

美国比特币钱包和交易平台(Coinbase)的网址是 https://www.coinbase.com/,该平台已获得美国多个州监管机构的合法执照,可以通过 ACH 系统连接美国支票账户。网站首页如图 3-6 所示。

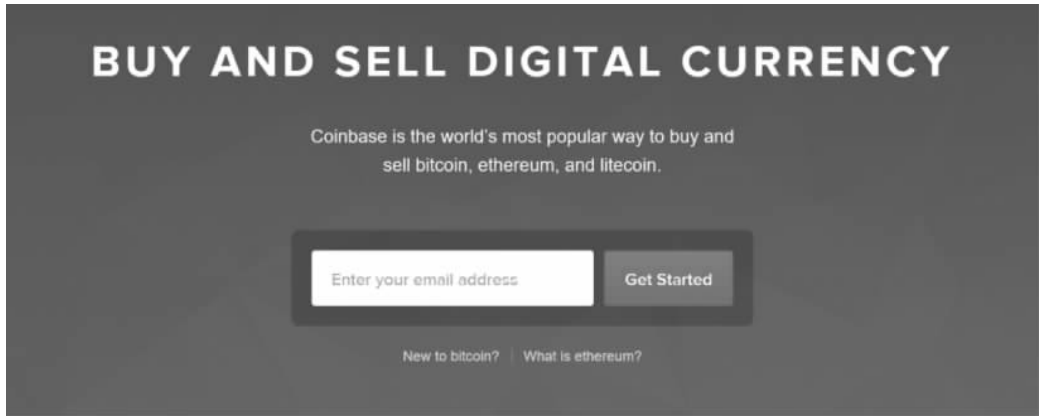

图 3-6 Coinbase 网站首页截图

3. Localbitcoins

总部设在芬兰的 LocalBitcoins,是目前全球最大的场外交易平台,通过该平台可以寻找当地比特币卖家,网址是 https://localbitcoins.com/,网站首页如图 3-7(a)所示,通过 localbitcoins 搜索中国比特币卖家的结果如图 3-7(b)所示。

4. CoinDesk

比特币新闻资源网(CoinDesk)是一个发布数字货币新闻和数据分析的平台,利用该平台提供的比特币 ATM 在线地图可以非常方便地查询本地区的比特币 ATM,网址是 https://www.coindesk.com/bitcoin-atm-map/,网站如图 3-8 所示。

比特币作为一种全球性流通的数字货币,可以在世界范围内进行交易,同时也可以换算为其他不同的币种,用户可以通过查询比特币市场汇率来获得第一手信息,有很多应用和网站都能满足该项需求,下面列举几个。

1. Bitcoincharts

Bitcoincharts 是市场数据服务网站,网址是 https://bitcoincharts.com/,该网站显示了全球众多交易所的比特币市场汇率,以当地不同的汇率来进行结算,网站如图 3-9 所示。

2. BitcoinAverage

BitcoinAverage 提供每个币种的交易量加权平均价格的简单视图,是比特币历史价格数据的主要来源,网址是 https://bitcoinaverage.com/,网站如图 3-10 所示。

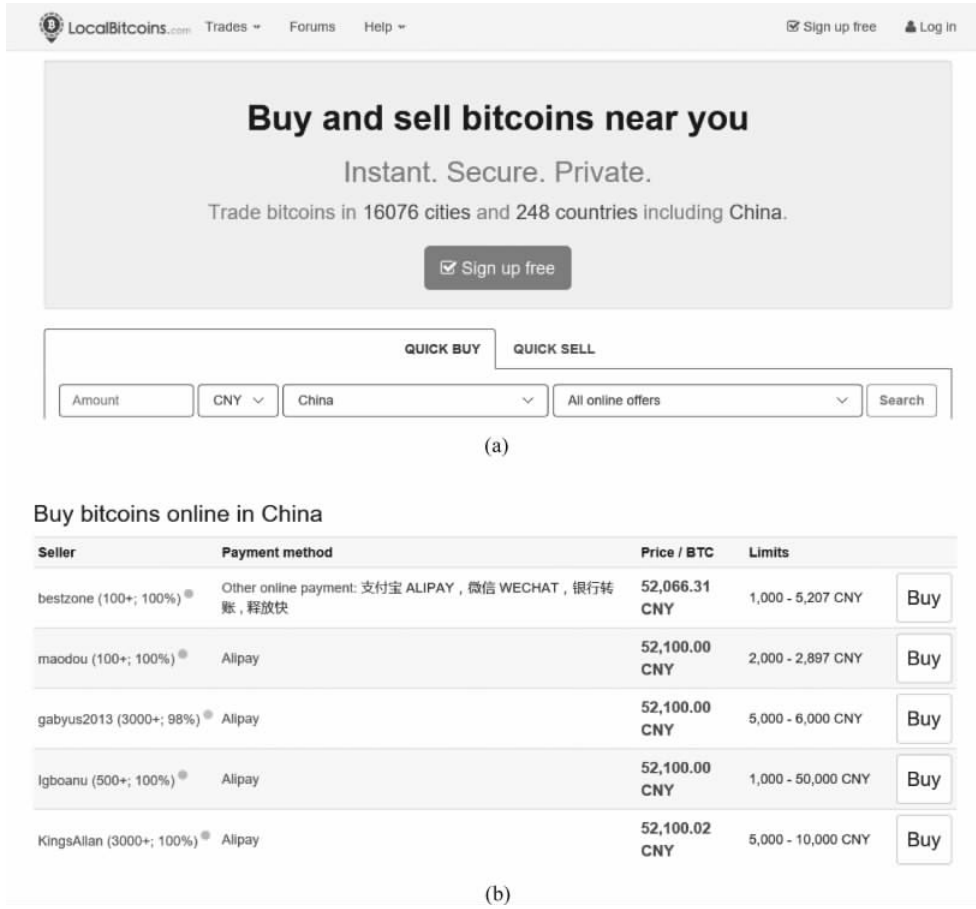

图 3-7　Localbitcoins 网站首页截图和通过 Localbitcoins 搜索中国比特币卖家结果截图

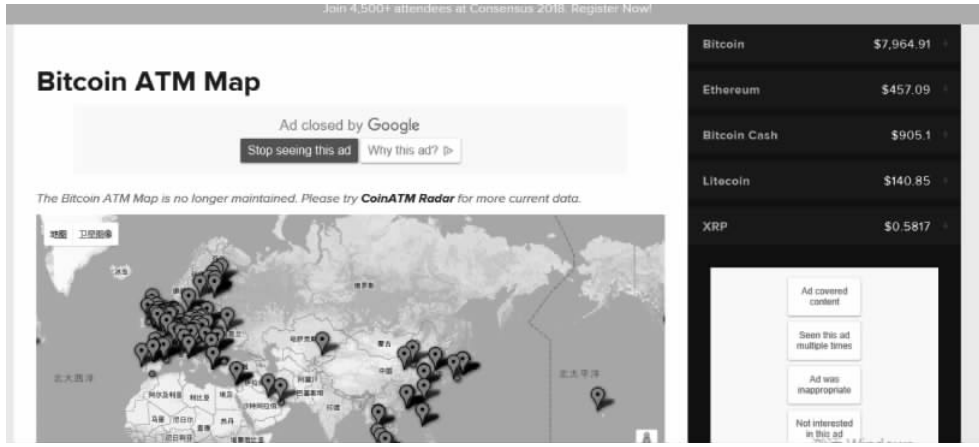

图 3-8　CoinDesk 网站截图

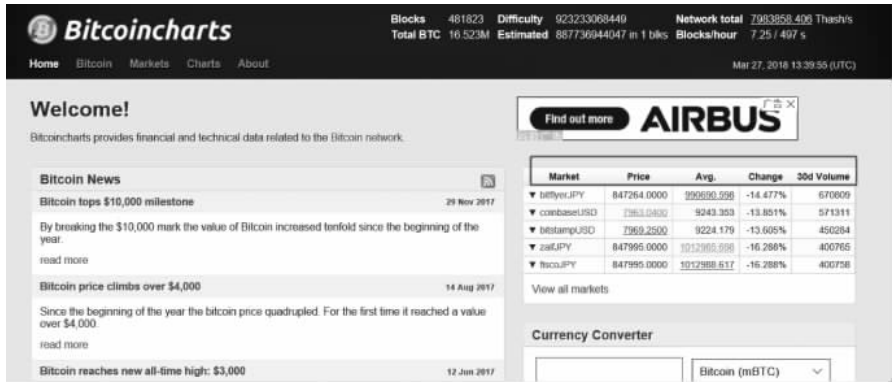

图 3-9　Bitcoincharts 网站截图

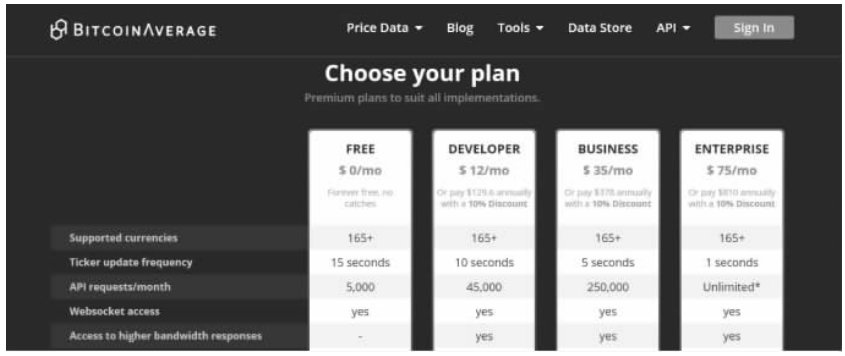

图 3-10　BitcoinAverage 网站截图

3. ZeroBlock

ZeroBlock 是一个免费的安卓和 iOS 应用程序，可以显示不同交易所的比特币价格，网址是 https://zeroblock.com/，网站如图 3-11 所示。

图 3-11　ZeroBlock-PC 端截图

4. BitcoinWisdom

BitcoinWisdom 是一家数字货币行情网站，提供市场数据索引服务站，网址是 https://bitcoinwisdom.com/，网站如图 3-12 所示。

图 3-12　BitcoinWisdom 网站截图

3.2　创建比特币账户

对于比特币的交易用户，客户端会自动为用户生成钱包，类似于在银行开户，用户也需要一个如同银行卡密码的账户密码来保护自己的数字资产。专属密钥就成为使用钱包中比特币的必要条件。而用户的密钥不同于银行卡密码短小精悍方便记忆，密钥对一般是经过特殊处理的长串随机数列，很难记忆，为了保证其安全性和隐匿性，一般将密钥存储在数字钱包中。

3.2.1　密钥对：私钥和公钥

比特币作为一种数字资产，并不存在实体，网络通过密钥、比特币地址和数字签名来确认比特币的所有权。用户通过客户端或钱包自动生成密钥文件，存储在本地，密钥为自己独有，不需要进行区块链的网络连接；用户在交易环节，公钥通过其数字指纹表示为比特币地址，同时比特币地址也存在脚本类型等其他表现形式，但公钥最为常见；数字签名证明了交易的有效性，交易只有携带有效的数字签名才能存储在区块链中。

密钥对是比特币地址和数字签名的基础。一个比特币钱包可以包含一系列的密钥对，一个密钥对包含一个私钥和一个公钥。私钥为系统生成的随机数，用于产生支付时的数字

签名。公钥是私钥通过椭圆曲线乘法产生的，用于产生地址接收比特币，如图 3-13 所示。公钥是公开的，不会影响用户钱包的安全性，但是用户必须对私钥进行保密储存，同时注意对私钥文件的多重备份。在生活大爆炸中，伦纳德、霍华德和拉杰什就因为谢尔顿弄丢了私钥的备份文件，失去了一夜暴富的机会。因为私钥一旦丢失就难以复原，所有的比特币也将一同消失。

图 3-13　私钥、公钥和比特币地址关系图

私钥通过椭圆曲线乘法生成公钥，这个过程是单向不可逆的，公钥无法反推出私钥。椭圆曲线乘法的原理已在第 2 章进行了阐述：在椭圆曲线上，以一个随机产生的私钥 k 为起点，将其与曲线上已经规定的一点 G 相乘，即可获得公钥。G 点为标准规定的一部分，所有比特币的生成点都是相同的。

按照定义，根据椭圆曲线乘法生成的公钥实际上为一个点(x,y)的坐标。因为区块中的交易数据包含了公钥字段，为了优化数据结构、压缩硬盘储存的区块链数据，同时根据椭圆曲线算法公式利用 x 值能够推导出 y 值，因此引进了压缩公钥。根据选用的公钥是(x,y)坐标值还是 x 值，将公钥划分为非压缩格式和压缩格式两种类型。非压缩格式代表公钥选用了坐标(x,y)表示，通常此种情况下的公钥前缀为 04；压缩格式为公钥只选用 x 值，而舍弃了坐标中的 y 值。根据 x 值的奇偶性不同，前缀可分为 02 或 03，如图 3-14 所示。

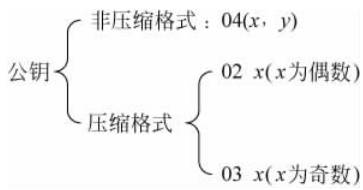

图 3-14　公钥压缩类型

私钥是用户在比特币网络的通行证，它是由客户端随机生成的一个 256bit 的二进制码，也就是 32 字节，在不同的使用场景中采用不同的编码方式可推导出不同格式的私钥。表 3-1 列出了各种类型的私钥格式。

表 3-1　私钥的格式

类型	前缀	特 征 描 述
Raw	无	32 字节
Hex	无	64 位十六进制数
WIF	5	采用 Base58Check 编码，版本前缀为 128，有 32 位的校验和
WIF-compressed	K/L	采用 Base58Check 编码，版本前缀为 128，有 32 位的校验和，有附加后缀 01

32 字节的字符串是原生(Raw)形式,其十六进制(Hex)形式多在编码中使用,普通用户不会接触到。为了方便复制私钥并减少出错,在钱包之间导入/导出私钥时一般使用 WIF (wallet import format)格式。在 WIF 格式的私钥后增加后缀 01,表明该私钥只能用于生成压缩格式的公钥,将该形式的私钥称为压缩格式私钥(WIF-compressed),对应的有时也会将 WIF 格式的私钥称为非压缩格式的私钥。同一个 256bit 的随机数对应的不同格式的私钥如表 3-2 所示。

表 3-2　不同格式私钥举例

类型	举　　例
Hex	1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
WIF	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
WIF-compressed	KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

需要留意的是,压缩格式的私钥并没有压缩,反而比 WIF 非压缩格式多出 1 字节。一般所说的压缩格式的私钥只能用于生成压缩格式的公钥,其本身并不是压缩格式。同时,非压缩格式的私钥只能用于生成非压缩格式的公钥,具体转换关系如图 3-15 所示。

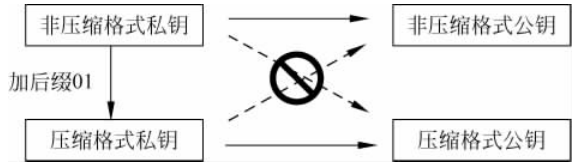

图 3-15　压缩、非压缩格式私钥和公钥的对应关系

WIF 格式的私钥产生过程如图 3-16 所示。

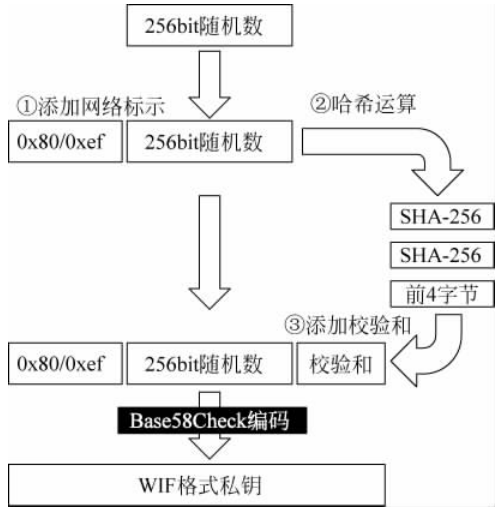

图 3-16　WIF 格式私钥产生过程示意图

（1）系统产生 256bit 的随机数，用十六进制表示，例如产生一个随机数：0C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D。

（2）对随机数添加网络标示，前缀 0x80 表示 mainnet 网络，前缀 0xef 表示 testnet 网络，例如对上一步产生的随机数添加网络标示 0x80 之后生成 800C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D。

（3）如果需要产生压缩公钥，那么需要使用压缩私钥，在字符串末尾增加后缀 01；若使用非压缩公钥，则不追加。例如使用非压缩公钥，字符串保持不变，依然是 800C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D。

（4）对步骤 3 产生的字符串执行 SHA-256 算法，生成 8147786C4D15106333BF278D71DADAF1079EF2D2440A4DDE37D747DED5403592。

（5）对步骤 4 产生的字符串再次执行 SHA-256 算法，生成 507A5B8DFED0FC6FE8801743720CEDEC06AA5C6FCA72B07C49964492FB98A714。

（6）取步骤 5 产生的字符串的前 4 字节作为校验和，添加至步骤 3 产生的字符串末尾，生成 800C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D507A5B8D。

（7）执行 Base58Check 编码算法，得到 WIF 格式私钥为 5HueCGU8rMjxEXxiPuD5BDku4MkFqeZyd4dZ1jvhTVqvbTLvyTJ。

3.2.2 比特币地址

人们可以根据 E-mail 地址互相发送邮件，通过 E-mail 地址对应的密码可以读取邮件内容。类似 E-mail 系统，比特币网络中的交易实际就是根据比特币地址发送比特币，拥有某个比特币地址的私钥就可以获得这个比特币地址中的比特币。

比特币地址是由数字和字母组成的字符串，公钥通过一系列哈希算法和编码算法生成比特币地址，可以简单地将比特币地址理解为公钥的一种摘要表示。因为公钥产生地址利用了单向哈希算法，不具备可逆性，所以地址不能反推出公钥。比特币地址并不是固定且唯一的，用户的一个钱包就可以包含多个比特币地址，钱包能够针对不同的交易产生不同的地址，从而保证账户的安全性。

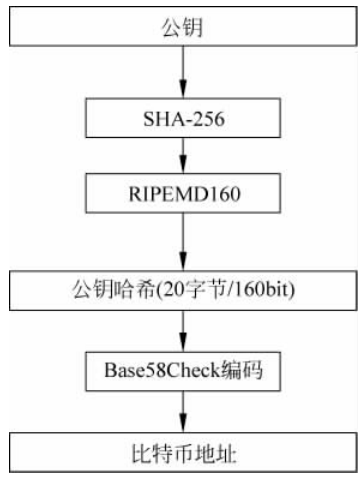

图 3-17　公钥产生比特币地址的示意图

公钥通过 SHA-256 和 RIPEMD160 的双哈希算法生成 160bit(20 字节)的公钥哈希。为了提高地址的可读性和鲁棒性，公钥哈希还会经过 Base58Check 编码生成最终的比特币地址。Base58Check 编码算法利用了 Base58 数字系统中的 58 个字符和校验码，能够有效防止地址在具体使用过程中产生的错误，具体过程如图 3-17 所示。

3.2.3　数字钱包

私钥作为随机产生毫无规律的随机数，如果仅凭用户大脑记忆很容易产生私钥错误或遗忘。因此在具体使用过程中，钱包不仅生成密钥对和地址，还能帮助用户存储它们，一个钱包中可以存储多个私钥。但是钱包客户端可能也会出现数据丢失或其他遗失私钥的可能。因此，如何安全方便地生成、保存和备份密钥是判断钱包性能好坏的关键因素。

为了完善这些因素，数字钱包不断发展，经历了大致三个阶段：非确定性(随机)钱包、确定性(种子)钱包和分层确定性钱包。

非确定性钱包，也称为随机钱包，如图 3-18(a)所示。钱包只是随机生成的私钥们的储存容器，私钥之间没有任何联系。相互独立的私钥拥有更好的匿名性，如果一个私钥丢失或者被盗，非确定性钱包能够将损失风险降到最低，账户的安全性得到保障。大量随机私钥意味着，一旦出现新的私钥，用户就需要重新将所有私钥文件再备份一遍，每次备份后都要再次导入，工作重复且烦琐，用户使用体验并不良好。

为了方便用户操作，解决经常性备份的问题，出现了确定性(种子)钱包。在种子钱包中，所有私钥都是通过一个公共的种子文件生成的，种子能够回收所有已经产生的私钥，用户只需要备份种子，就能够完成对所有私钥的备份。如果想要更换钱包，只需要将种子文件导入即可，如图 3-18(b)所示。

在 BIP0032/BIP0044 协议中，又对确定性钱包做了进一步的升级，即分层确定性钱包，如图 3-18(c)所示。对比确定性钱包，共同之处是，两者的所有私钥都是通过一个公共种子文件生成的，但不同于确定性钱包中所有私钥的并行关系，分层确定性钱包中私钥的衍生结构是树状结构。父密钥可以衍生一系列子密钥(例如：私钥 1、私钥 2、私钥 3)，每个子密钥又可以衍生一系列孙密钥(例如：私钥 2-1、私钥 2-2、私钥 2-3)，以此类推。根据其树状结构，分层确定性钱包的应用场景和安全性得到了进一步的扩展和提升。树状结构可以表示额外的组织架构的含义，因此，比如在企业环境中，不同职能部门可以被分配使用不同分支的密钥，也可以一个特定的分支子密钥用来接收收入而另一个分支的密钥用来支付花费。另外，如果想要在不安全的环境下进行交易，分层确定性钱包保证了在每笔不同的交易中能够发行不同的公钥，而不需要访问相对应的私钥，保证了账户的安全性。

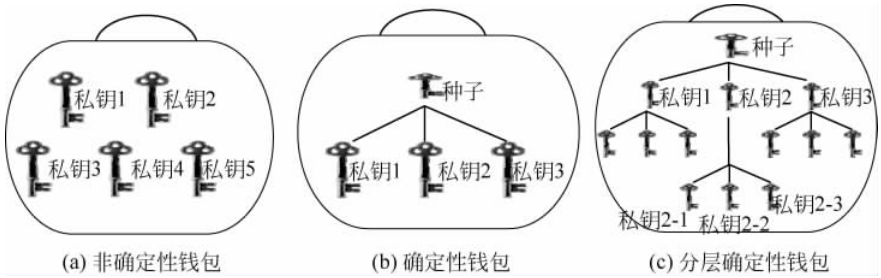

图 3-18　三种数字钱包示意图

3.3 比特币交易过程

用户如果想发起一笔交易，需要指定目标地址和金额，其余细节都会在比特币客户端（钱包）后台进行确认。成功创建交易后，交易会被广播至比特币网络，每个节点独立验证交易的有效性后进行再次传播，直到交易被纳入新区块存储在区块链中完成最终确认，这就是比特币进行交易的整体流程。

3.3.1 UTXO

例如 Bob 花费了一笔钱，购买了 Alice 店里的汉堡，但同时 Bob 又想利用同一笔钱款向 Carol 购买她店里的比萨。在现实中这种可能是完全不存在的，除非 Bob 将支付给 Alice 的钱偷回来。

但是因为比特币是一种没有物理实体的数字货币，存在数据的可复制性，所以很可能存在同一笔资产因不当操作被重复使用的情况，这就是“双花”问题，即双重支付。

双花问题是任何一种数字货币都需要解决的问题。现金容易被交易双方确认，但数字化货币的确认并不容易。相同数额的数字货币背后的数据是相同的，而伪造数据的成本低于伪造现金。

在生活中被大量使用的支付宝、微信支付等线上支付虽然也没有使用现金，是如何避免双花问题的呢？存储在支付宝账户中的货币，支付宝对其进行中心化管理，通过实时修改账户余额数据保证不会出现双花问题；如果通过支付宝花费绑定银行卡中的钱币，支付宝仅充当了一个第三方的中介，用户实际花费的是存在于银行账户的钱财。银行从交易和货币本身来控制，假如 Bob 的卡中只有 100 元的余额，想利用银行处理交易之间的时间差进行双重支付，几乎在同一时间花费了这 100 元，银行也会按照顺序一笔一笔处理，同时同一张卡的磁道信息难以复制，银行也利用了信息安全加密等技术杜绝了同一笔钱款的多次重复使用。

但是比特币作为一种去中心化的点对点电子现金系统，不同于使用纸币或黄金等真实存在的物理等价物，也不像电子货币有第三方中心机构进行严格的监控。比特币网络整合利用了未花费的交易输出（unspent transaction output，UTXO）、时间戳等技术来解决双花问题。

在比特币中没有余额的概念，UTXO 是交易的基本单元，不能再分割。UTXO 分散存在于区块链中。每笔比特币交易都是一个创建新的 UTXO 的过程：给某人的地址发送比特币的过程，相当于是给其的地址所代表的用户创造了新的 UTXO，这些 UTXO 能被该用户用于新的支付，交易的输入也只能由之前交易创建的 UTXO 组成。一笔比特币交易的输

入可以有来自多个交易输出的 UTXO，但必须是从用户拥有地址中可用的 UTXO 中创建出来的。

用户不能把 UTXO 进行进一步的细分，因此用户钱包在用户发起交易的时候就自动会从后台储存的地址中拼凑出一个大于或等于一笔交易输入的比特量，故大多数比特币都会产生找零。例如 Alice 想要购买 Bob 售卖的标价为 5 比特币的比萨，钱包自动计算 Alice 现有的 UTXO，发现在之前三个交易的地址分别各有 2 比特币，因此这笔交易的输入就是 2＋2＋2 比特币，交易的输出对象除过支付给 Bob 的 5 比特币外，还需要标注将找零归还给 Alice，如果不加找零的归属，那么系统就会将找零当作奖励送给矿工。

总的来说，比特币交易的交易输入是即将被交易消耗的 UTXO，交易输出的是该笔交易创建的 UTXO。每个交易输出的地址所代表的 UTXO 是需要所有用户利用数字签名进行解锁的，只有对未使用过的 UTXO 进行签名才是有效的，这样就有效避免了一个 UTXO 被花费两次或多次。

另外，时间戳将交易记录通过时间顺序连接起来，发生在前的交易未被纳入主链时，之后的交易是没有办法发生的。UTXO 和时间戳技术的结合，能够有效避免双花问题。

此外，节点创建交易后会将交易信息进行广播，其他节点会对交易内容进行独立的校验。校验过程中的两条准则会避免双花问题的出现：

(1) 对于每一个输入，引用的输出即 UTXO 必须是存在且没有被花费的。

(2) 对于每一个输入，矿工会检索自己所有的该用户的区块链数据并聚合，如果引用的输出即 UTXO 存在于区块链或交易池中的任何交易，该交易会被拒绝。

3.3.2　数字签名

比特币网络中利用椭圆曲线数字签名算法对交易进行签名，椭圆曲线数字签名算法(ECDSA)是使用椭圆曲线密码(ECC)对数字签名算法(DSA)的模拟。

根据椭圆曲线相乘的加密算法，私钥是一个随机产生的随机数，假设用 d 表示；公钥是私钥与加密算法中定义的固定点 G 按 ECC 定义标准相乘的结果，即 PubKey＝(d，$d\times G$)。对交易信息签名的过程大致包括如下步骤：

(1) 钱包生成一对密钥对(d，$d\times G$)。

(2) 对交易信息做哈希运算，同时将哈希值转换为大端模式(big endian)的整数，即 e＝Hash(message)。

(3) 选择随机数 k，利用 ECC 生成一对坐标值(x，y)，即(x，y)＝$k\times G$。

(4) 令 $r=x$。

(5) 输出数字签名$\left[r,e+\left(\frac{r\times d}{k}\right)\right]$。

大部分的锁定脚本中会包含数字签名，在对交易进行验证的过程中，节点需要完成对签名部分的验证。假定交易中包含的签名为(r，s)，其中 $s=e+\left(\frac{r\times d}{k}\right)$，对交易进行验证的过

程大致包括如下步骤：

（1）对公开的交易信息做哈希运算，即 $e=\mathrm{Hash}(\mathrm{message})$。

（2）计算 $u_1=e\times s^{-1}$，$u_2=r\times s^{-1}$。

（3）计算 $k\times G=u_1\times G+u_2\times(d\times G)$，如果结果为零点，则签名无效。

（4）令 $(x,y)=k\times G$，若 x 等于 r，则签名有效，反之签名无效。

ECDSA 算法按理只能单向计算，无法根据签名和公钥暴力破解出用户私钥，但是如果签名过程中随机数被多次使用，那么根据公开的签名信息和公钥，就能够反推出私钥，对用户账户的安全而言是致命性打击。具体推导过程如下：

（1）对于不同交易签名时，重复使用签名过程步骤 3 中的随机数 k，ECC 中生成点 G 为固定值，得到坐标值相等：

$$\left.\begin{aligned} k_1 &= k_2 \\ (x_1,y_1) &= k_1\times G \\ (x_2,y_2) &= k_2\times G \end{aligned}\right\}\rightarrow x_1=x_2,y_1=y_2$$

（2）令 $r_1=x_1$，$r_2=x_2$。

（3）得到签名 sig_1 和 sig_2，其中 $\mathrm{sig}_1=e_1+\left(\frac{r\times d}{k}\right)$，$\mathrm{sig}_2=e_2+\left(\frac{r\times d}{k}\right)$。

（4）根据公开的签名信息 sig_1 和 sig_2，以及交易哈希 e_1 和 e_2，就能够得出随机值 k，即

$$\left.\begin{aligned} \mathrm{sig}_1&:\left[r,\left(e_1+\frac{r\times d}{k}\right)\right] \\ \mathrm{sig}_2&:\left[r,\left(e_2+\frac{r\times d}{k}\right)\right] \end{aligned}\right\}\rightarrow k=\frac{e_1-e_2}{\frac{e_1+r\times d}{k}-\frac{e_2+r\times d}{k}}$$

（5）根据 k 值，能够推出用户私钥 d，即

$$d=\frac{\frac{e_1+r\times d}{k}\times k-e_1}{r}$$

式中，$\frac{e_1+r\times d}{k}$为公开的数字签名；e_1 为交易摘要哈希，如果反复利用 k，那么用户账户私钥有极大可能会被破解，造成不可估量的损失。

3.3.3 交易脚本

比特币交易验证依赖于锁定脚本和解锁脚本这两类脚本，其中锁定脚本代表了花费条件或者说需要解除的障碍，只有满足条件的用户才能无障碍地进行交易输入量中的 UTXO 的花费。解锁脚本就是这个条件的答案，它允许交易新产生的 UTXO 被花费。锁定脚本中常会包含比特币地址，以及利用私钥产生的数字签名，但并不是所有解锁脚本都一定包含数字签名。

节点通过堆栈形式执行锁定脚本和解锁脚本来验证一笔交易，首先检索交易输入的 UTXO，UTXO 中就包含了定义花费条件的锁定脚本，之后会验证解锁脚本，并执行。但实

际上，本次交易输入的 UTXO 是上一笔交易产生的输出，可以用一系列单输入/单输出交易对该过程进行模拟，如图 3-19 所示：当前交易的输入都引用了前一笔交易的输出。

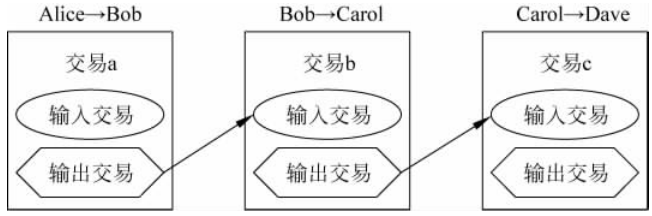

图 3-19　一系列单输入/单输出交易示意图

当 Bob 想要利用交易 a 中输出的比特币用于支付 Carol 需要的费用时，就需要解开 Alice 在交易 a 的输出脚本，也就是锁定脚本，Bob 会在交易 b 的输入脚本，即解锁脚本中给出题解。这种类似于题目和答案的关系，在图 3-19 中箭头对应的输入和输出才是真正的一对锁定脚本和解锁脚本。

比特币在交易中使用的是脚本系统，它基于堆栈，从左至右处理，脚本语言被设计为非图灵完备的，没有 LOOP 语句。在交易输入中使用了用于解锁 UTXO 的 Signature Script，在交易输出中使用了锁定脚本 PubKey Script。针对使用的 Signature Script 和 PubKey Script 的不同，可以将比特币网络中的标准交易分为以下几种类型：P2PKH(pay to public key hash)、P2PK(pay to public key)、MS(multiple signature)、P2SH(pay to script hash)。

1. P2PKH

Sigscript：< sig >< PubKey >

PubKeyscript：OP_DUPOP_HASH160 < PubKey > OP_EQUALVERIFYOP_CHECKSIG

以刚才的单输入/单输出系列交易为例，在交易 a 中，Alice 在交易的输出填写了 Bob 的比特币地址(对应公钥哈希)，当 Bob 想要利用交易 a 产生的 UTXO 时，他必须要证明自己拥有该比特币地址对应的私钥，才能解开交易 a 的锁定脚本。因此 Bob 就在交易 b 的输入中加入了自己的公钥和利用私钥产生的签名，P2PKH 的标准交易类型的脚本组合如图 3-20 所示。

<sig><PubKey>	DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
解锁脚本 (scriptSig)	锁定脚本(scriptPubKey)

图 3-20　P2PKH 标准交易类型脚本组合

解锁脚本中主要有两个验证，一个是公钥是否能转换为正确的公钥哈希，一个是签名是否有效。如果两个验证均成功，也就是脚本被执行有效时，输出结果为 OP_TRUE。

利用堆栈形式表示的具体流程大概如下：

(1) 输入解锁脚本，因为脚本按照堆栈形式从左至右执行，那么先入栈的是签

名< sig >，随后是公钥< PubKey >，如图 3-21 所示。

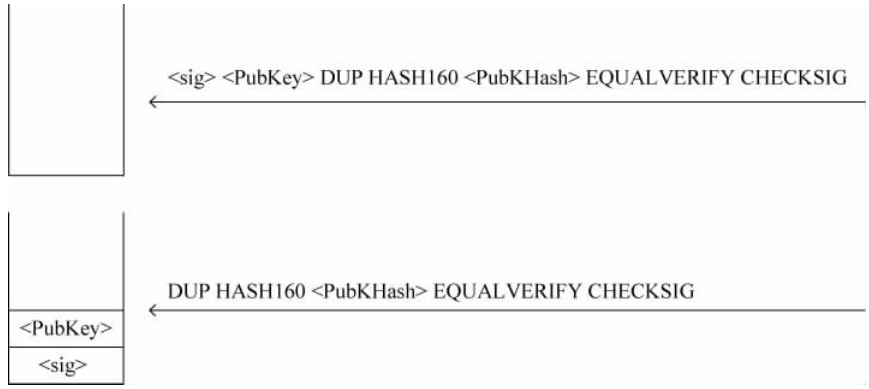

图 3-21 < sig >< PubKey >入栈示意图

（2）执行解锁脚本，输入第一个指令 OP_DUP（复制栈顶元素操作），将< PubKey >复制并存储，如图 3-22 所示。

图 3-22 OP_DUP 操作示意图

（3）OP_HASH160：对栈顶元素< PubKey >进行哈希 160 计算，得到了公钥哈希< PubKHash >，如图 3-23 所示。

图 3-23 OP_HASH160 操作示意图

（4）将解锁脚本中的公钥哈希字段< PubKHash >入栈，表示为< PubKHash' >，如图 3-24 所示。

（5）OP_EQUALVERIFY：判断栈顶两个元素是否相等，如果相等，则继续执行，否则中断执行，返回失败信息，如图 3-25 所示。

（6）OP_CHECKSIG：执行签名校验操作，判断签名信息和公钥是否匹配，如果相等，则返回成功，锁定脚本和解锁脚本匹配，能够执行花费操作；否则返回失败，如图 3-26 所示。

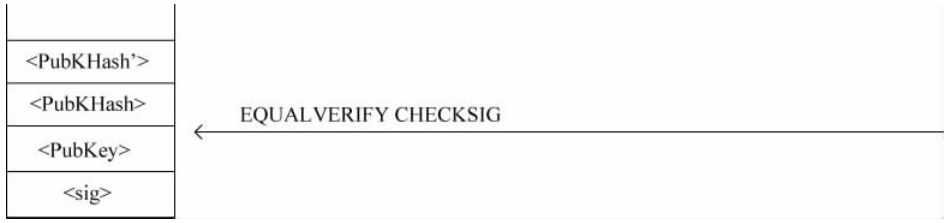

图 3-24　< PubKHash >入栈示意图

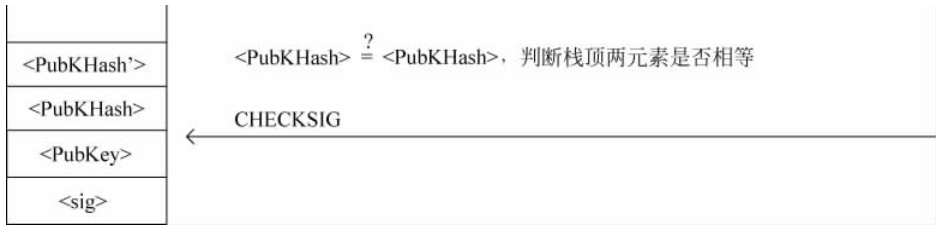

图 3-25　OP_EQUALVERIFY 操作示意图

图 3-26　OP_CHECKSIG 操作示意图

2. P2PK

Sigscript：< sig >

PubKeyscript：< PubKey > OP_CHECKSIG

对比 P2PKH，P2PK 简化了验证过程，只剩一步验证签名过程，少了地址验证的步骤。P2PK 的核心仍然是 P2PKH，其被创建的目的主要是使得验证简洁更加方便使用。

3. MS

Sigscript：OP_0 < sig1 >< sig2 >...< sig$m$ >　//OP_0 为占位符，无实际意义；m 为激活交易需要的最少公钥数

PubKeyscript：M < PubKey1 >< PubKey2 >...< PubKey$n$ > N OP_CHECKMULTISIG

//n 为存档公钥总数

多重签名表示一个账户对应多个密钥，如果想要使用该账户对应的比特币，则需要有多个签名才能够完成。M-N 多重签名中，N 指的是存档的公钥总数，也就是对应的密钥对数目；M 是要求激活交易的最少公钥数，其中需要满足 $N \geqslant M$。

通用的多重签名锁定脚本的形式为：

M < Public Key 1 > < Public Key 2 >...< Public Key *N* > *N* OP_CHECKMULTISIG

M 和 *N* 可以被设定，假设 *M* 取 2，*N* 取 3，即某个地址所代表的比特币有三个关联的密钥，只有拥有其中的两个或两个以上的签名时才能花费这个比特币，这种情况下的 2-3 多重签名条件可以表示为：

2 < PublicKeyA >< PublicKeyB >< PublicKeyC > 3OP_CHECKMULTISIG

在具体验证过程中，每个公钥地址都需要被验证，譬如说在 2-3 多重签名中，验证脚本可以表示为：

OP_0 < SignatureB >< SignatureC > //OP_0 为占位符，无实际意义

M-*N* 的组合可以自由设定，如 1-3(存档公钥数为 3，激活交易需要至少 1 个签名)、3-3(存档公钥数为 3，激活交易需要 3 个签名)或者 4-5(存档公钥数为 5，激活交易需要至少 4 个签名)等，但比较常见的还是 2-3(存档公钥数为 3，激活交易需要至少 2 个签名)多重签名。

多重签名对保护用户的账户安全有着深刻的意义。用户的私钥虽然不能被暴力破解，但还是存在被黑客攻击盗用的风险。多重签名保证了在用户的某个私钥丢失或者被盗的情况下账户中比特币的安全。

另外，多重签名在资产能够安全化管理方面有着重要作用，尤其是在暴露私钥的交易中，有效提高了账户的安全系数。多重签名代表在只有几方共同确认的情况下，账户资金才能被动用，这也有着广泛的实用场景，譬如说电子商务、财产分割、资产监管等方面。

4. P2SH

Sigscript：[signature]{[PubKey]OP_CHECKSIG} //{[PubKey]OP_CHECKSIG}即 redeemScript 代码

PubKeyscript：OP_HASH160[20-byte-hashof{[PubKey]OP_CHECKSIG}]OP_EQUAL

P2SH 是 MS 多重签名的简化版本，在 BIP16 中进行了具体阐述，主要目的是容许发送者构造更加丰富的交易类型，其次是对锁定脚本使用了 SHA-256 哈希算法，将制作脚本的责任给了接收方，暂缓节点的存储压力。可以理解为只要提供一段 script，当它的二进制哈希与目标匹配的情况下，款项就能够被使用。

利用堆栈形式表示大致过程如下：

(1) 输入签名信息< sig >，公钥信息< PubKey >和运算符 OP_CHECKSIG 统一存储，如图 3-27 所示。

(2) OP_HASH160：对栈顶元素{[PubKey]OP_CHECKSIG}进行哈希 160 运算，完成运算后保存哈希计算后的结果 Hash160({[PubKey]OP_CHECKSIG})，如图 3-28 所示。

(3) OP_EQUAL：将运算结果 Hash160({[PubKey]OP_CHECKSIG})与目标哈希值[20-byte-hashof{[PubKey]OP_CHECKSIG}]进行匹配，如果不相等，则签名无效，如果相等则进入下一步。

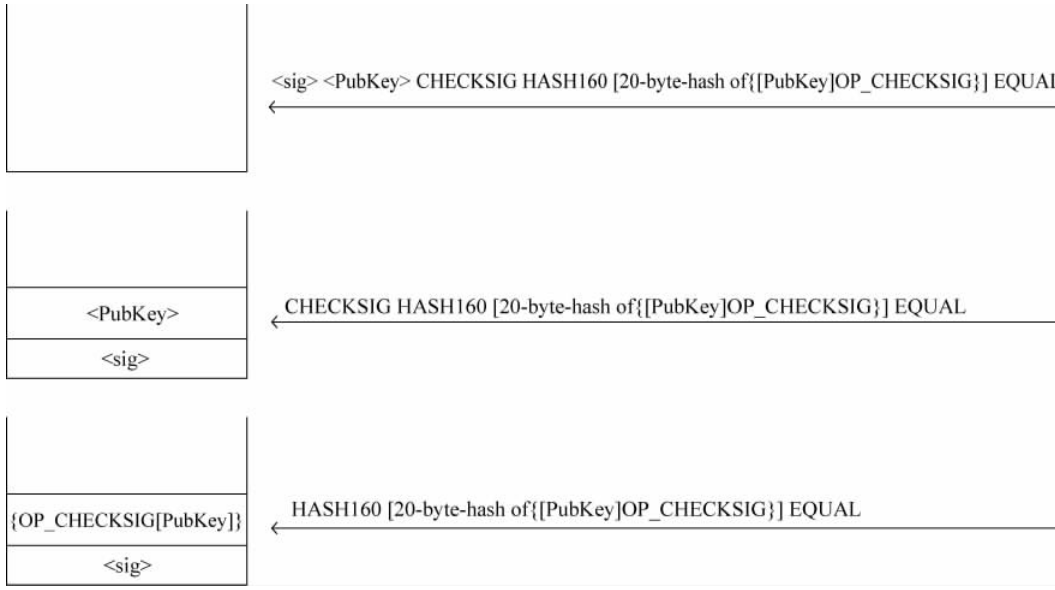

图 3-27　< sig >< PubKey >和操作符 CHECKSIG 入栈示意图

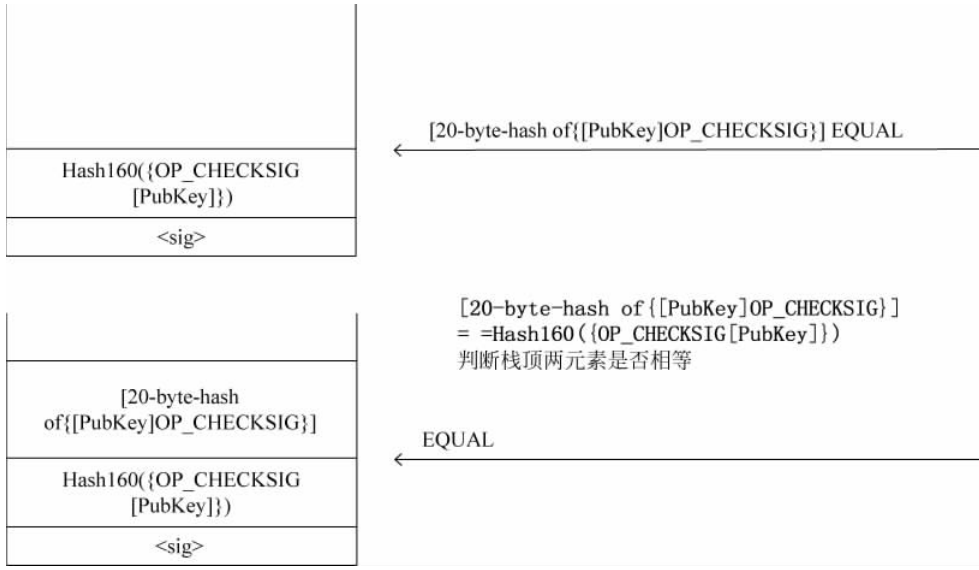

图 3-28　操作符 HASH160 入栈示意图

（4）之后的验证步骤与多重验证相同。

P2SH 的验证步骤有一大部分与多重验证相同，需要注意的是：这是两个完全不同的交易类型，但可以利用 P2SH 来实现 MS 多重验证，P2SH 的意义是拓展了现有的交易类型，实现了脚本简单易扩展的特性。

3.3.4 交易结构

创建一笔交易，需要明确本次交易的输入和输出，但实际上用户在使用钱包进行比特币交易的过程中，只需要对方的地址即可，客户端会在后台自动补充剩余的内容，具体交易的结构并不需要展现在用户面前，如图 3-29 所示，实际比特币的交易结构大致可以分为以下几个部分：交易版本信息、输入交易的数量、输入交易的数组、输出地址的数量、输出交易的数组和锁定时间。

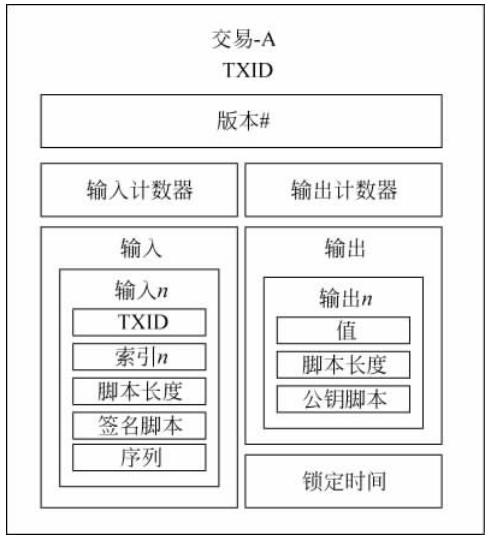

图 3-29　交易结构示意图

版本信息明确了该笔交易参照的规则，数组存储比特币地址，输入交易的数组存储的是上一笔交易产生的 UTXO 的地址，输出交易的数组表示该笔交易产生的 UTXO 存储地址。一个交易可以包含多个输入和输出，具体取决于每笔交易的发起人，而该笔交易的比特币来源于多少个交易产生的 UTXO、输出给多少个地址，交易的数据结构中都会有表明的字段。交易锁定时间(lock_time)是一个多意字段，表示在某个高度的区块之前或者某个时间前，该交易处于锁定状态，定义了该笔交易能够被添加到区块链中的最早交易时间，相当于将一张纸质支票的生效时间予以延后，包括如下三种情况：

(1) lock_time=0，表示交易立即生效，无延迟效果。

(2) lock_time$<5\times10^8$，定义为区块高度，处于该区块之前为锁定状态，交易不生效。

(3) lock_time$\geq5\times10^8$，定义为 UNIX 时间戳，处于该时刻之前为锁定状态，交易不生效。

在输入交易中有一个字段为“索引”，因为引用的输入交易中可能有多个输出，所以索引字段表示引用的 UTXO 为上一笔交易的第几个输出。在具体源代码中，hex 字段包含所有交易信息，解析之后可得到各个字段的信息。

```
"0100000003c9f3b07ebfca68fd1a6339d0808fbb013c90c6095fc93901ea77410103489ab7000000008a47
```

```
3044022055bac1856ecbc377dd5e869b1a84ed1d5228c987b098c095030c12431a4d5249022055523130a9d
0af5fc27828aba43b464ecb1991172ba2a509b5fbd6cac97ff3af0141048aefd78bba80e2d1686225b755da
cea890c9ca1be10ec98173d7d5f2fefbbf881a6e918f3b051f8aaaa3fcc18bbf65097ce8d30d5a7e5ef8d100
5eaafd4b3fbeffffffffc9f3b07ebfca68fd1a6339d0808fbb013c90c6095fc93901ea77410103489ab70100
00008a47304402206b993231adec55e6085e75f7dc5ca6c19e42e744cd60abaff957b1c352b3ef9a022022a
22fec37dfa2c646c78d9a0753d56cb4393e8d0b22dc580ef1aa6cccef208d0141042ff65bd6b3ef04253225
405ccc3ab2dd926ff2ee48aac210819698440f35d785ec3cec92a51330eb0c76cf49e9e474fb9159ab41653
a9c1725c031449d31026affffffffc98620a6c40fc7b3a506ad79af339541762facd1dd80ff0881d773fb72
b230da010000008b483045022040a5d957e087ed61e80f1110bcaf4901b5317c257711a6cbc54d6b98b6a85
63f02210081e3697031fe82774b8f44dd3660901e61ac5a99bff2d0efc83ad261da5b4f1d014104a7d1a57e
650613d3414ebd59e3192229dc09d3613e547bdd1f83435cc4ca0a11c679d96456cae75b1f5563728ec7da1
c1f42606db15bf554dbe8a829f3a8fe2fffffffff0200bd0105000000001976a914634228c26cf40a02a05d
b93f2f98b768a8e0e61b88acc096c7a6030000001976a9147514080ab2fcac0764de3a77d10cb790c71c74c
288ac00000000"
```

下面将 hex 字段逐个分解，就能看到一笔交易中需要的每个详细字段的信息：

```
    01000000        //版本号反序,UINT32
    03              //Tx 输入数量即几笔输入,变长 INT.3 个输入

    /* * * 第一组 InputTx * * * /
    //TxHash,固定 32 字节
    c9f3b07ebfca68fd1a6339d0808fbb013c90c6095fc93901ea77410103489ab7
    00000000        //消费的 Tx 位于前向交易输出的第 0 个"索引",UINT32,固定 4 字节
    8a              //签名脚本的长度,0x8A = 138 字节
    //138 字节长度的签名,含有两个部分:公钥 + 签名
    47              //签名长度,PUTDATA47,将 47 字节压入栈中,0x47 = 71 字节

3044022055bac1856ecbc377dd5e869b1a84ed1d5228c987b098c095030c12431a4d5249022055523130a9d
0af5fc27828aba43b464ecb1991172ba2a509b5fbd6cac97ff3af
    01              //SIGHASH_ALL 指令
    41              //公钥长度,0x41 = 65 字节

048aefd78bba80e2d1686225b755dacea890c9ca1be10ec98173d7d5f2fefbbf881a6e918f3b051f8aaaa3fc
c18bbf65097ce8d30d5a7e5ef8d1005eaafd4b3fbe
    ffffffff        //sequence,顺序编号,0xffffffff = 4294967295,UINT32,固定 4 字节

    /* * * 第二组 InputTx. 与上同理,省略分解 * * * /
c9f3b07ebfca68fd1a6339d0808fbb013c90c6095fc93901ea77410103489ab7010000008a47304402206b9
93231adec55e6085e75f7dc5ca6c19e42e744cd60abaff957b1c352b3ef9a022022a22fec37dfa2c646c78d
9a0753d56cb4393e8d0b22dc580ef1aa6cccef208d0141042ff65bd6b3ef04253225405ccc3ab2dd926ff2e
e48aac210819698440f35d785ec3cec92a51330eb0c76cf49e9e474fb9159ab41653a9c1725c031449d3102
6affffffff

    /* * * 第三组 InputTx * * * /
c98620a6c40fc7b3a506ad79af339541762facd1dd80ff0881d773fb72b230da010000008b483045022040a
5d957e087ed61e80f1110bcaf4901b5317c257711a6cbc54d6b98b6a8563f02210081e3697031fe82774b8f
```

```
44dd3660901e61ac5a99bff2d0efc83ad261da5b4f1d014104a7d1a57e650613d3414ebd59e3192229dc09d
3613e547bdd1f83435cc4ca0a11c679d96456cae75b1f5563728ec7da1c1f42606db15bf554dbe8a829f3a8
fe2fffffffff

    02              //Tx 输出数量,变长 INT.两个输出

    /* * * 第一组输出 * * * /
    00bd010500000000  //输出的币值,即交易的数额,UINT64,8 字节.字节序需翻转,
                      //~ = 0x000000000501bd00 = 84000000satoshi
    19                //输出目的地址字节数,锁定脚本大小,0x19 = 25 字节,由一些操作码与数
                      //值构成
    //目标地址
    //0x76 -> OP_DUP(stackops)
    //0xa9 -> OP_HASH160(crypto)
    //0x14 ->长度,PUTDATA14 压入栈中,0x14 = 20 字节
    76a914
    //地址的 HASH160 值,20 字节
    634228c26cf40a02a05db93f2f98b768a8e0e61b
    //0x88 -> OP_EQUALVERIFY(bitlogic)
    //0xac -> OP_CHECKSIG(crypto)
    88ac

    /* * * 第二组输出 * * * /
    c096c7a603000000
    19
    76a9147514080ab2fcac0764de3a77d10cb790c71c74c288ac

    00000000        //lock_time,UINT32,固定 4 字节
```

Hex 字段中涉及签名的类型,目前签名共有三类,分别是 SIGHASH_ALL、SIGHASH_NONE、SIGHASH_SINGLE。SIGHASH_ALL 为默认类型,也是目前绝大多数交易采用的,也就是对整单交易都进行签名确认;SIGHASH_NONE 最自由松散,交易发起者只对输入进行签名,不对输出签名,输出可以随意指定,意思就是只想花费这笔钱,具体怎么花让谁花并不关心;SIGHASH_SINGLE 是只对自己输入/输出签名,假设单子里有其他交易者的输入/输出,那么只关心自己这笔钱,同意花费,并指定输出,其他的并不关心。

3.4 比特币共识机制

比特币的区块链技术作为分布式数据库的一种展现形式,不同独立节点需要面对在无第三方中心化机构的情况对数据达成一致,称为去中心化共识。共识是数以千万计的独立

节点分别遵循相同的规则通过异步交互自发形成的产物，比特币的去中心化共识就是由所有网络节点相互作用而形成的。为了保证不同节点存储的交易信息真实完备并且在多数情况下能够全网一致，比特币共识可以简单描述为四个过程：

（1）每个全节点依据标准对广播至网络的每笔交易都要进行独立验证。

（2）矿工节点通过算得工作量证明解，独立地将网络中的交易记录打包进新区块。

（3）节点依照标准对广播至网络的新区块进行独立校验并组装进本地存储的区块链中。

（4）对于可能存在的区块链分叉情况，每个节点依照工作量证明机制独立选择累计工作量最大的区块链。

用户想要执行一笔交易，客户端确保钱包交易合法的情况下，交易会被广播至临近节点，每一个收到交易信息的节点会独立对交易进行验证，只有节点确保交易有效的情况下，节点才会继续对交易进行广播，无效的交易会在第一轮广播至临近节点被判定无效后当即废弃，确保了不会浪费网络的剩余资源。

节点并不会只收到一笔交易，所有交易都需要进行全网广播和验证，节点会对未被纳入区块链的有效交易进行存储，矿工节点会按照接收顺序对交易进行排序统一存储，可以把存储这些交易的地方称为“交易池”。矿工节点在打包整理新区块的时候，新区块中的交易记录便来自于交易池。等待确认的交易数量是较为庞大的，矿工节点会给交易记录设置优先级，优先级高的交易记录会更先被选择纳入新区块中。优先级主要由块龄决定，块龄指的是交易输入所花费的 UTXO 被记录到区块链为止所经历的区块数，也就是这笔 UTXO 在区块链中的深度。交易输入值高、块龄大的交易拥有更高的优先级，在纳入新区块的时候将会被优先考虑。但是即使有效交易被存储到交易池中，也会存在丢失的情况。如果一笔交易只被广播了一次，那么就只会存储在最邻近的一个矿工节点的交易池中，因为交易池是以未持久化的方式保存在矿工节点的存储器中的，如果节点重新启动，交易池中的数据就会被完全擦除，所以如果一笔交易长时间未得到确认，那么就会从挖矿节点的交易池中消失。

比特币采用工作量证明（PoW）机制，矿工节点每时每刻都在进行算力竞争，在完成交易内容的验证后，矿工节点会从交易池中将所有未被纳入区块确认的交易按照 Merkle 树的形式整合到一个候选区块中，根据候选区块的区块头计算工作量证明解。每个挖矿节点都会各自独立地进行相同的工作，假如有两个不同的节点同时算得新区块的工作量证明解，这两个区块中包含的交易记录大部分是相同，可能在排列顺序上有所差别。当一个矿工节点算得工作量证明，赢得新区块的算力竞赛，该节点会将新区块广播至临近节点，其他在算力竞争中失败的区块会接受广播，并独立对新区块进行验证。如果新区块通过验证，这些节点会放弃之前对构建这个相同高度区块的计算，同时检查交易池中的全部交易，将已经被纳入新区块中的交易记录从交易池中移除，利用交易池剩下的交易构造下一个区块的候选区块。另外，节点将新区块添加到自身节点的区块链副本中，完成对新区块的传播和自身区块数据的更新。

具体的校验包括：区块的数据结构语法上有效；区块头的哈希值小于目标难度（确认

包含足够的工作量证明)；区块时间戳早于验证时刻未来两个小时(允许时间错误)；区块大小在长度限制之内；第一个交易(且只有第一个)是 coinbase 交易；使用检查清单验证区块内的交易并确保它们的有效性。每个节点都会完成的独立校验过程确保了只有有效的区块才能在网络中进行传播,无效区块在第一次广播后就会被丢弃。这样确保了只有诚实的矿工生成的区块才会被纳入区块链从而获得挖矿奖励,无效区块无法被加入区块链,生成无效区块的作弊行为将白白浪费电力,从而鼓励矿工节点进行诚实的工作。

比特币去中心化共识机制的最后一步是区块会集合到有最大工作量证明的链中。任何时候,网络中的主链都是已知累积了最多难度的链条,一般情况下,主链也是包含了最多区块的链条,除非有两个等长度的链,而其中一条累积了更多的工作量证明。最长链选择规定解决了区块链网络可能存在的临时性分叉造成的差异,只要所有节点都遵循该原则,整个比特币网络最终都会收敛到一致的状态。

3.5 比特币账户安全

比特币作为一种虚拟的数字货币,实际上是通过网络节点存储的交易记录来表示的,因此保护用户的比特币安全,应当从网络抗攻击能力和用户自身安全意识两方面入手。一方面需要比特币网络提高鲁棒性,另一方面是用户保证自己钱包私钥的隐秘性。

相对比传统的金融机构,譬如说银行,需要依赖第三方进行不良用户的审核和排除。在对所有人开放的去中心化的比特币网络中,系统则将责任和控制权都移交给了用户,网络的安全性依靠的是工作量证明而非人为的记录控制。如果私钥丢失或被破解,不存在第三方机构能够执行找回服务,因此在比特币的使用中,用户需要更多地留意如何保管好自己的账户私钥。

私钥一般存储在用户的钱包客户端中,对于钱包应用的核心问题是如何安全方便地生成、保存和备份恢复密钥。助记词帮助使用者方便记忆和复制钱包,多重签名增加了账户的安全性,硬件钱包和物理存储将私钥保管在隔绝网络的环境下,使黑客攻击的可能性降为零,这些方法都是为保证用户的比特币安全而做出的努力。多重签名已经在交易脚本中做了仔细阐述,本节主要讲解助记词、硬件钱包和物理存储。

3.5.1 助记词和种子文件

私钥的本质是随机产生的固定长度(64 位)的十六进制数组,不便于记忆和复制。助记词相当于一种明文私钥,通过固定词库,将 64 位的私钥对应为不同的英文单词,方便使用者进行复制。助记词只是私钥的一种不同的表现形式,但是相比完全是随机数字顺序来说,助记词更容易被读出来并且正确地抄写。

助记词由钱包使用 BIP-39 中定义的标准化过程自动生成，这个过程如图 3-30 所示。

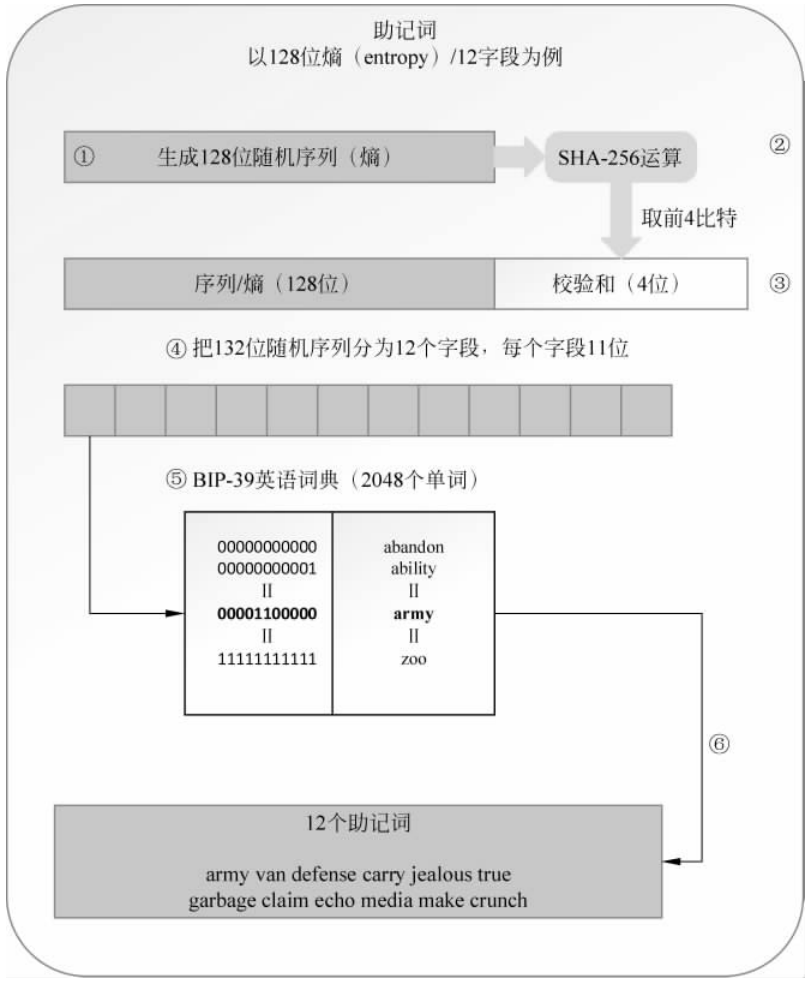

图 3-30　助记词生成过程

可以简要地描述如下：

(1) 创建一个 128 位的随机序列/熵(entropy)。

(2) 对随机序列做 SHA-256 运算，取前几位(熵/32)做随机序列的校验和。

(3) 将校验和(checksum)添加到随机序列末尾。

(4) 对随机序列进行划分，分为 12 个字段，每字段长度 11 位。

(5) 将每个包含 11 位的数组与预先定义好的单词字典做对应，该单词字典在 BIP-39 中规定，总共 2048 个单词。

(6) 生成的有顺序的单词组就是助记词。

生成助记词的过程从熵源开始，增加校验和，然后将随机序列映射到单词列表。根据熵

源的长短不同，助记词的长度也会不同。

如前面所说，钱包是用于发送和接收比特币的客户端，是私钥的容器，实现了对密钥的存储和管理。根据私钥的生成方法，钱包可以分为非确定性钱包和确定性钱包。非确定性钱包是一堆随机生成的私钥的集合，钱包中的多个私钥相互独立，两个私钥之间是没有关联的。这种钱包难以管理和备份。如果生成很多私钥，则必须保存它们所有的副本。这就意味着这个钱包必须被经常性地备份。每个私钥都必须备份，否则一旦钱包不可访问时，无法找回钱包。确定性钱包则不需要每次转账都要备份，确定性钱包的私钥是对种子进行单向哈希运算生成的，种子是一串由随机数生成器生成的随机数。在确定性钱包中，只要有这个种子，就可以找回所有私钥，只需备份种子就相当于备份所有钱包，所以这个种子也相当重要，一定要备份到安全的地方。种子能够收回所有的已经产生的私钥，种子也能够使钱包完成输入或者输出，使得使用者的私钥能够在钱包之间轻松转移输入，允许使用者去建立一个公共密钥的序列而不需要访问相对应的私钥。

确定性钱包和升级后的分层确定性钱包，在生成密钥的过程中都需要种子文件。由助记词生成种子，需要使用私钥抻拉函数 PBKDF2，这个函数有两个参数，第一个参数是私钥生成的助记符，第二个参数是“盐”(Salt)，盐值＝字符串常数“助记词”＋(可选)用户提供的密码字符串。PBKDF2 抻拉函数利用 HMAC-SHA512 算法，进行 2018 次哈希运算来延伸助记符和“盐”参数，最终产生一个 512 位的值作为最终的输出，这个输出就是钱包需要的种子文件。

3.5.2 硬件钱包和物理存储

比特币所采用的区块链技术虽然避免了常见的信用卡盗用等情况，但是账户安全程度极大地依赖用户对于私钥的保管。对于大多数用户来说，保管好私钥是一件很不容易的事情。互联网的普及，让用户单纯将私钥保存在钱包里的行为变得危险。常用的计算机、手机上运行着大量的第三方软件，这些软件往往不受约束地访问本地存储的各类文件，只要有一个恶意软件，那么就存在威胁比特币密钥存储安全的可能性。同时，比特币立即转移给他人的操作无法撤回，被窃取后比特币就无法被追回，盗取者本身也不需要身份掩饰和资金洗白，另外，比特币具有高昂的市场价值，这些都为黑客提供了强烈的作案动机。要想在联网的设备中进行数据的存储，使用者需要一定的计算机技巧才有可能保证账户私钥不因病毒或其他类似原因被窃取。显然，这对大部分普通用户来说很困难。

因此，将虚拟的数据存储转换为物理实体，将比特币私钥转换为物理形式，更符合人们平时对于资产的保存习惯。可以将比特币私钥写在纸上或者印刻在其他物体上，这样保护私钥就代表着简单的保护带有私钥信息的物理实体。其中，“纸钱包”就是将私钥印在纸张上进行存储。比特币的离线保存称为冷存储，指的是在一个从未连接过互联网的离线系统上生成私钥，并离线存储在纸上或其他媒介，是最有效的安全技术之一。

硬件钱包也十分常见，硬件钱包是指将比特币的私钥单独存储在一个芯片中，目的也是与互联网隔离，实现了即插即用。硬件钱包只提供一个接口，为普通的非专业用户提供了很

高安全等级的密钥存储方法。

但是实际上，无论哪种存储手段都会存在风险。为了规避风险，建议不要在一个账户中存放过多的比特币，用于资金流动的比特币账户可以存储在在线钱包中，把大量的比特币存储在离线系统中是比较稳妥的，做好最坏的打算也会将自己的损失降到最低。

3.6　比特币扩容方案

在比特币诞生之际，中本聪并未严格限制区块的大小，按照比特币的数据结构规则，一个区块最大可达到 32MB。在比特币初始阶段，平均被打包的区块大小只有 1～2KB，远远没有到达区块的上限值，因此造成了资源浪费，同时也容易发生分布式拒绝服务攻击。为了保证比特币系统的安全和稳定，中本聪才将区块大小限制在 1MB。

按照每笔交易占 250B，平均每 10 分钟产生一个区块的速度计算，比特币区块链网络理论上每秒最多可以处理 7 笔交易。但是随着用户体量的增大，交易迟迟不能得到网络确认，网络拥堵现象严重，而用户为了让矿工节点将自己的交易打包过程前提，只能增加交易费，但是每秒 7 笔的处理速度根本无法满足用户的需求。交易积压问题日益严重，最高时有上万笔交易有待确认，比特币网络扩容问题迫在眉睫。

3.6.1　比特币扩容之争

针对扩容问题，不同的用户群体有不同的意见。分歧主要分为两派：希望保持比特币小区块特性的 core 开发组和拒绝使用闪电网络等弱中心化操作的矿工及支持矿工的开发者。

中本聪在退出比特币社区之前将代码维护工作交给了 core 开发组，最开始 core 开发组的核心是被称为中本聪继承人的 Gavin Andresen。但是比特币社区的意见并不完全由 core 开发组主宰，根据比特币网络的特性，交易的确认需要通过矿工节点的算力竞争实现，而随着挖矿成本的提高，非专业设备的普通计算机和个人基本不可能成功挖矿，只有大量专业矿机集体作用才能成功，也就是矿池。在 2017 年之前，中国的数个大矿池集中了超过全网 90％的算力。矿池公司代表了矿工利益，因为除非绝大部分用户一致同意改变目前的挖矿方式，否则负责维护区块链的矿工拥有是否将全网升级到某个新版本的决定权，如果矿工强硬拒绝，即使用户再希望使用新版本，也无可奈何。扩容之争在矿池公司和 core 开发组之间展开。

core 开发组一直希望比特币保持小区块，提出采用隔离见证和闪电网络的方式解决比特币区块链拥堵的问题。一方面能够保证区块链的交易速度和安全性，另一方面能够防止矿工权力过大导致比特币的中心化。但实际上，在 2017 年 5 月，维护全网安全的记账矿工

就已经不超过 25 个了，并且前 5 名的算力也已经超过了 51%，比特币区块链的中心化已经发生了。

但是隔离见证和闪电网络的形式却极大地损失了矿工的利益。隔离见证在不扩展区块容积的情况下，将比特币交易过程中的签名字段和交易内容分开，将一个比特币交易分成“交易状态”和表明交易合法性的“见证”，将见证即签名信息隔离出来。UTXO 中存储一个指向签名信息的指针，因为一般用户只需要说明交易结余情况的交易过程信息，只有需要验证的矿工节点才需要完整信息，通过隔离见证就能存储更多的交易内容，这就实现了区块的变形扩容。而闪电网络的目的是给用户提供可以在链下进行交易的双向支付通道，在比特币区块链现有基础上搭建一个二层支付网络，鼓励小额交易在二层网络上进行，只有大额的区块链的交易才会在主链中被确认。通过闪电网络的支付通道能够缓解比特币主链的拥堵压力，而将大量小额交易转移到链下，需要矿工直接处理的交易数量也得到了控制和缩减。随着比特币挖矿产生的奖励越来越少，矿工的主要收益逐渐来自处理交易过程交易费累计，而这两部分却是矿工大部分的利益来源。

从 2015 年 Gavin Andresen 和 Mike Hearn 将区块大小提高到 8MB 的提议未被 core 成员认可，并被剥夺代码合并权，到 2016 年 bitcoin core 团队拒绝执行“香港共识”，再到“纽约共识”将 core 开发团队排除在外，比特币扩容问题一直是各方利益团体的角逐战场。

3.6.2 比特币扩容协议

到目前为止，就比特币的扩容问题，人们提出了很多版本的“比特币改进协议（bitcoin improvement proposal，BIP）”。BIP 仅仅是提议，因为每个 BIP 的实际执行都牵扯到对比特币源码的改动，具体的扩容方案的达成需要整个比特币社区达成共识。

1. BIP100

2015 年 6 月，由 Bitcoin core 前开发员兼 Bitpay 员工 Jeff Garzik 提出。通过硬分叉，删除静态 1MB 块大小的限制，同时增加一个新的浮动的块大小限制，浮动值为 1MB，上限为 32MB。测试区块链在 2015 年 9 月进行了硬分叉，主链在 2016 年 1 月 11 日进行了硬分叉。改变 1MB 限制的方式，类似于 BIP34，12 000 个区块（三个月）中需要有 90% 的区块支持 BIP100。区块体积上限提升至 8MB，实施前预留时间。而且区块体积上限可变大，也可变小，只要矿工投票达成共识即可，但绝对上限应该设置在 32MB。

2. BIP101（BitcoinXT）

2015 年 6 月，由 Bitcoin core 前首席开发员兼比特币基金首席科学家 Gavin Andresen 提出，他建议将比特币起始区块上限设定为 8MB，然后每两年上限加倍，直至 2036 年区块达到 8GB 上限。触发条件是当最近的 1000 个区块中有多于 750 区块是 BIP101 版本号的区块，就能达到硬分叉扩容的条件，将会有两周的缓冲时间，而生成大区块的时间不会早于 2016-01-1100：00：00UTC。其中 BitcoinXT 就是使用了 BIP101 规则的“分叉”比特币软件，在 BitcoinXT 软件中，另外还包含了 Mike Hearn 的争议规则。

3. BIP102

2015 年 6 月 23 日，Jeff Garzik 又提出 BIP102，他提议将比特币区块一次性增加到 2MB，并再也不改变。并且当支持算力超过 95%时被激活。

4. BIP103

2015 年 7 月 21 日，由 Bitcoin core 开发者、Blockstream 联合创始人 Pieter Wuille 提出，他建议将区块上限设为最近 11 个区块大小的中位数，或者利用代码 GetMaxBlockSize (pindexBlock-> pprev-> GetMedianTimePast())来控制区块的大小，从 2017 年 1 月到 2063 年 7 月，每 97 天调整一次，幅度不超过 4.4%。

5. BIP105

2015 年 8 月 21 日，由 Bitcoin core 开发员 Btc Drak 提出，以 1MB 为起点，每创建一个块，矿工投票决定增加或者减少容量，调整幅度不超过 10%，期望增加区块大小的矿工投票时需要额外提高挖矿的难度。

6. BIP106

2015 年 8 月 24 日，由比特币开发者 Upal Chakraborty 提出，以 2000 个区块为周期决定区块容量扩大两倍或减半。如果 90%的区块达到了上限的 90%，容量扩大两倍；如果 90%的区块小于上限的 50%，则容量减半。

7. BIP109

2016 年 1 月，Gavin Andresen 又提出了 BIP109 方案，区块增加到 2MB，当支持该方案的算力超过 75%时可被激活，同时规定，矿工将区块的版本号设置为 0×10000000 以示支持。

8. BIP141(隔离见证)

2015 年 12 月，由 Ciphrex 的联合创始人兼首席技术官 Eric Lombrozo 与比特币技术爱好者 Johnson Lau 和 BlockStream 的联合创始人 Pieter Wuille 提出，他们都是 Bitcoin core 的开发员。通过移除比特币交易中的签名字段，使得交易记录和签名分开，实现区块大小不变的情况下变相扩容。连续两周内超过 95%的算力在区块数据中发出 bit1 支持信号，则方案激活。

9. BIP148(用户激活软分叉)

由于 BIP141 一直被矿工阵营反对，为了推进隔离见证的升级，2017 年 3 月，由自称 Shaolinfry 的匿名社区成员提出，他建议将由矿工决定是否进行升级更改比特币网络，转向由用户、交易所、支付处理商等来决定。该协议将原本由算力决定的锁定信号交给由全网节点来决定 。约定激活日期为 8 月 1 日，如果约定激活日期前没有激活，升级了 BIP148 的节点将会拒绝没有发送支持信号的区块，产生软分叉。

10. BIP91

为了避免在 8 月 1 日出现比特币分叉的局面，2017 年 5 月，由比特币开发者 Blockstream

的支持者 James Hilliard 提出一个兼容性的新方案 BIP91。该协议实质上是一个兼容 BIP141 的 BIP148 方案，但是激活阈值在 80%。如果 80%的算力在持续两天内支发出支持信号，它就会被锁定。该协议可以使得无论通过 BIP91 还是 BIP148 升级后的节点互相兼容，能够同时接收 bit1 和 bit4 的信号。意味着无论 core 阵营支不支持纽约共识，只要纽约共识的签署算力（超过了 80%）支持该方案，那么比特币的分裂就暂时能够被避免。

对于扩容问题，除过 BIP 协议，还有如下这些具体的解决方案。

1. BitcoinXT

2015 年底，20MB 扩容计划落空的 Gavin Andresen 联合开发者 Mike Hearn 提出了将区块大小调整至 8MB 的 BitcoinXT 方案。该方案基于 BIP101 协议将起始块的上限设为 8MB，随着时间的推移，区块上限逐渐提高。但这个方案同样没有获得开发组其他成员的认可。2016 年初 Gavin Andresen 被取消了比特币维护权，Mike Hearn 退出比特币社区。

2. Bitcoin Classic

该方案在 2016 年 3 月份，由前比特币基金董事 Olivier Janssens、Final Hash 首席执行官 Marshall Long 与比特币矿工与开发人员 Jonathan Toomim 提出，他们基于 BIP109 协议，延续了中本聪的思想，在他的代码库基础上将区块大小扩大到 2MB，并获得了 Gavin Andresen 和 Jeff Garzik 等开发者的支持。

该方案需要获得 75%以上算力支持，才能够被激活，激活之后 28 天才会发生硬分叉。但该方案遭到了 BlockStream 等区块链技术开发公司的反对。

3. BIP141＋闪电网络

对于 Bitcoin Classic 方案，Core 团队持反对态度，他们希望坚持主链区块 1MB 大小不变，因此提出采用隔离认证（segwit）＋闪电网络的方案解决比特币交易拥堵的问题。

4. 香港共识

2016 年 2 月，Core 开发者和矿工双方在香港数码港达成协议，实施 BIP141＋硬分叉 2MB，并且限制矿工不能运行 Bitcoin Classic。但是由于 Core 团队参加会议的几个主要开发人员回去后遭到其他人的反对，香港共识被迫中止。

5. Bitcoin Unlimited

该方案提出不给单个区块设立上限，产生新区块后，由矿工通过“紧急共识”作出决策，决定区块大小。

6. Teechan

2016 年 12 月，由伦敦帝国理工学院和康奈尔大学的开发小组提出，该方案建议使用安全硬件进行扩容。

7. 侧链扩容

2015 年，BlockStream 提出开发一个侧链扩容项目。当年 6 月，BlockStream 的首席战略官 SamsonMow 创立 LiquidNetworks 侧链项目，通过创建点对点的侧链网络，达到扩容

目的。

8. 纽约共识(Segwit2X)

2017 年 5 月,Barry Silbert 旗下的数字货币集团(DCG)和包括大型矿池运营商比特大陆(Bitmian)在内的其他 57 家公司共同签署 Segwit2X 扩容方案。该方案将隔离验证激活阈值设为 80%,并以 bit4 作为信号发送方式;在 6 个月内执行一次 2MB 硬分叉扩容。

9. UAHF(用户激活硬分叉)

2017 年 6 月比特大陆发布该方案,开发者增加了一个命令规则集以更改节点软件。这些更改将使得先前无效的区块在 flagday 后生效,更改也无须绝大多数的算力来执行。UAHF 的实质是比特大陆为应对 UASF 的一种紧急预案。

总的来说,比特币扩容的方案可以大致划分为两类:直接改变区块容量或者不改变容量通过其他手段提高交易处理能力。无论哪种方法,都需要经过艰难的达成共识的过程,但为了比特币的可持续发展,扩容是一个必须要面对和解决的问题。

3.6.3　闪电网络

在闪电网络出现之前,虽然比特币社区试图通过区块扩容、隔离见证等方法在一定程度上增加网络的交易处理能力,但实际上这些方式并不能将交易处理能力进行数量级的提升。1MB 的区块体积和 10 分钟左右的出块速度,决定了每秒约 7 笔的交易处理速度,每年可以处理大约 2.2 亿笔交易,但是如果将比特币视作一个全球的结算系统,这些交易量甚至不足以支撑一个城市。而比特币和现有成熟的支付系统对比相差甚远,VISA 每秒的交易处理量是 2.4 万笔,峰值是 5 万笔每秒。如果单纯地只将区块体积扩大,就会造成算力集中而失去比特币网络最初去中心化的构想。交易速度和区块容积已经成为一对不容易调节的矛盾。过于缓慢的交易处理速度导致比特币无法成为即时支付系统,同时对于金额较少的微交易也十分不友好,因为想要尽快地验证交易需要支付一笔昂贵的交易费。

闪电网络跳出了常规的扩容思路,既然在比特币区块链中优化性能十分艰难,在不更改任何基础性规则和协议的前提下,直接把大量的微支付放到链外执行。微支付需要考虑交易处理的速度和成本问题迎刃而解,只将最后的交易结果放在链上公示,也大幅度节约了珍贵的算力。

闪电网络提供了一个可扩展的微支付通道。交易双方如果在区块链上预设有支付通道,双方在支付通道中预存一部分资金,之后的每次交易都是对资金分配方案的重新确定,当交易双方决定终止交易进行提现时,将最终交易公布在区块链上,被最终确认。双方借助闪电网络进行多次、高频、双向的通过轧差方式实现瞬间确认的微支付。交易双方如果没有直接支付通道,只要网络中存在一条连通双方的、由多个支付通道构成的支付路径,闪电网络便可以通过这条支付路径实现资金在双方之间的可靠转移。

闪电网络这种零确认交易的核心概念主要有两个:可撤销的顺序成熟度合约(recoverable sequence maturity contract,RSMC)和哈希的带时钟的合约(hashed timelock

contract，HTLC）。闪电网络的基础是双方之间的双向微支付通道，RSMC 定义了通道的基本工作模式，保证了双方的直接交易能够在链下完成。HTLC 进一步实现了有条件的资金支付，将闪电网络从最简单的 RSMC 进阶到任何两人只要能找到一条由多个支付通道组成的支付路径，就能够完成链下交易。闪电网络建立在 RSMC 和 HTLC 的概念和技术基础上，下面分别进行介绍。

1. RSMC

RSMC 的原理可以类比为准备金机制。交易双方存在一个共同的资金池，交易开始之前双方预存一部分资金，通道负责记录双方对资金的分配方案。任何一次对于资金的重新分配和方案的更新都需要经过双方的签名。

例如 Alice 和 Bob 是一对夫妻，他们之间需要经常进行相互汇款和转账，于是他们选择在闪电网络上创建了一个直接支付通道。首先需要他们创建一个多重签名的钱包并各自存储 3 比特币，钱包可以通过各自的私钥进行访问。他们之间的交易就可以基于这 6 比特币进行无限制的转账。如果 Bob 想要向 Alice 发送 1 比特币，他需要将这个比特币的所有权转给 Alice，资产分配方案只有在两个人都用自己的私钥进行签名才会成立，有任何一方不进行签署，那么交易不会成立。Alice 和 Bob 对该笔转账无异议并进行私钥签名后，Alice 就获得了 4 比特币的实际控制权。如果 Alice 暂时不需要将通道中的属于她的比特币提现，她无须及时更新区块链中关于比特币所有权的记录。因为两人之间存在频繁多次的交易可能，很短时间之后可能 Alice 又需要向 Bob 支付新数额的比特币。当交易双方中的任一方或两方都需要终止通道并动用资金时，假设 Alice 想要进行体现，她就可以向区块链主网络出示双方签字的余额分配方案，如果在一段时间之内 Bob 不提出异议，区块链就会终止通道并将资金按协议转入各自预先设立的地址。如果 Alice 提供的是一个已作废的方案，当 Bob 在允许时间段之内提供证据，Alice 的资金将被惩罚给 Bob。惩处措施进一步保证了交易双方的诚信度。

为了鼓励双方尽可能长久地利用通道进行交易，RSMC 对主动终止通道方给予了一定的惩罚：主动提出终止通道的一方资金到账比对方晚，因此谁发起终止谁就会在一定程度上吃亏。

2. HTLC

RSMC 支持了最简单的无条件支付，HTLC 实现了有条件支付，保障了不存在直接交易支付通道的双方可以通过一条“支付”通道完成。HTLC 本质上为限时转账，通过智能合约，双方约定发起转账的一方先冻结一笔钱，并提供一个哈希值，如果在限定时间内有人能提出一个字符串，使得字符串的哈希值和已知哈希值匹配，则这笔钱转给接收方。哈希值就类似一个接头暗语，只有知道正确答案的人才能拿到情报。

假如 Alice 想要转 0.01 比特币给 Dave，但是她们之间并不存在直接支付通道，Alice 找到了一条通过 Bob 和 Carol 到达 Dave 的支付路径，该路径经由 Alice/Bob、Bob/Carol、Carol/Dave 之间的三条微支付通道构成，具体的交易过程包括如下步骤：

(1) Dave 生成一个密码 R,并通过任何安全方式将 Hash(R)发送给 Alice,Alice 并不知道 R 具体是什么,如图 3-31 所示。

(2) Alice 和 Bob 通过微支付通道商定一个 HTCL 合约,只要 Bob 能够在限定的时间内,假设为 3 天,向 Alice 出示匹配的 R,Alice 就会支付 0.01 比特币给 Bob。如果超出 3 天的限定时间,钱款通过微支付通道退回给 Alice, 如图 3-31 所示。

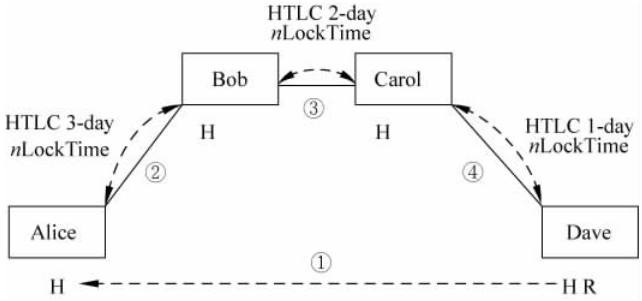

图 3-31　建立支付路径过程示意图

(3) Bob 和 Carol 同样也确定一个 HTCL 协议,限定时间缩短,假定为 2 天,Bob 向 Carol 转账的条件与步骤 2 中相同,Carol 需要向 Bob 出示正确的 R,否则超出限定时间比特币退回给 Bob, 如图 3-31 所示。

(4) Carol 和 Dave 同样商定一个 HTCL 合约：只要 Dave 能在 1 天内向 Carol 出示正确的 R,Carol 支付 0.01 比特币给 Dave,如果 Carol 做不到这一点,钱款自动退回给 Carol,如图 3-31 所示。

(5) 所有合约签订后,Dave 向 Carol 披露正确的 R 成功拿到 0.01 比特币的转账；Carol 知道正确的 R 后向 Bob 出示密码,拿到转账；Bob 知道 R 后向 Alice 出示 R 并拿到他的转账。这样就完成了 Alice 向 Dave 的转账,如图 3-32 所示。

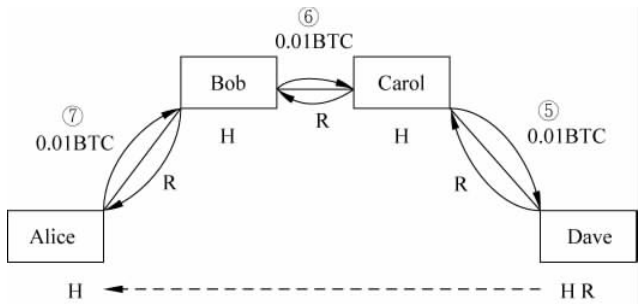

图 3-32　完成支付过程示意图

闪电网络的基础逻辑容易理解,但具体实现的过程十分复杂。从 2015 年闪电网络白皮书发布,到 2018 年第一个闪电网络官方测试版本——Ind 正式发布,闪电网络正在用一种新的方式重新定义比特币。

3.7 小结

比特币引领了数字资产行业的蓬勃发展，其底层的区块链技术也为未来科技的发展创造了一种新的可能。本章首先从使用者的角度介绍了如何加入比特币网络和进行比特币交易，然后介绍了比特币的共识机制和安全机制，最后对比特币面临的扩容问题进行了详细的阐述。