
第1章 初识区块链

当我们坐在飞机上，开启一段美妙的旅程时，可否会想起当初的莱特兄弟；当我们坐在高铁里，享受着高效的都市穿梭时，可否会想起当初的蒸汽机；当我们住在舒适的房屋里，享受着安心的睡眠时，可否会想起当初的茅草房。是的，这个世界给了我们很多原材料，我们使用原材料，制造出了一个又一个工具，通过这种方式，改造这个世界，改善我们的生活。区块链，便是这样的一个改造世界的原材料，而有人，用它制造出了第一个工具，它的名字叫比特币。

本章我们将从区块链的原理及分类、技术组成、技术特点等出发来初步的介绍区块链的概念，并通过对比特币结构的分析让大家有一个感性的认识，作为区块链技术的第一个应用，它的原理设计影响深远。

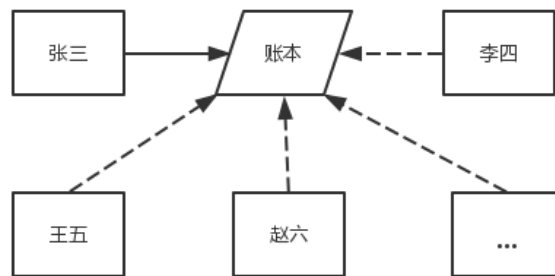
1.1 例说区块链

1.1.1 从一本账本说起

早先的时候，农村里一般都会有个账房先生，村里人出个工或者买卖些种子肥料等等，都会依靠这个账房先生来记账，大部分情况下其他人也没有查账的习惯，那个账本基本就是这个账房先生保管着，到了年底，村长会根据账本余额购置些琐碎物件给村里人发发，一直以来也都是相安无事，谁也没有怀疑账本会有什么问题。账房先生因为承担着替大家记账的任务，因此不用出去干活出工，额外会有些补贴，仅此一点，倒也是让一些人羡慕不已。下图便是当时账本的记账权图示：



终于有一天，有个人无意中发现了账房先生的那本账，看了下账面，发现数字不对，最关键的是支出、收入、余额居然不能平衡，对不上，这下可不干了，立即报告给其他人，结果大家都不干了，这还得了。经过一番讨论，大家决定，轮流来记账，这个月张三，下个月李四，大家轮着来，防止账本被一个人拿在手里。于是，账本的记账权发生了如下的图示变化：



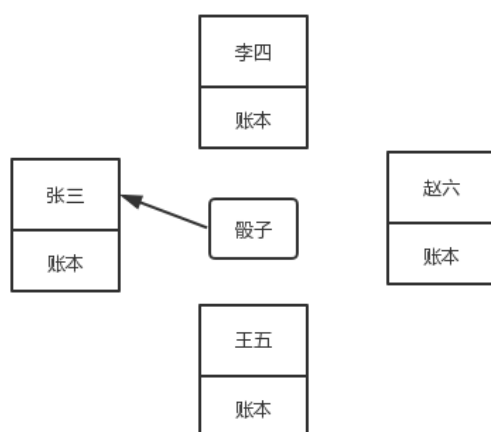
通过上图我们可以看到，村里的账本由大家轮流来保管记账了，一切又相安无事了，直到某一天，李四想要挪用下村里的公款，可是他又怕这个事情被后来记账的人发现，怎么办呢？俗话说恶向胆边生，李四决定烧掉账本的一部分内容，这样别人就查不出来了，回头只要告诉大家这是不小心碰到蜡烛了，别人也没什么办法。

果然，出了这个事情以后，大家也无可奈何，可是紧接着，下一个赵六也说不小心碰到了蜡烛；王五说不小心掉水里；张三说被狗啃了……终于大家决定坐下来重新讨论这个问题，经过一番争论，大家决定启用一种新的记账方法：每个人都拥有一本自己的账本，任何一个人改动了账本都必须告知其他所有人，其他人会在自己的账本上同样的记上一笔，如果有人发现新改动的账目不对，可以拒绝接受，到了最后，以大多数人都一致的账目表示为准。

果然，使用了这个办法后，很长一段时间内都没有发生过账本问题，即便是有人真的是不小心损坏了一部分账本的内容，只要找到其他的人去重新复制一份来就行了。

然而，这种做法还是有问题，时间长了，有人就偷懒了，不愿意这么麻烦的记账，就希望别人记好账后，自己拿过来核对一下，没问题就直接抄一遍，这下记账记的最勤的人就有意见了，最终大家开会决定，每天早上掷骰子，根据点数决定谁来记当天的账，其他人就只要核对一下，没问题就复制过来。

我们可以看到，在这个时候，账本的记账权变成了这样：



通过上图，我们可以看到，经历了几次风雨之后，大家终于还是决定共同来记账，这样是比较安全的做法，也怕账本损坏丢失了，后来大家还决定，每天掷骰子掷到要记账的人，能获得一些奖励，从当天的记账总额中划出一个奖励的比例。

实际上，最后大家决定的做法，就是区块链中记账方法的雏形了，接下来我们就来了解一下区块链的技术理念。

1.1.2 区块链技术理念

区块链在本质上就是一种记账方法，当然了，并不是通过人来记账的，而是通过一种软件，我们暂且简称为区块链客户端。以上面的例子来说，张三、李四、王五、赵六等人，就相当于是一个个的区块链客户端软件，它们运行在不同的设备上，彼此之间独立工作，通常我们把运行中的客户端软件称之为节点，这些节点运行后，彼此之间会认识一下，它们彼此之间是这样认识的：张三认识李四也认识王五，赵六联系到了张三，让张三把他认识的人的联系方式发给自己，这样赵六也认识了李四和王五，通过这样的方式，大家就形成了一张网，有什么事只要招呼一声，立马消息就会传遍整个网络节点，这个方式跟新闻转发差不多，不需要依靠某一个人，大家就能互通消息了，在区块链软件的结构中，这种互相通信的功能称之为是网络路由。

在这个网络中，每个节点都维护着自己的一个账本，账本中记录着网络中发生的一笔笔的账务，具体是什么样的账务呢？这得看具体是什么样的功能网络，区块链技术是属于一种技术方法，可以用来实现各种不同的业务功能，小到如上例中的日常记账，大到各种复杂的商业合约等等，记录的数据也就不同了。网络中的节点是独立记账的，可是记账的内容要保持彼此一致，所用的方法就是设定一个游戏规则，通过这个规则选出一个记账的节点，就如上例中的掷骰子，在区块链系统中，

这个所谓的“掷骰子”称之为是共识算法，就是一种大家都遵守的筛选方案，我们现在可以就这么先简单的理解下。选出一个节点后，则一段时间内的账务数据都以这个节点记录的为准，这个节点记录后会把数据广播出去，告诉其他的节点，其他节点只需要负责接收，通过网络来接收新的数据，接收后各自根据自己现有的账本验证一下能不能接的上，有没有不匹配不规范的，如果都符合要求，就存储到自己的账本中。

在有些系统中，会考虑到被骰子投中的节点的劳动付出，毕竟它要负责整理数据，验证数据，打包数据，还要再广而告之，这个活还是辛苦的，于是会设计一种激励机制，轮到打包的那个节点，可以获得系统的奖励，这个奖励类似于论坛积分一样，站在软件技术的角度，就是一个数据，这个数据可以被认为是奖金，有时候大家会很积极的去争取那个奖金，于是就希望骰子能投中自己，有些区块链系统在这个环节会设计出一种带有竞争的机制，让各个节点去抢，谁能抢到这个机会谁就能获得打包数据的权力并且同时获得这笔奖励，在这种情况下，我们会形象的将这个竞争的过程称之为是“挖矿”。

那么，话又说回来了，在上面我们将这一个个的运行客户端称之为是节点，那到底怎么标记不同的使用者呢？也是通过用户名注册么？实则不然，在区块链系统中，这个地方的设计是很有意思的，是通过一种密码算法来实现的，具体的说是通过一种叫公开密钥算法的机制来实现的，我们知道，对于一种密码算法来说，无论算法过程是什么样的，都会有一个密钥，而公开密钥算法拥有一对也就是两个密钥，跟虎符一样，是彼此配合使用的，可以互相用来加解密，其中一个叫私钥，另外一个叫公钥，公钥可以公开给别人，私钥要自己保管好，在区块链系统中，公钥就是用来当成用户的识别身份的，一般不会直接就使用公钥，因为不容易让人记住，往往都是比较长的，实际处理的时候都会进行一个转换，比如取得公钥的最后 20 个字节或者经过一系列更复杂的转换，最后得到一个称之为是“地址”的转换结果，这个“地址”就能代表一个用户。

为什么在区块链系统中要搞这么一个奇怪的用户身份表示方法呢？似乎看起来除了有些创意外，也没特别的用处啊。这里我们就得再介绍下这个公开密钥算法的特别能力，之前提到说这种算法有两个密钥，那么这两个密钥怎么配合工作的呢？我们来简单的说明一下，用公钥加密的数据必须用对应的私钥来解密，而用私钥加密（通常习惯称为“签名”）的数据必须用对应的公钥来解密。这个特点可是能发挥很大的用处的，就如上述的例子中，如果张三要发送给李四一张支票，那怎么传送呢？就这么发过去，会被那个记账的人拿到，风险可就大了。于是张三想了一个办法，他在支票上用李四的公钥加了个密，然后再签上自己的名字（使用自己的私钥签名），这个时候其他人就算拿到支票也没用，因为只有李四才有自己的私钥，也只有李四才能解开这张支票来使用。这种功

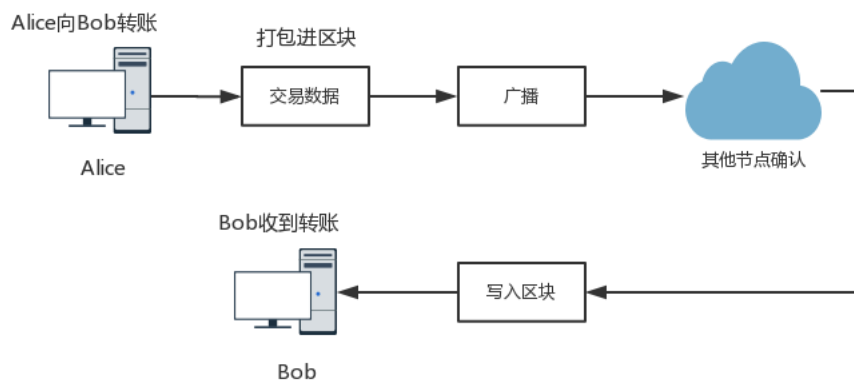
能设计在区块链系统中称之为脚本系统。

现在我们知道了，区块链的技术理念，其实就是大家共同来参与记账，通过一种规则不断地选出账务打包者，其他节点接收验证，并且每个用户都有一对密钥表示自己，通过脚本系统的功能可以实现现在这样的在一个公共的网络中定向发送有价值的信息。

1.1.3 一般工作流程

通过上面的例子，相信读者朋友对区块链已经有了一个基本的概念认识了，区块链系统有很多种，第一个应用区块链技术的软件就是比特币，事实上区块链的概念就是比特币带出来的，到现在为止，已经出现了相当多基于区块链技术的衍生系统了，比如闪电网络、公证通、以太坊、超级账本项目等，每一类系统都有自己的特点，就好比是汽车设计，有的是设计成跑车；有的是设计成运输车；有的是设计成商务车，但是有一点，无论是是什么类型的车，它的工作方式或者说工作流程都是类似的，在本质上它们都是同一类技术结构的产物。在这一小节，我们就来站在一个一般性的角度，阐述一下区块链系统的工作流程，为了便于说明，我们会选取一些场景例子。

我们先来看一个转账交易的流程，转账交易本质上就是发送一笔数据，这个数据可以表示为资产，也可以表示为订单或者其他各种形式的数据，我们看一下图示：

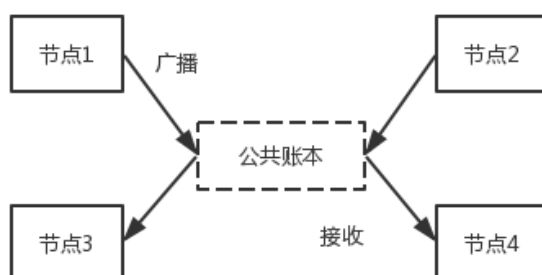


通过图中我们可以看到，整个数据的发送过程其实还是很简单的，数据发送出去后，会被打包进区块，然后广播出去给所有的节点确认，确认没有问题后就写入到各自的本地区块链账本中，当网络中的大多数节点都确认写入后，这个转账过程就算是定论了。有朋友可能会问，在这种分布式的网络中，怎么能知道是被大多数节点确认写入了呢？这里并没有什么服务器登记啊？这个问题我们先留着，在下面讲到区块链分类的时候会有详细的解释，大家可以先思考一下。

这个工作流程图是有代表性的，其他各种系统都是在这个基础上进行衍生和扩展，比如有些会

增加身份认证功能，以确保只有符合身份验证的用户才能发送数据；有些则扩展交易数据的表达能力，使得不但能用来表示一般的交易转账，还能表示更复杂的商业逻辑。各种应用很多，但是万变不离其中。

实际上，说一千道一万，整个一个区块链网络，就是大家共同来维护一份公共账本，注意了，这个公共账本是一个逻辑上的概念，每个节点各自都是独立的维护自己的账本数据的，而所谓的公共账本，就是说各自的账本要保持一致，保持一致的部分就是公共账本，我们看下图示：



如图所示，有些节点在广播新的数据，有些节点在接收数据，大家共同维持一个账本，确保达成一致，区块链技术其实就是围绕着如何来保持数据的一致，如何让这个公共账本的数据不被篡改来展开的，为了解决这些问题，区块链技术拥有一套技术栈，我们通过以下章节来阐述。

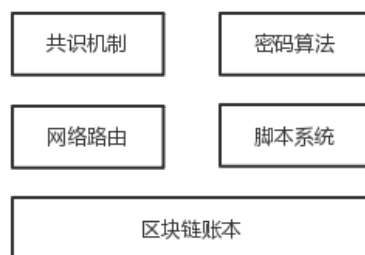
1.2 区块链技术栈

区块链本身只是一种数据的记录格式，就像我们平常使用的 Excel 表格、Word 文档一样，按照一定的格式将我们的数据存储在上，与传统的记录格式不同的是，区块链是将产生的数据按照一定的时间间隔，分成一个一个的数据块记录，然后再根据数据块的先后关系串联起来，也就是所谓的区块链了，按照这种规则，沿着时间线不断的增加新的区块，就好像是时光记录仪一样，记录下发生的每一笔操作。

这种数据记录的方式很是新颖，在这种记录方式下，数据很难被篡改或者删除，有朋友可能会有疑问，这有什么不好修改删除的，比如我在自己电脑上保存了一份 Excel 数据，再怎么复杂的格式，我也能随便改啊。如果区块链的数据格式只是应用在单机环境或者是一个中心的服务器上，那确实是的，毕竟自己对自己的数据拥有完全的支配权力。然而，一切才刚刚开始，我们接着看。

如果说区块链代表的仅仅只是一个记录格式的话，那么也实在算不上是伟大的发明，也看不出有什么特别的能力比如难以篡改之类。事实上，区块链是一整套技术组合的名词代表，在这一组技

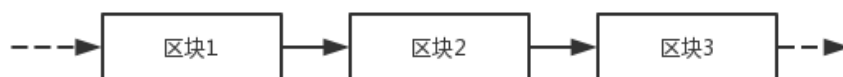
术的配合下，才能焕发出惊人的能力和耀眼的光芒。区块链系统有很多种，就像聊天软件有很多种，电子邮箱有很多种一样，而无论是什么样的区块链系统，其技术部件的组合都是类似的，就像汽车基本都是由发动机、底盘、车身、电器四大部件组成的；计算机都是由 CPU、存储器、输入输出设备组成的，不管是比特币、莱特币、以太坊还是其他，核心结构和工作原理都是共同的。我们就来看看最基本的技术组合都有哪些：



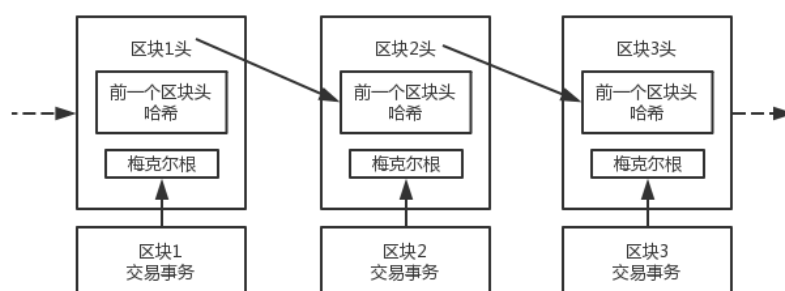
如图所示，这是区块链系统结构的基本组成，各种系统本质上都是在这个经典结构之上的直接实现或者是扩展实现，这些零部件装配在一起，组成了一个区块链软件，运行起来后就称之为一个节点，多个这样的节点在不同的计算机设备上运行起来，就组成了一个网络，在这个网络中每个节点都是平等的，大家互相为对方提供服务，这也是被称之为是点对点的对等网络。为了让大家对这些组成模块的名词有更具体的感受理解，我们一一来解释一下：

1. 区块链账本

如上所述，它表示一种特有的数据记录格式。区块链，就是“区块+链”，所谓的区块就是指数据块的意思，每一个数据块之间通过某个标志连接起来，从而形成一条链，我们看下示意图：



如图，一个区块一个区块的衔接，大家看到这样的格式，可以发现在生活中有很多相似的记录方式，比如企业的会计账簿，每个月会计将记账凭证汇总为账簿并且月结，每个月都汇总一次，这样一段时间下来，就形成了一个月一个月的连续的账簿，每个月的数据就相当于区块，区块与区块之间通过年月串联起来。以比特币来说，大约是为十分钟一个区块，区块中主要包含了交易事务数据以及区块的摘要信息，我们看下比特币中区块链数据的组成示意图：



通过上图我们可以看到比特币中区块链账本的数据组成以及关系，并且可以看到区块数据在逻辑上分成了区块头和区块体，每个区块头中通过梅克尔根关联了区块中众多的交易事务（梅克尔根也称之为是梅克尔根哈希值，具体的概念后续有详细的介绍，暂且可以认为就是一个区块中所有交易事务的集体身份证号），而每个区块之间通过区块头哈希值（区块头哈希值就是一个区块的身份证号）串联了起来。这是一个很有趣的数据格式，它将连续不断的發生的数据分成了一个一个的数据块。在下载同步这些数据的时候，可以并行的从各个节点来获得，无论数据先后，到达本地后再根据身份证号组装起来就行，另外，这种格式是一种链条的格式，链条最大的特点就是一环扣一环，很难从中间去破坏，比如有人篡改了中间的2号区块，那么就同时把2号区块后续的所有区块都要更改掉，这个难度就大了。在区块链系统中，一个节点产生的数据或者更改的数据要发送到网络中的其他节点接受验证，而其他节点是不会验证通过一个被篡改的数据的，因为跟自己的本地区块链账本数据匹配不起来，这也是区块链数据不可篡改的一个很重要的技术设计。

这种格式还有个巧妙的地方，如果这个数据总是由一个人来记录的，那自然也没什么，但是如果放到网络中，大家共同来记录这个数据，那就有点意思了，每个区块数据由谁来记录或者说打包，是可以有一个规则的，比如说掷骰子，大家约定谁能连续3次掷出6，那就让他来记录下一个区块的数据，为了补偿一下他的劳动投入，奖励给他一些收益，比特币正是使用了这样的原理来不断的发行新的比特币出来，奖励给打包记录区块数据的那个人的比特币就是新发行的比特币。

2. 共识机制

所谓共识，就是指大家都达成一致的意思，在生活中也有很多需要达成共识的场景，比如开会讨论，双方或多方签订一份合作协议等，在区块链系统中，每个节点必须要做的事情就是让自己的账本跟其他节点的账本保持一致，如果是在传统的软件结构中，这几乎就不是问题，因为有一个

中心服务器的存在，也就是所谓的主库，其他的从库向主库看齐就行了。在实际生活中，很多事情人们也都是按照这种思路来的，比如国家发布了一个政策，大家都照着办；企业老板发布了一个通知，员工照着做。但是在区块链的结构中，是一个分布式的对等网络结构，没有哪个节点是特别的“老大”，一切都要商量着来，因此在区块链系统中，如何让每个节点通过一个规则将各自的数据保持一致，是一个很核心的问题，这个问题的解决方案就是制定一套共识算法。

共识算法其实就是一个规则，每个节点都按照这个规则去确认各自的数据。我们暂且先抛开算法的原理，先来想一想在生活中我们会如何去解决这样一个问题，假设一群人开会，这群人中并没有一个领导或者说老大，大家各抒己见，那么最后如何统一出一个决定出来呢？实际处理的时候，我们一般会在某一个时间段中选出一个人来发表意见，那个人负责汇总大家的内容，然后发布出完整的意见，其他人投票表决，每个人都有机会来做汇总发表，最后谁的支持者多就以谁的最终意见为准。这种方法思路其实就算是一种共识算法了，然而在实际过程中，如果人数不多并且是数量确定的，那还好处理些，如果人数很多，而且数量也不固定，那我们就很难让每个人都去发表一遍意见然后再来投票决定了，这个时候大家轮一圈的做法，效率就太低了，我们需要通过一种机制筛选出最有代表性的人，在共识算法中就是筛选出具有代表性的节点。

如何筛选呢，其实就是设置一组条件，就像我们筛选运动员，帅选尖子生一样，给一组指标让大家来完成，谁能更好的完成指标，谁就能有机会被选上，在区块链系统中，存在着多种这样的筛选方案，比如 POW(Proof of Work 工作量证明)、POS (Proof of Stake 权益证明)、DPOS(Delegate Proof of Stake 委托权益证明)、PBFT(Practical Byzantine Fault Tolerance 实用拜占庭容错算法)以及等等，各种不同的算法，其实就是不同的游戏玩法，限于篇幅，这里暂不进行算法过程的详述，大家只要知道这些都是一套筛选算法就行了，区块链系统就是通过这种筛选算法或者说是共识算法来使得网络中各个节点的账本数据达成一致。

3. 密码算法

密码算法的应用在区块链系统中是很巧妙的，应用的点也很多，我们在这里不展开详细介绍密码算法的原理作用，就从几个很关键的应用来介绍下。

首先我们来回顾下区块链账本格式，通过上述的了解，我们已经知道了，区块链账本就是一个区块一个区块的连接，那么到底是通过什么来连接的呢？学过软件数据结构的朋友知道，在数据结构中，有一种变量叫指针，是可以用来指向某个数据的地址的，那么区块的连接是不是通过这样的数据地址呢？生活中，通过地址来连接指向的例子很多，比如路牌、门牌等。然而，区块之间的

连接，往往都不是靠数据地址来关联的，而是靠一种叫做哈希值的数据来关联的，什么叫哈希值？这是通过密码算法中的哈希算法来计算得出的，哈希算法可以通过对一段数据计算后得出一段摘要字符串，这种摘要字符串与原始数据是唯一对应的，什么意思呢？如果对原始数据进行修改，哪怕只是一点点的修改，那么计算出来的哈希值就会发生完全的变化。区块链账本中对每个区块都会计算出一个哈希值，称之为区块哈希，通过区块哈希来串联区块，有一个很好的作用就是，如果有人篡改了中间的某一个区块数据，那么后面的区块就都要进行修改，这个时候并不是简单的靠修改一下后面区块的地址指向就能结束的，由于后面的区块是通过区块哈希来指向的，只要前面的区块发生变动，这个区块哈希就无效了，就指不到正确的区块了。

另外一个对密码算法的应用就是称之为梅克尔树的结构，梅克尔树的结构在后续章节中有详细的介绍，我们这里先初步认识下，通过上述的解释，我们知道，每个区块会被计算出一个哈希值，实际上，除了整个区块会被计算哈希值外，区块中包含的每一笔事务数据也会被计算出一个哈希值，称之为是事务哈希，每一个事务哈希都可以唯一的表示一条事务，通过对一个区块中所有的事务进行哈希计算后，可以得出一组事务哈希，再通过对这些事务哈希进行一个加工处理，最终会得出一棵哈希树的数据结构，哈希树的顶部就是树根，被称之为是梅克尔根，通过这个梅克尔根就可以将整个区块中的事务约束起来，只要区块中的事务有任何改变，梅克尔根就会发生变化，利用这一点，可以确保区块数据的完整性。

当然，密码算法在区块链系统中的应用还远不止这些，比如通过密码算法来创建账户地址、签名交易事务等，这些应用点我们在后续章节中会逐步的介绍。

4. 脚本系统

脚本系统在区块链中是一个相对抽象的概念，也是极其重要的一个功能，可以说区块链系统之所以能形成一个有价值的网络，依靠的就是脚本系统，它就像是发动机一样，驱动着区块链系统不断的进行着各种数据的收发。所谓的脚本，就是指一组程序规则，在区块链系统中，有些系统中的程序规则是固定的，比如比特币，在比特币系统中，只能进行比特币的发送与接收，这个发送与接收的过程就是通过实现在比特币中的一组脚本程序来完成的；而有些系统是允许用户自行编写一组程序规则的，编写好后可以部署到区块链账本中，这样就可以扩展区块链系统的功能，比如以太坊就是通过实现一套可以自定义功能的脚本系统，进而实现了智能合约的功能。

脚本系统使得在区块链中可以实现各种各样的业务功能，本来大家只是通过区块链来记个财务账的，通过脚本系统，大家可以使用区块链来记录各种各样的数据，比如订单、众筹账户、物流信

息、供应链信息等等，这些数据一旦可以记录到区块链上，那么区块链的优点就能够被充分的发挥出来。有关脚本系统的具体使用和开发，大家可以通过后续的第六章、第七章以及第八章来理解。

5. 网络路由

这个功能模块比较简单，区块链系统是一个分布式的网络，这些网络中的节点如何来彼此进行连接通信呢？依靠的就是网络路由功能。在上述的章节中，我们有看到，张三李四以及王五赵六是通过彼此介绍来认识的，这个其实就是网络路由的雏形了，在分布式的网络结构中，不存在一个指定的服务器，大家没法通过一个服务器来直接交换彼此的身份信息，就只能依靠彼此联系，像传销一样将信息传播出去，在区块链系统中，这个一般会定义成一种协议，称之为节点发现协议。

除了要发现节点外，更重要的一个功能需求就是同步数据，节点要保持自己的账本数据是最新的，就必须时时更新自己的数据，从哪更新呢？既然没有服务器来下载，那就是通过邻近的节点了，通过向邻近节点发送数据请求来获得最新的数据，节点彼此都充当服务者和被服务者，通过这种方式，网络中的每一个节点都会在某一个时刻达成数据上的一致。

网络路由可以说就是区块链系统中的触角，通过大量的触角将每个节点连入了网络，从而形成一个功能强大的区块链共识网络。

1.3 区块链分类与架构

通过上述的了解，我们知道了区块链系统实际上就是一个维护公共数据账本的系统，一切技术单元的设计都是为了更好的维护好这个公共数据账本。通过共识算法达成节点的账本数据一致；通过密码算法确保账本数据的不可篡改性以及数据发送的安全性；通过脚本系统扩展账本数据的表达范畴。站在本质的角度，我们甚至可以认为，区块链系统实际上就是一种特别设计的数据库系统或者说是分布式数据库系统，在这个数据库中可以去存储数字货币，也可以去存储逻辑更复杂的智能合约以及范围更加广阔的各种业务数据。在区块链系统的发展过程中，也经历了这样一个阶段，从比特币开始，早期的区块链系统都是面向数字货币的，如比特币、莱特币等，这个阶段我们可以认为区块链系统是一个支持数字货币合约的系统；之后便出现了更加灵活的，能够支持自定义智能合约的系统，其代表作是以太坊，可以认为以太坊就是对比特币这样的数字货币系统的扩展，不过以太坊仍然是内置一个数字货币的支持的，延续了比特币系统的金融特征，也使得以太坊的应用更多的还是面向金融范畴；再之后的代表就是超级账本项目，尤其是其中的 fabric 子项目，在这个系统中，超越了对金融范畴的应用，支持各个领域的数据定义，我们分别将这三个阶段称为区块链系统

的 1.0\2.0\3.0 架构时期。为了让大家对发展过程中的区块链系统有一个整体的概念，在本节中，我们来描述一下通常的区块链系统的架构，并站在不同的角度对区块链系统进行一个分类。

1.3.1 区块链架构

1. 区块链 1.0 架构

如上所述，这个阶段区块链系统主要是用来实现数字货币的，我们看一下示意图：



如图所示，在整个架构中，分为核心节点和前端工具，这里提一下核心节点中“矿工”的功能，矿工在 1.0 架构的系统中，主要是承担两个任务，第一个是通过竞争获得区块数据的打包权后将内存池（发送在网络中但是还没有确认进区块的交易数据，属于待确认交易数据）中的交易数据打包进区块，并且广播给其他节点；第二个是接受系统对打包行为的数字货币奖励，从而系统通过这种奖励方式完成新增货币的发行。

在前端工具中，最明显的就是一个钱包工具，钱包工具是提供给用户管理自己的账户地址以及余额的；浏览器则用来查看当前区块链网络中发生的数据情况，比如最新的区块高度、内存池的交易数、单位时间的网络处理能力等；RPC 客户端和命令行接口都是用来访问核心节点的功能的，在这个时候，核心节点就相当于是一个服务器，通过 RPC 服务提供功能调用接口。

2. 区块链 2.0 架构

区块链 2.0 架构的代表产品就是以太坊，因此我们可以套用以太坊的架构来说明，先看下示意图吧：



如图所示，与 1.0 架构相比，最大的特点就是支持智能合约，在以太坊中，我们使用智能合约开发工具开发合约程序，并且编译为字节码，最终部署到以太坊的区块链账本中，部署后的智能合约是运行在一个虚拟机上的，称之为以太坊虚拟机。正是通过这样的一个智能合约的实现，扩展了区块链系统的功能，同时我们也能看到，在以太坊中还是支持数字货币的，因此在应用工具中也是有一个钱包工具的。

3. 区块链 3.0 架构

在 3.0 的架构中，超越了对数字货币或者金融的应用范畴，而将区块链技术作为一种泛解决方案，可以在其他领域比如行政管理、文化艺术、企业供应链、医疗健康、物联网、产权登记等方面的应用，可以认为是面向行业应用。

行业应用一般是需要具备企业级的属性的，比如身份认证、许可授权、加密传输等，并且对数据的处理性能也会有要求，因此企业级场景下的应用，往往都是联盟链或者私有链。我们来一下示意图：



如图所示，首先在 3.0 架构中，数字货币不再是一个必选组件了，当然如果需要，我们也是可以通过智能合约的方式来实现数字货币的。与之前的架构相比，最大的特点就是增加了一个网关控制，实际上就是增加了对安全保密的需求支持，并且通过数据审计加强对数据的可靠性管理。

在 3.0 架构中，实际上可以看成是一套框架，通过对框架的配置和二次开发可以适应各行各业的需求，比如图中的“可插拔共识”，意思就是共识机制不是固定的，而是可以通过用户自己去选用配置。

1.3.2 区块链分类

1. 根据网络范围

可以划分为公有链、私有链、联盟链。

(1) 公有链

所谓公有就是完全对外开放，任何人都可以任意使用，没有权限的设定，也没有身份认证之类，不但可以任意的参与使用，而且发生的所有数据都可以被任意查看，完全的公开透明，比特币就是一个公有链网络系统，大家在使用比特币系统的时候，只需要下载相应的软件客户端，创建钱包地址、转账交易、挖矿等操作，都可以自由的使用。公有链系统由于完全没有一个第三方管理，因此

依靠的就是一组事先约定的规则在进行,这个规则要确保每个参与者在不信任的网络环境中能够发起可靠的交易事务。通常来说,凡是需要公众参与,需要最大限度保证数据公开透明的系统,都适用于公有链,比如数字货币系统、众筹系统、金融交易系统等。

这里说一个注意点,在公有链的环境中,节点数量是不固定的,节点的在线与否也是无法去控制的,甚至节点是不是一个恶意节点也不能保证。我们在上述章节中讲解区块链的一般工作流程的时候,提到过一个问题,在这种情况下,如何知道数据是被大多数的节点写入确认的?实际上在公链环境下,这个问题没有很好的解决方案,目前最合适的做法就是通过不断的去互相同步,最终网络中大多数节点都同步一致的区块数据所形成的链就是被承认的主链,这也被称之为是最终一致性。

(2) 私有链

这是与公有链相对的一个概念,所谓私有就是指不对外开放,仅仅在组织内部使用的系统,比如企业的票据管理、账务审计、供应链管理等,或者是一些政务管理系统。私有链在使用过程中,通常是有注册要求的,也就是需要提交身份认证,而且具备一套权限管理体系。有朋友可能会有疑问,比特币、以太坊等系统虽然都是公链系统,但如果将这些系统搭建在一个不与外网连接的局域网中,这个不就成了私有链了吗?从网络传播范围来看,可以算,因为只要这个网络一直与外网隔离着,就只能是一直自己在使用,只不过由于使用的系统本身并没有任何的身份认证以及权限设置,因此从技术角度来说,这种情况只能算是使用公链系统的客户端搭建的私有测试网络,比如以太坊就可以用来搭建私有链环境,通常这种情况可以用来测试公有链系统,当然也可以适用于企业应用。

在私有链环境中,节点数量和节点的状态通常是可控的,因此在私有链环境中一般不需要去通过竞争的方式来筛选区块数据的打包者,可以采用更加节能环保的方式,比如在上述共识机制的介绍中提到的 POS (Proof of Stake 权益证明)、DPOS(Delegate Proof of Stake 委托权益证明)、PBFT(Practical Byzantine Fault Tolerance 实用拜占庭容错算法)以及等等。

(3) 联盟链

联盟链的网络范围介于公有链和私有链之间,通常是使用在多个成员角色的环境中,比如银行之间的支付结算、企业之间的物流等等,这些场景下往往都是由不同权限的成员参与的,与私有链一样,联盟链系统一般也是具有身份认证和权限设置的,而且节点的数量往往也是确定的,对于企业或者说机构之间的事务处理很合适。联盟链并不一定要完全的管控,比如政务系统,有些数据可以对外公开的,则可以部分开放出来。

由于联盟链一般是使用在明确的机构之间,因此与私有链一样,节点的数量和状态也是可控的,

并且通常也是采用更加节能环保的共识机制。

2. 根据部署环境

（1）主链

所谓主链，也就是部署在生产环境的真正的区块链系统，软件在正式发布前会经过很多内部的测试版本，用于发现一些可能存在的 BUG，并且用来内部演示以便于查看效果，直到最后才会发布正式版。主链，也可以说是由正式版客户端组成的区块链网络，只有主链才是会被真正推广使用的，各项功能的设计也都是相对最完善的。另外，有些时候，区块链系统会由于种种原因导致分叉，比如挖矿的时候临时产生的小分叉等，此时将最长的那条原始的链条称之为是主链。

（2）测试链

这个很好理解，就是开发者为了方便大家学习使用而提供的测试用途的区块链网络，比如比特币测试链，以太坊测试链等，当然了，倒也不是说非得是区块链开发者才能提供测试链，用户也可以自行搭建测试网络。测试链中的功能设计与生产环境中的主链是可以有一些差别的，比如主链中使用工作量证明算法进行挖矿，在测试链中可以更换算法以更方便进行测试使用。

3. 根据对接类型

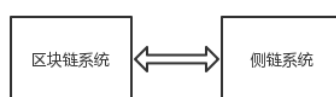
（1）单链

能够单独运行的区块链系统都可以称之为是单链，例如比特币主链、测试链；以太坊主链、测试链；莱特币的主链、测试链；超级账本项目中的 Fabric 搭建的联盟链等，这些区块链系统拥有完备的组件模块，自成一个体系。大家要注意了，对于有些软件系统，比如基于以太坊的众筹系统或者金融担保系统之类，这些只能算是智能合约应用，不能算是一个独立的区块链系统，应用程序的运行需要独立的区块链系统的支撑。

（2）侧链

侧链是属于一种区块链系统的跨链技术，这个概念主要是由比特币侧链发起的，随着技术发展，除了比特币，出现了越来越多的区块链系统，每一种系统都有自己的优势特点，如何将不同的链结合起来，打通信息孤岛，彼此互补呢？侧链就是其中之一技术。以比特币来说，比特币系统主要是设计用来实现数字加密货币的，且业务逻辑也都固化了，因此并不适用于实现其他的功能例如金融智能合约、小额快速支付等，然而比特币是目前使用规模最大的一个公有区块链系统，在可靠性、去中心化保证等方面具有相当的优势，那么如何利用比特币网络的优势来运行其他的区块链系统

呢？可以考虑在现有的比特币区块链之上，建立一个新的区块链系统，新的系统可以具备很多比特币没有的功能比如私密交易、快速支付、智能合约、签名覆盖金额等，这些新的功能的使用又通过比特币网络创造更多的其他应用，并且能够与比特币的主区块链进行互通，简单的说，侧链是以锚定比特币为基础的新型区块链，锚定比特币的侧链，目前有 ConsenSys 的 BTCRelay、Rootstock 和 BlockStream 的元素链等。大家要注意，侧链本身就是一个区块链系统，并且侧链并不是一定要以比特币为参照链，这是一个通用的技术概念，比如以太坊可以作为其他链的参照链，也可以本身作为侧链与其他的链去锚定。实际上，抛开链啊网络啊这些概念，就是不同的软件之间互相提供接口，增强软件之间的功能互补，我们看下侧链的示意图：



通过这个简单的示意图，我们可以看到，区块链系统与侧链系统本身都是一个独立的链系统，两者之间可以按照一定的协议进行数据互动，通过这种方式，侧链能起到一个对主链功能扩展的作用，很多在主链中不方便实现的功能可以实现在侧链中，而侧链再通过与主链的数据交互增强自己的可靠性。

（3）互联链

曾经我们的计算机是不联网的，所有的软件都是单机运行的，大家以为计算机也不过如此，就是能够做点办公，玩玩游戏之类而已，后来有了互联网，各种好玩的、强大的应用雨后春笋般冒出来，如今我们的生活可以说几乎已经离不开互联网了，仅仅一个互通互联，带来的能量是如此的巨大。

区块链也是这样，目前各种区块链系统不断涌现，有的只是实现了数字货币，有的实现了智能合约，有的实现了金融交易平台，有些是公有链，有些是联盟链，等等等等，这么多的链，五彩缤纷，功能各异，各种新奇的应用，脑洞大开的设想，不断的在刷新着更新颖的应用玩法。那么，这些链系统如果能够彼此之间互联会发生些什么样的化学反应呢？与传统软件不同的是，区块链应用拥有独特的性质，比如数据不可篡改性、完整性证明、自动网络共识、智能合约等，从最初的数字货币到未来可能的区块链可编程社会，这些不单单会改变生活服务方式，还会促进社会治理结构的变革，如果说每一条链都是一条神经的话，一旦互联起来，就像是神经系统一般，将会给我们的社会发展带来更新层次的智能化。

另外一个，从技术角度来讲，区块链系统之间的互联，可以彼此互补，每一类系统都会有长处

和不足之处，彼此进行功能上的互补，甚至可以彼此进行互相的验证，可以大大的加强系统的可靠性以及性能。

1.4 创世元灵：一切源自比特币

当我们坐在飞机上，开启一段美妙的旅程时，可否会想起当初的莱特兄弟；当我们坐在高铁里，享受着高效的都市穿梭时，可否会想起当初的蒸汽机；当我们住在舒适的房屋里，享受着安心的睡眠时，可否会想起当初的茅草房。是的，这个世界给了我们很多原材料，我们使用原材料，制造出了一个又一个工具，通过这种方式，改造这个世界，改善我们的生活。区块链，便是这样的一个改造世界的原材料，而有人，用它制造出了第一个工具，它的名字叫比特币。

1.4.1 比特币白皮书

通常，在介绍一个比较重量级的人物时，我们常常会在他的名字前面加上很多定语，比如某著名歌唱家、慈善大使、两届 XX 奖获得者等等，然后最后才报出名字，为的就是让大家竖起耳朵听明白，这个牛逼的人物都能干些啥。而在介绍一个物件时，比如一辆汽车，我们就不会这么说了，那要说起来，能把人说睡着喽，一个东西嘛，写个说明不就完了，一目了然。

那么，比特币白皮书就是这么一个说明书，当然了，人家这份说明书可是有正式名字的，可不叫什么比特币白皮书，人家的大名叫《Bitcoin: A Peer-to-Peer Electronic Cash System》，翻译过来叫《比特币：一种点对点的电子现金系统》。之所以称它是白皮书，是因为这份文件基本就是宣告了比特币的诞生，严格的说是理论上宣告了比特币的诞生，这份文件是在 2008 年 11 月，由一个叫 Satoshi Nakamoto（中本聪）的人发布的，当然了，并不是发布在什么知名论坛或者学术期刊上的，而是发布在一个小众的密码学讨论小组。在这份白皮书发布后的第二年，也就是 2009 年 1 月 3 日，比特币软件就正式启动运行了，也就是在这个时候，世界上第一个区块链数据诞生了，而这个由中本聪构造出来的第一个区块，也称之为创世区块或者上帝区块，反正就是神话中创世元灵那么个意思，从此以后，比特币以及由比特币技术衍生出来的其他各种应用就一发不可收拾，开启了互联网应用的一个新纪元。

咱们还是回到这个白皮书上来，注意看它的标题，有两个关键字：“点对点”和“电子现金”，有朋友说了，这俩词语压根就没提什么区块链嘛，别着急，咱们先来解释一下，看人看眼睛，读文读标题。“点对点”，就是指这个软件不需要一个特定的服务器，比如我们登录 QQ 就需要连接腾讯

的 QQ 服务器，登录支付宝就需要连接阿里巴巴的支付宝服务器，倘若这些服务器关闭或者出个问题什么的，那就没法正常使用这些软件了，2015 年 5 月份，杭州电信光缆被施工队不甚挖断，直接导致通过这些光缆联网的支付宝服务器断网，影响了正常的运行，而点对点的网络结构，并不依赖于某一个或者某一群特定的服务器，相当于人人都是服务器，人人也都是使用者；再来看“电子现金”，顾名思义，现金嘛，就是钱或者说货币的意思，也就是说这份白皮书，介绍的是一种数字货币系统，这个系统的运行不依赖于某些特定服务器，而是通过点对点的这么一种网络结构来运行的。相信有些读者朋友看到这里，还是会有些懵，不要紧，毕竟咱们到这会儿才看标题嘛，有个概念就行了。

翻开白皮书正文，可以发现，整个篇幅主要介绍了几个关键点。

（1）简介

提出了一个场景设想，如何不通过一个所谓的权威第三方结构比如银行，来构建一个可信的交易网络，中本聪的语文还是不错的，先抛出个问题给你玩玩，然后吸引你继续看下去-

（2）交易

描述了一种通过密钥签名进行交易验证的方式，实际上就是计算机密码学在比特币中的应用，我们在银行转账交易用什么来证明自己呢，是通过账户和密码，必要的时候还可以通过身份证确认，而在比特币系统中没有银行这样的一个角色，那靠什么来确定身份呢，只有靠现代计算机密码学技术了。当然，密码学技术在比特币中的应用并不只是用来证明身份，是贯穿在各个环节的，可以说，密码学技术就是比特币系统的骨骼。

（3）时间戳服务器

这一节中，提到了区块以及通过时间戳运算连接成一条链的概念，这也是区块链概念的来源，同时在这里也说明了比特币数据的存储方式。

（4）工作量证明

介绍了一种点对点网络中，如何对各自的数据进行一致性确认的算法，为什么叫工作量证明呢？因为这种算法很消耗 CPU 的算力，等于人们干活一样，要付出工作劳动的。

（5）网络

比特币软件是一种网络软件，而且是一个不依靠某个服务器来交换数据的网络软件，那么一个个的节点之间，如何来确认一笔笔的交易数据呢？这一章就是介绍了交易确认的过程，这个实际上就是比特币网络的应用协议，跟日常使用的邮件收发协议、文件传输协议、超文本传输协议等，是一个层面上的意思。

（6）激励

激励就是奖励的意思，你干了活，得到一笔奖金，哇好开心，就会继续努力干活，这就是激励，第 4 点说了，比特币软件的数据一致性确认是需要耗费 CPU 的算力的，那凭什么愿意来耗费这些个算力，白干活么？当然不是，系统会奖励给你比特币还有别人交易的手续费（哦，好开心，那我为什么没被奖励到过啊，别急，在 1.1.3 节中会有详述）

（7）回收硬盘空间

比特币系统从创世区块开始，每 10 分钟一个区块，也意味着区块链账本的体积一直在增长，事实上就写作本书的时候，已经超过了 120G，只要比特币网络一直存在，数据就会一直增长（实际上，只有运行全功能节点的客户端才会一直保持完整的区块链数据，这些在 1.1.2 中会有详述），这里提出了一个思路，删除掉过老的一些交易数据，同时不破坏区块的随机散列值，通过这种方法压缩区块数据。

（8）简化的支付确认

上述提到了，比特币客户端的数据量很大，这么一来，等于不管是用比特币系统干什么都要带上这么大一坨，这岂不是很不方便，而且也限制在其他一些终端比如手机上的使用。这个章节就提出了一个模型，这个模型主要是为比特币的支付服务的，在这个模型下实现的比特币支付功能并不需要携带那么一大坨的数据，而只需要保留体积相对很小的区块头，具体细节可以查看 1.1.4 节。

（9）价值的组合与分割

这一节介绍的是比特币中的交易事务组成方式。什么叫价值？在比特币系统中，价值就是比特币。什么叫组合？就是我口袋里有 5 枚 1 元硬币，1 张 2 元纸币，1 张 10 元纸币，这个时候我要给你 5 块钱，怎么给呢，我可以给你 5 枚 1 元硬币，也可以给你 3 枚 1 元硬币加上 1 张 2 元纸币，这就是不同的组合。什么叫分割？分割其实就是转出的意思，我通过不同的组合，构成了总计 5 元的金额，然后转出给你，这个过程就是价值的组合和重新分割。在这个例子中，还有一张 10 元的，假如我就直接转了你 10 元，那会怎样？就需要找零 5 元了，这个找零其实也是一种重新价值分割。

（10）隐私

作为一个货币系统，保密性也就是隐私毫无疑问是人人都会关心的，传统的体系，完全是依赖比如银行这个第三方的保护，大家相信银行，银行也设立了各种管理制度和方法来防止账户和交易信息的泄密。比特币系统则不同，它不依赖谁，每个人在比特币系统中也不用登记什么身份证、名

称、性别什么的，就是一个地址，谁也不知道地址后面代表的是谁，而且，只要你需要，可以自己创建任意多个地址（你到银行去开任意多个户试试！），这使得比特币系统中的交易带有很大的匿名性和隐秘性。

（11）计算

这一节主要是站在概率统计的角度，计算了一下攻击者成功的概率，以及经过多少个区块后还能攻击成功的概率，计算过程这里就不再赘述了。

白皮书的内容就介绍到这里了，若有刚刚接触比特币、区块链这些概念的朋友，或许还会是一头雾水吧，没关系，我们在下面的章节会有详细的解释，毕竟，能够只通过一份白皮书就能完全明白比特币设计的人，或许只有中本聪这个大神了。

小提示：白皮书的原文可以在 <https://bitcoin.org/bitcoin.pdf> 进行查看，感兴趣的朋友可以阅读一下，英文不那么擅长的朋友，可以到巴比特网站（著名的区块链资讯与技术服务网站）上查看中文版，地址是 <http://www.8btc.com/wiki/bitcoin-a-peer-to-peer-electronic-cash-system>。读明白了这份说明书，基本也就理解了比特币的原理，也就入了区块链这个坑，哦不，这个门了。

1.4.2 比特币核心程序：中本聪客户端

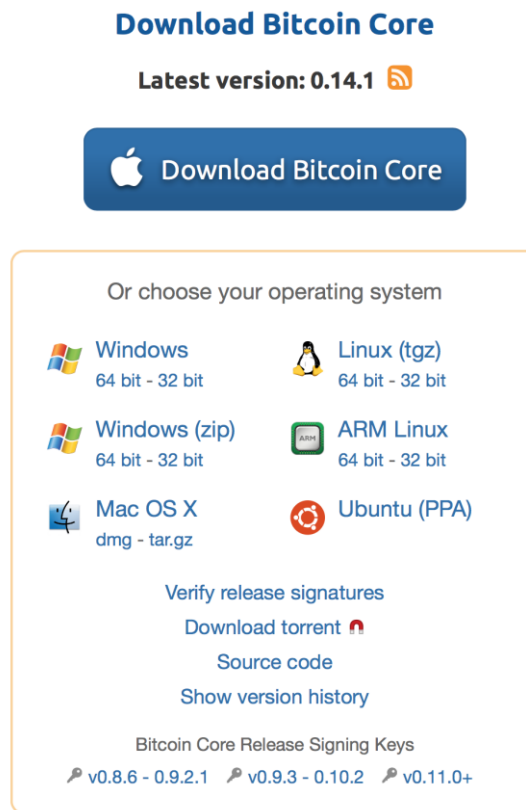
1. 客户端程序介绍

我们知道，比特币其实就是一个软件，既然是软件，那还是百闻不如一见，看看到底长什么样。大家可以到这个地址去下载客户端程序 <https://bitcoin.org/en/download>，可以看到，网站提供了多种操作系统的运行版本，选择自己需要的版本下载安装即可运行了，就能看到庐山真面目啦。






在具体介绍之前，咱们先说明一下，为什么这个程序叫比特币核心程序，难道还有非核心程序？我们在上述提供的下载页面上，可以看到比特币程序的名字叫 **Bitcoin core**，这个翻译过来就是比特币核心的意思，这是最经典，也是中本聪一开始发布的那一支程序版本，这个版本也是使用人数最多的。可问题是，比特币程序是开源的，任何一个人或者一个组织都可以根据需要去修改源码发布出一个新的版本，事实上经过多年的发展，比特币程序已经出现了多个版本，比如 **Bitcoin Classic**、**Bitcoin XT** 以及 **Bitcoin Unlimited**，这些不同的版本实际上都是比特币核心程序的分叉版本，本节我们使用的是比特币核心程序的客户端。

现在我们先给自己的电脑安装一个比特币核心客户端吧，按照下载地址进入页面后，可以在这个页面看到有列出不同操作系统之上的下载版本，读者朋友可以自行选择，无论哪个系统环境下，

其功能都是一样的，见下图：



我们以 windows 版本为例来说明，我们下载图中所示的 0.14.1 版，大家注意到没有，比特币发展了这么多年，到现在程序都还没进化到 1.0 版（通常一个软件的 1.0 版是首个正式版本），某种程度上，也是因为比特币是一种实验性的软件吧，因此大家研究学习比特币可以带着一种玩的姿态，不要搞那么严肃，任何的思路想法，任何的可能性都是有的，我们学习了解比特币是为了更好的应用它的设计思想，而不是去迷信它的神秘和权威，咱们继续。下载完成后，打开软件目录，可以看到有一个 bin 文件夹，其中有 5 个文件，如图所示：

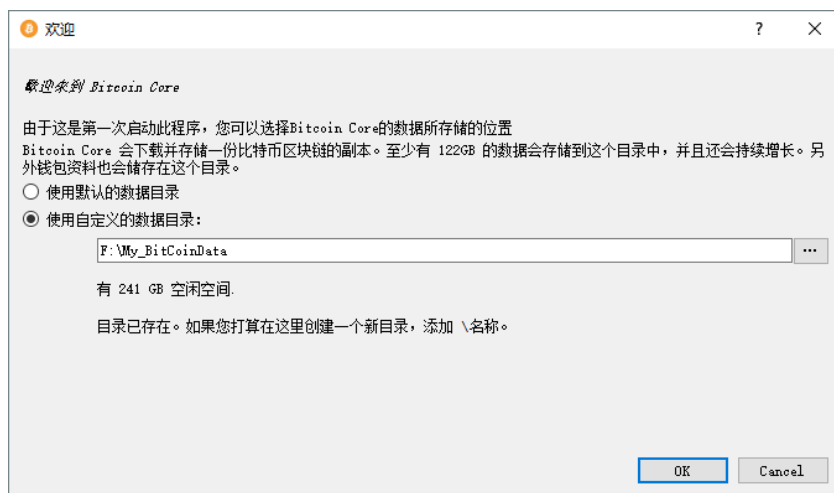
 test_bitcoin.exe
 bitcoin-tx.exe
 bitcoin-qt.exe
 bitcoind.exe
 bitcoin-cli.exe

我们一一来说明一下：

（1）bitcoin-qt.exe

包含了比特币的核心节点以及一个钱包的前端功能，这是一个带有图形界面的客户端程序，运

行后可以看到有如下提示：



图中所示，需要选择一个比特币的区块链副本数据存储目录，目前整个区块链账本数据已经很大了，选择一个空间足够大的目录然后点击“OK”即可进入到主界面了，我们看下主界面的样子：



如图所示，这便是比特币核心客户端了（等等，标题上不是说是钱包么，怎么又是核心客户端，到底运行的这个程序是什么？），是的，这个客户端也叫“中本聪客户端”（satoshi client），它实现了比特币系统的所有方面，包括钱包功能，对整个交易数据也就是区块链账本完整副本的交易确认功能，以及点对点比特币对等网络中的一个完整网络节点。换句话说，这个客户端软件包含除了挖矿以外的其他所有比特币的功能模块，我们当然也可以分别去自己实现一个个的独立功能客户端，比如仅仅实现一个钱包功能，仅仅实现一个核心节点功能，只不过这个官方的客户端都集成在一起了。

通过这个界面，我们也能看到在底部有显示“正在连接到节点”以及“落后 8 年和 16 周”的字样，

这是指运行中的核心客户端通过发现与连接网络中其他的节点，进行区块链账本数据的一致同步，如果是首次开始同步，需要花费不少时间，一百几十 G 的数据下载真够喝一壶的，需要注意的是，所有的操作都要等到同步完成后才能进行。点击那个“落后 8 年和 16 周”的区域可以看到具体的同步进度信息：

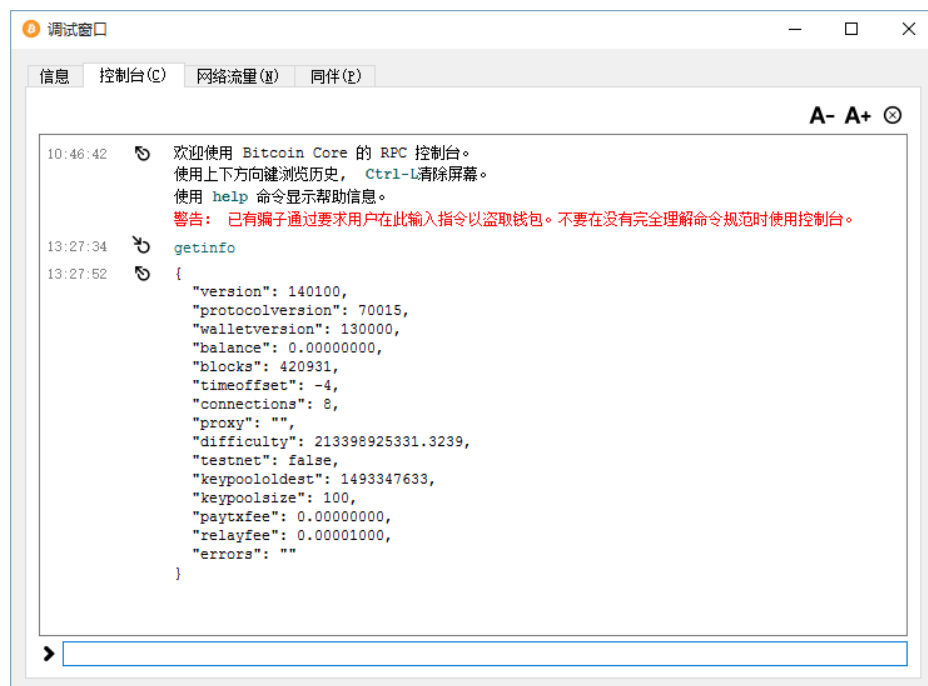


图中可以看到有剩余的区块数、进度以及剩余时间等信息，耐心等待就是了。如果想查看一下当前客户端的版本以及网络连接等信息，可以点击”帮助”→”调试窗口”调出如下界面：



在”信息”标签页下可以看到软件版本、当前的网络连接数、数据目录等摘要信息，注意这里的“客户端版本“，比特币是一个分布式的点对点系统，不存在一个中心服务器来统一管理软件的版本升级，因此不同的节点是有可能运行着不同版本的客户端的，不同版本的客户端在一些功能支持上可能会有些差异，大家在操作时，一定要注意自己的版本。在”信息”标签页旁边有个“控制台”，

这可是个很有用的功能，在控制台可以通过命令来访问核心客户端，调取一些信息，进行一些操作，我们来看下控制台：



我们在控制台底部的输入框中输入了一个 `getinfo` 命令，回车确认后可以发现返回了一段信息，是关于当前运行的核心客户端节点的一些摘要信息，比如 `version` 表示核心客户端版本，`protocolversion` 表示协议版本，`walletversion` 表示钱包版本，`balance` 表示当前钱包中的比特币余额等。通过这个我们发现，比特币的核心客户端其实是充当了一个服务器的角色，通过控制台可以连接访问，通过界面也能看到有提示：“欢迎使用 Bitcoin Core 的 RPC 控制台”。实际上比特币核心客户端就是在启动时同时启动了一个本地的 `RPC` 服务以方便外部程序进行相应的数据操作和访问。

有朋友问，比特币一下子要同步这么多的数据，我只是想看一看，了解了解的，有没有试用的版本呢？还真有，不过不叫试用版，而是测试网络，为了方便大家测试使用，比特币有提供测试网络，那么如何连接到测试网络呢？可以通过配置文件来进行配置，比特币的配置文件名为 `bitcoin.conf`，可以在数据目录也就是钱包数据文件 `wallet.dat` 所在目录下，创建一个文本文件，命名为 `bitcoin.conf` 即可，这就是 `bitcoin-qt` 默认读取的配置文件了，接下来我们就来配置一下如何进入测试网络，只需在 `bitcoin.conf` 中写入如下配置项：

```
testnet=1
```

保存即可，然后重新启动 `bitcoin-qt.exe`，我们可见如下画面：



我们发现颜色都变了，变成了淡绿色，标题上也有“测试网络”的字样，进入到主界面后，界面样式基本还是那样：



进入到测试网络后的比特币客户端，其区块链数据会小一些，在功能操作上基本还是一样的。需要注意的是，在配置文件中的配置项，也是可以直接通过参数来传递的，假设只是想临时进入测试网络看看，那么就不需要去设置配置文件了，通过如下指令来运行即可：

```
bitcoin-qt -testnet
```

在控制台中执行后，同样会进入测试网络中，有朋友会问，我一开始在运行 bitcoin-qt 时指定了一个数据目录，现在我想更换可以吗？当然是可以的，如下：

```
bitcoin-qt -datadir="D:\mybitcoin_data"
```

这样在启动 bitcoin-qt 的时候重新指定了一个自己创建的数据目录，当然了，不但可以重新指

定数据目录，也可以重新指定配置文件，如下：

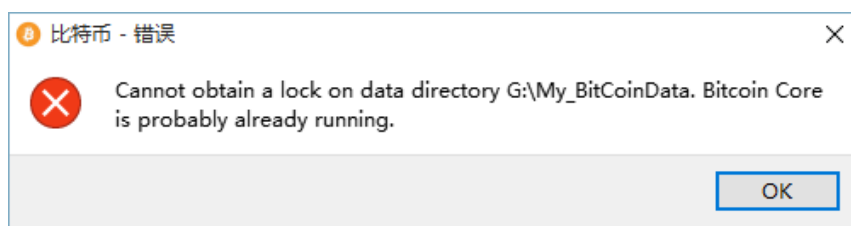
```
bitcoin-qt -conf="c:\mybitcoin.conf"
```

可以发现，另外指定的配置文件，其文件名可以是自定义的。需要注意的是，**bitcoin-qt** 支持的所有参数中，除了 **-datadir** 与 **-conf** 只能是通过命令参数指定外，其他参数都是既可以在命令参数中直接传递，也可以通过在配置文件中指定的。

(2) bitcoind.exe

这个其实就可以看做是不带界面的 **bitcoin-qt.exe**，其中同样包含了比特币的核心节点，并且提供了一个 **RPC** 服务，比特币使用的是 **JSON-RPC** 协议，以便通过命令行交互的方式访问比特币系统的数据，比如访问区块链账本数据，进行钱包操作和系统管理等。

bitcoin-qt 与 **bitcoind** 是互相兼容的，有同样的命令行参数，读取相同格式的配置文件，也读写相同的数据文件，使用的时候，这两个程序，根据需要启动一个即可，同时启动也不会出错，但是同时启动两个 **bitcoin-qt** 或者两个 **bitcoind** 会出错，如下：



图中所示，大致的意思就是对数据文件的访问冲突了。

bitcoind 默认读取的配置文件，在不同操作系统下路径也不尽相同，如下所示：

- ☐ Windows: %APPDATA%\Bitcoin\
- ☐ OSX: \$HOME/Library/Application Support/Bitcoin/
- ☐ Linux: \$HOME/.bitcoin/

除了上述的默认配置路径外，与 **bitcoin-qt** 一样，也是可以在启动的时候通过传递参数来重新指定其他路径下的配置文件或者数据目录的，我们来看下：

```
bitcoind -datadir="c:\bitcoin_data" -conf="C:\mybitcoin.conf"
```

如上所示，启动时，使用 **-datadir** 指定了数据文件需要存储的目录，使用 **-conf** 指定了 C 盘目录下的一个配置文件，此时这个配置文件的名称是自定义命名的。**bitcoind** 启动后可以通过 **bitcoin-cli** 来进行访问，**bitcoin-cli** 的使用在下一节介绍。

看到这里，有些朋友可能会有些疑问，比特币核心客户端运行后可以与其他节点进行互相的连接通信，那就得开放一个服务端口，而访问比特币节点信息又是通过 **RPC** 的方式，那相当于开启

了一个 RPC 服务，这么说来，比特币网络中的每个节点其实就相当于是一个个的服务器，确实如此，这些开启的服务端口说明如下。

- ❑ 8333，用于与其他节点进行通信的监听端口，节点之间的通信是通过 bitcoin protocol 进行的，通过这个端口才能进入比特币的 P2P 网络。
- ❑ 8332，这是提供 JSON-RPC 通信的端口，通过这个端口可以访问节点的数据。
- ❑ 如果是测试网络，分别是 18333 和 18332。

以上端口是可以另外指定的，通过参数-port 与-rpcport 参数可以分别重新指定。

(3) bitcoin-cli.exe

bitcoin-cli 允许你通过命令行发送 RPC 命令到 bitcoind 进行操作，比如 bitcoin-cli help，因此这是一个命令行客户端，用来通过 RPC 方式访问 bitcoind 的 RPC 服务。我们可以通过命令行来查看一下当前的 bitcoin-cli 的版本：

```
bitcoin-cli -version
```

运行后会返回如下这般的描述信息：“Bitcoin Core RPC client version v0.14.2”。通过返回的信息也能看到，bitcoin-cli 就是一个 RPC 客户端工具，那么如何去连接核心客户端呢，首先 bitcoin-cli 与 bitcoind 是使用同样路径下的配置文件（当然这里是指默认情况下，如果各自都使用参数-conf 重新指定了配置文件那就另当别论了），因此在使用 bitcoin-cli 之前，我们需要先运行 bitcoind，然后后来执行 bitcoin-cli 命令：

```
bitcoin-cli getinfo
```

可以看到有如下格式的信息输出：

```
{
  "version": 140100,
  "protocolversion": 70015,
  "walletversion": 130000,
  "balance": 0.00000000,
  "blocks": 48,
  "timeoffset": 0,
  "connections": 0,
  "proxy": "",
```

```
"difficulty": 1,

"testnet": false,

"keypoololdest": 1503043764,

"keypoolsize": 100,

"paytxfee": 0.00000000,

"relayfee": 0.00001000,

"errors": ""

}
```

看到信息的返回，表明已经正常连接且可以访问了，如果想要停止掉 `bitcoind`，则可以发送如下指令：

```
bitcoin-cli stop
```

`bitcoind` 会接收到停止的命令，执行后就退出了运行服务。

我们再来看一个例子，在这个例子中，通过参数重新指定了数据目录和配置文件：

```
bitcoind -datadir="c:\bitcoin_data" -conf="C:\bitcoin.conf"
```

此时，如果仍然要通过 `bitcoin-cli` 来访问这个运行的 `bitcoind`，则需要如下：

```
bitcoin-cli -datadir="c:\bitcoin_data" -conf="c:\bitcoin.conf" getinfo
```

运行后则正常返回了运行的 `bitcoind` 中的信息。

至此，我们可以发现，`bitcoin-qt`、`bitcoind` 以及 `bitcoin-cli` 都能读取相同格式的配置文件，也拥有一样的命令参数，具体支持的各种参数很多，大家可以自行去查阅，这里不再赘述了。再一个就是，比特币中的很多功能调用都是通过 `RPC` 命令提供的，比如区块信息查询、交易事务查询、多重签名使用等等，因此要了解完整功能调用的朋友可以去具体了解一下这些 `RPC` 命令的使用，笔者这里也推荐一些不错的网站方便大家学习使用，如下：

- ❑ <https://blockchain.info>：方便检索各项比特币网络的数据
- ❑ <https://chainquery.com/bitcoin-api>：基于网页的比特币 `RPC` 命令使用

（4）`btcoin-tx.exe`

这是一个独立的工具程序，可以用来创建、解析以及编辑比特币中的交易事务，我们在通常使用比特币系统的时候，使用上述介绍的钱包功能也就足够了，但是如果需要单独来查看或者创建一份交易事务数据，就可以使用这个工具了。那么，既然是用于操作交易事务的，我们就来试一试，

比特币的交易事务在本质上就是一段二进制数据，我们就任意找一段过来，看看 `bitcoin-tx` 能解析成什么样子，为了方便，将二进制的交易事务数据转成十六进制的格式来显示，如下：

```
0100000001e0772cd81114d0993922a280e2b29209d6c6c5d2f22d807018d1ef0d55cfe4041c0
000006a473044022008650b496ea573a2d42efbcbfb49288ab3c7f9968a1fa6072155a028a4de
b39e02201b2dd03307fcd1fbb2f9928a8904d50a84ae9d600986a3a8a125fe248b4faf1001210
354eb6c85025f3abecde8236e86aabf6b819a72154e69d39f7ae591a92436c166fffffffff01d9
38890c000000001976a914fe5d8413d80c3d3f9b975f45990cf432455b13ef88ac00000000
```

这就是一段交易事务的数据，接下来我们就来解析一下，将这段数据转换成容易阅读的格式，为了方便阅读，我们就转换为 JSON 格式，命令如下：

```
bitcoin-tx -json

0100000001e0772cd81114d0993922a280e2b29209d6c6c5d2f22d807018d1ef0d55cfe4041c0
000006a473044022008650b496ea573a2d42efbcbfb49288ab3c7f9968a1fa6072155a028a4de
b39e02201b2dd03307fcd1fbb2f9928a8904d50a84ae9d600986a3a8a125fe248b4faf1001210
354eb6c85025f3abecde8236e86aabf6b819a72154e69d39f7ae591a92436c166fffffffff01d9
38890c000000001976a914fe5d8413d80c3d3f9b975f45990cf432455b13ef88ac00000000
```

执行后，可以得到如下的输出：

```
{
  "txid":
    "2aff308e3e1a9b251ecb701762f6f2c1d28952fe6d0d94efc78880e8a62d2cbb",
  "hash":
    "2aff308e3e1a9b251ecb701762f6f2c1d28952fe6d0d94efc78880e8a62d2cbb",
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid":
        "04e4cf550defd11870802df2d2c5c6d60992b2e280a2223999d01411d82c77e0",
      "vout": 28,
      "scriptSig": {
```

```

        "asm":
"3044022008650b496ea573a2d42efbcbfb49288ab3c7f9968a1fa6072155a028a4deb39e0220
1b2dd03307fcd1fbb2f9928a8904d50a84ae9d600986a3a8a125fe248b4faf10[ALL]
0354eb6c85025f3abecde8236e86aabf6b819a72154e69d39f7ae591a92436c166",

        "hex":
"473044022008650b496ea573a2d42efbcbfb49288ab3c7f9968a1fa6072155a028a4deb39e02
201b2dd03307fcd1fbb2f9928a8904d50a84ae9d600986a3a8a125fe248b4faf1001210354eb6
c85025f3abecde8236e86aabf6b819a72154e69d39f7ae591a92436c166"

    },

    "sequence": 4294967295

}

],

"vout": [

    {

        "value": 2.10319577,

        "n": 0,

        "scriptPubKey": {

            "asm": "OP_DUP OP_HASH160
fe5d8413d80c3d3f9b975f45990cf432455b13ef OP_EQUALVERIFY OP_CHECKSIG",

            "hex": "76a914fe5d8413d80c3d3f9b975f45990cf432455b13ef88ac",

            "reqSigs": 1,

            "type": "pubkeyhash",

            "addresses": [

                "1QBxfKsz2F7xwd66TwMj5wEoLxCQghy54c"

            ]

        }

    }

],

```

```
    "hex":  
  
    "0100000001e0772cd81114d0993922a280e2b29209d6c6c5d2f22d807018d1ef0d55cfe4041c  
0000006a473044022008650b496ea573a2d42efbcbfb49288ab3c7f9968a1fa6072155a028a4d  
eb39e02201b2dd03307fcd1fbb2f9928a8904d50a84ae9d600986a3a8a125fe248b4faf100121  
0354eb6c85025f3abecde8236e86aabf6b819a72154e69d39f7ae591a92436c166fffffffff01d  
938890c000000001976a914fe5d8413d80c3d3f9b975f45990cf432455b13ef88ac00000000"  
  
    }
```

通过输出信息，我们可以很方便的看到其中包含的各个数据组成项，比如 **txid** 是指交易事务的哈希值，这个值与 **hash** 数据项一样；**vin** 是指交易事务中的输入部分；**vout** 是指交易事务中的输出部分，具体每一项的含义这里暂且不多解释，在第八章中我们通过模拟比特币来构建一个最简易的区块链系统，其中有更具体的介绍。通过使用这个工具，除了能解析交易事务数据外，也能创建交易事务，读者朋友们可以去具体尝试使用一下。

(5) test_bitcoin.exe

这是用于比特币程序 **bitcoind** 的单元测试工具，与程序开发相关，除了这个，实际上还有一个用于 **bitcoin-qt** 的单元测试工具 **test_bitcoin-qt**，这些工具普通用户一般用不到，这里不再展开详述。

2. 客户端逻辑结构

通过上述的介绍，我们了解了中本聪客户端程序的基本组成，为了让大家有一个更加清晰的理解，我们来看下，中本聪客户端在逻辑结构上包含了哪些功能模块，见下图：



图中所示的 4 个功能模块，共同组成了称之为全节点的比特币程序结构，我们看到，“挖矿”部分标记了虚线，这是因为在中本聪客户端中没有包含挖矿功能，挖矿是另外独立的程序。钱包的功能，我们已经比较了解了，主要是用于管理用户的密钥以及提供转账操作等功能，是属于比特币的前端部分功能，事实上，钱包功能是可以独立出来的，专门提供一个独立的钱包程序，这部分的

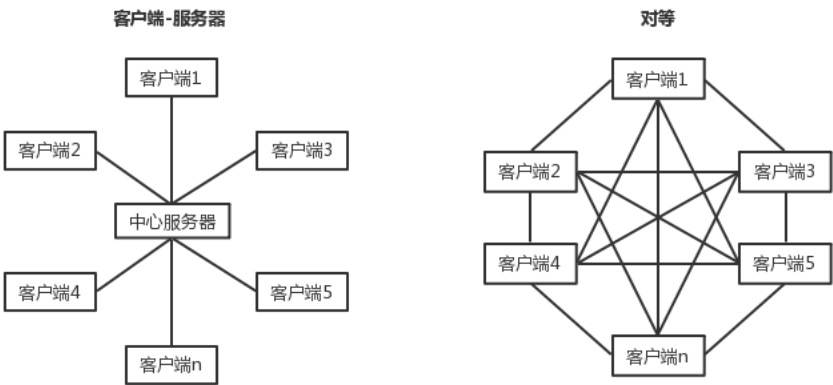
阐述在下面章节中有详细描述。接下来，我们主要对”完整区块链”和”网络路由”部分进行一个说明，刚才说了，钱包只是一个前端功能，那么比特币的后端功能是什么呢，请看下文：

(1) 完整区块链

中本聪客户端保留了完整的区块链账本数据，因此能够独立自主地校验所有交易，而不需借由任何外部的调用。当然，另外一些节点只保留了区块链的一部分比如说区块头，可以通过一种名为“简易支付验证（SPV）”的方式来完成支付验证，这样的节点被称为“SPV 节点”。除了中本聪客户端外，一些挖矿节点也保有了区块链的完整数据拷贝，还有一些参与矿池挖矿的节点是轻量级节点，它们必须依赖矿池服务器维护的全节点进行工作。保有完整区块链数据的节点是非常重要的，比特币网络之所以能够成为一个可信任的去中心化网络，就是依赖于这些全节点，目前很多场合为了方便使用，提供了不少的轻量级节点如轻钱包等，但是这些轻量级节点的正常使用都是要通过全节点才能完成的，是一个依赖关系，如果网络中保有完整区块链数据的节点越来越少，那么比特币网络就会受到影响，无论性能、安全性等都会降低。

(2) 网络路由

比特币网络是属于 P2P 网络架构，P2P 也就是对等的意思，与此相对的是“客户端-服务器”架构，有一个提供服务功能的中心服务器，其他客户端通过调用服务器的功能来完成操作，比如我们通常使用的微信、支付宝、网银等，如果提供商的服务器关闭了，那也就完全没法使用这些软件了。在对等网中，每个节点共同都提供网络服务，不存在任何所谓的中心服务器，也因此对等网络的网络架构中是没有层次的，大家都平等的，每个节点在对外提供服务的同时也在使用网络中其他节点所提供的服务，我们来看下两者的区别示意图：



相信读者朋友也是一目了然了，在“客户端-服务器”网络架构中，总是有一个中心的，一旦中心服务器出了问题，基本等于天塌了，而“对等”网络结构，显然对于中心化服务器这种单点故障结

构有很强的抵抗能力，我们可以看到，“对等”结构中的节点都是可以与其他节点去互联的，而且某个节点出问题也不影响其他节点之间通信，这种结构的好处也是很显然的了。当然，无论是哪种网络结构，底层的网络协议都是一样的，还是 TCP/IP 那一套。

比特币是属于区块链技术的首创应用，其特点就是去中心化或者说是分布式，由比特币节点组成的网络自然也就是属于“对等”网络了，那么既然没有一个服务器，大家彼此如何来认识对方呢，也就是如何来发现其他的节点呢？这是需要通过一个协议的，首先节点会启动一个网络端口（通常是 8333 但也可以参数指定，在上述章节中介绍中本聪客户端时已经说明过），通过这个网络端口与其他已知的节点建立连接，连接时，会发送一条包含认证内容的消息进行“握手”确认，比特币网络中是靠彼此共享节点信息来寻找其他节点的，当一个节点建立与其他节点的连接后，会发送一条包含自身 IP 地址的消息给相邻的节点，而邻居收到后会再次发送给自己的其它邻居，当然节点也不是只能被动的等别人来告诉自己，也可以自己发送请求给其它节点索取这些地址信息，如果与发现的节点之间能够成功的连接，那么就会被记录下来，下次启动时就会自动去寻找上次成功连接过的节点。

简单的说，作为网络路由的功能，比特币节点在失去已有连接时会去发现新节点，同时自己也为其他节点提供连接信息，没有服务器的对等网络就是这么来认识陌生人的。

至此，大家对于比特币的核心客户端就有了一个较为完整的理解了吧。

1.4.3 比特币的发行：挖矿

很多朋友在第一次看到“挖矿”这个词语时都很疑惑包括本人，比特币不是一个软件吗？通过软件来挖矿是什么意思？从字面上来看，应当是通过投入某种工作，然后能得到一个“宝贝”也就是矿，当然了，“挖矿”自然不是我们通常认为的那个挖矿，它只是一套算法，在介绍算法过程前，我们先了解下挖矿在比特币软件中主要都有哪些用途：

- ❑ 抢夺区块打包权
- ❑ 验证交易事务
- ❑ 奖励发行新币
- ❑ 广播新区块

我们知道，比特币是一个对等网络，每个节点都可以独立维护自己的数据副本，那么问题就来了，怎么来保证彼此之间的数据一致呢？既然没有一个中心服务器，自然也就没有一个传统意义上的权威数据来源了，这就得有一个约定的规则，大家共同按照这个规则来进行竞争，谁竞争成功

了谁就有数据的打包权也就是记账权，打包完成后广播给别人，别人只要验证一下有无问题即可，没有问题就存入到自己的数据文件中。这个思路不错，等于就是大家来竞争临时中心服务器的资格，那么比特币中实行了一种什么样的规则呢？那就是被称之为工作量证明（PoW，Proof of Work）的一种算法，其实就是类似于掷骰子的一种游戏，比如说大家约定掷出一个 10 位长度的数字，前面 6 位要都是 0，后面的四位数得小于某个值，看谁先掷出符合要求的数字出来，谁就抢得了打包权。我们来看下比特币中具体是怎么来掷这个骰子的：

1. 难度值

首先，既然是大家都在竞争掷骰子，那掷出来的数字必然是要符合一个难度的，这个难度就是一个门槛，在比特币软件中，规定一个 256 位的整数：

```
x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

为难度1的目标值，在比特币诞生初期，当时的全网算力，大约需要10分钟左右的运算能得到一个符合这个难度1要求的值，这也是我们常常说比特币网络每隔大约10分钟出一个区块的来源，我们在查询创世区块也就是0号区块的信息时，可以看到当时的难度就是1。那么，所谓的符合这个难度为1的要求的值是什么意思呢？就是说通过工作量证明算法也就是比特币中的挖矿算法来计算出一个结果，这个结果要小于这个难度目标值，我们来看下0号区块的难度信息：

```
"nonce": 2083236893,  
"bits": "1d00ffff",  
"difficulty": 1,
```

这些信息可以通过比特币支持的 JSON-RPC 中的 "getblock" 命令方法来获得，其中的 difficulty 就是指难度级别，在 0 号区块的难度值是 1，nonce 是一个随机数，是挖矿计算得到的一个数字，这个等会儿再介绍，bits 是用来存储难度的十六进制目标值的，这个难度目标值是存储在区块的头部的，在源码中被定义为一个 4 字节长度的字段，4 字节也就是 32 位，要用来存储 256 位长度的难度目标值，因此这 256 位长度的值需要经过一个压缩处理后才能放到这个字段中，以这个难度 1 的目标值来说，我们查询区块信息后，看到的值是 "1d00ffff"，那么，这个值是怎么压缩而来的呢？规则其实很简单，我们一共有 4 个字节来存储，这 4 个字节的最高位字节用来存储难度值的有效字节数，什么叫有效字节数？就是从第一个不全为 0 的字节开始的部分，比如难度 1 的值有效位是 0x00FFFF.....（等等，怎么前面有 2 个 0 呢？这是因为在压缩规则中，规定了如果难度值有效位的最高位为 1（大于 0x80），则需要前面补上一个 0x00，这里的最高位是 F，也就是二进制的 1111，

因此是符合这个规则的), 难度 1 的目标值中, 有 4 个字节长度的 0, 减掉这些 0 的长度共 32bit, 剩余 $256-32=224$, 也就是 28 个字节, 加上补的 0x00, 因此, 有效位总计 29 个字节, 29 的 16 进制是 1D, 另外 3 个字节中存储的是目标值有效位的最高 3 个字节, 此时的目标值有效位前面已经加上了 2 个 0, 因此最高 3 个字节为 0x00FFFF, 合起来压缩后的值就是 0x1d00ffff。对于这样的一个压缩后的十六进制 4 字节难度目标值, 前 2 位通常称之为幂或者说指数, 后面 6 位称之为系数。

那么，有朋友问了，压缩是可以，那还原出来呢？我们看个公式：目标值=系数 2 的(指数-3))次方。

我们就以这个 0x1d00fff 为例来说明，系数是后面 6 位也就是 00ffff，指数是前面 2 位也就是 1d，代入进去就是： $0x00ffff2^{(8(0x1d-3))}$ ，计算后得到的值是：

```
0x00000000FFFF000000000000000000000000000000000000000000000000000
```

有朋友可能疑惑了，不对吧，这个跟规定的那个难度 1 的值不一样了啊，精度少了很多，确实是，存储在 `bits` 中的值是一个精度截断的近似值。

我们以 200000 号区块为例，查询一下难度值，得到如下：

```
"nonce": 4158183488,  
"bits": "1a05db8b",  
"difficulty": 2864140.507810974,
```

我们来看看这个 **difficulty** 的值是怎么来的，0 号区块的难度是 1，对应的目标值是 0x1d00ffff，200000 号区块的难度目标值是 0x1a05db8b，将两者的目标值按照上述公式进行转换后然后相除便能得到这个 2864140.507810974 的难度值，我们发现，200000 号区块的 **difficulty** 比 0 号区块的大许多，而 **bits** 的大小却比 0 号区块的小许多，这其实是表明了一个特点，随着全网算力的越来越强，**difficulty** 难度值就会越来越大，而 **bits** 表示的目标值是越来越小的，这两者成反比，目标值越小就越难挖矿。

刚才也提到了，难度值并不是一成不变的，比特币差不多每两周会调整一下新的难度值，因为计算的算力是会变化的，为了维持差不多 10 分钟出一个区块的节奏，难度要跟随算力变化而调整，不得不说比特币的设计还是相当完整的，新难度值的计算公式是这样的：新难度值 = 当前难度值 * (最近的 2016 个区块的实际出块时间 / 20160 分钟)，2016 个区块的意思是，假设按照理论的 10 分钟出一个块，2 周也就是 14 天的时间，应该出 2016 个区块，可以看到实际上就是计算一下实际与理论上的时间差值，弥补上这个差值即可。

2. 挖矿计算

我们了解了难度值的概念，现在来看看挖矿计算具体是怎么个过程，首先，我们说了挖矿是要抢夺区块打包权，那就得收集需要打包进区块的那些交易事务，那这些数据从哪来呢？这里有个概念需要大家注意，打包就像是记账，是把发生的交易事务记录下来存档，但是无论什么时候打包谁打包，在网络中发生的交易是持续不断的，就像企业仓库的进销存业务，无论记账员是一个月还是半个月记一次账，业务是持续进行的，在比特币系统中，每个人通过钱包进行的转账交易数据都会广播到网络中，这些都是属于等待打包的未确认交易数据，这些数据都会放在一个内存池中，总之就是一个缓冲区，当然，这些数据都会被进行基本的验证，用以判断是否是不合法的或者是不符合格式的交易数据。

挖矿程序从内存池中获取用来打包区块的交易数据，接下来就要干活啦，我们来看一下挖矿的计算公式：

```
SHA256 (
    SHA256(version + prev_hash + merkle_root + ntime + nbits + nonce )
) < TARGET
```

SHA256 是一种哈希算法，可以通过对一段数据进行计算后输出一个长度为 256 位的摘要信息，SHA256 在比特币中使用很广泛，不但是用于挖矿计算也用于计算区块的哈希值和交易事务的哈希值，比特币对 SHA256 算法是情有独钟啊，我们看到在这个公式中，是对参数进行两次 SHA256 计算，如果计算出来的值小于那个 TARGET 也就是难度目标值那就算是挖矿成功了。那么，这些参数都是由哪些组成的呢？请看下表：

名称	含义
version	区块的版本号
prev_hash	前一个区块的哈希值
merkle_root	准备打包的交易事务哈希树的根值，也就是梅克尔根
ntime	区块时间戳
nbits	当前难度
nonce	随机数

这些数据字段其实也是区块头的组成部分，将这些参数连接起来，参与 SHA256 的挖矿计算。在这些参数中，版本号是固定的值，前一个区块的哈希值也是固定的值，当前难度也是一个固定的

值，那么要想改变这个公式的计算结果，能改动的参数就只有梅克尔根、区块时间戳和那个随机数了。

1) 尔根是通过交易事务计算出来的，具体过程在下面章节中有介绍，挖矿程序从内存池中获取待打包的交易事务，然后计算出梅克尔根，获取交易事务本身也是有一些优先级规则的，比如根据手续费大小之类，这些细节就不赘述了。

2) 时间戳是指 UNIX 时间戳，用于记录区块的产生时间，我们知道比特币系统是分布式的网络，没有固定的时间服务器，因此每个节点获得的时间戳都可能是不一样的，由此，比特币系统中设置了一个规则：首先新产生区块的时间戳要大于之前 11 个区块的平均时间戳，其次是不超过当前网络时间 2 个小时。所以我们要注意了，因为这个原因，后一个区块的时间戳比前一个区块的时间戳反而小也是可能的。

3) 数是一个可自由取值的数值，取值范围是 $0 \sim 2^{32}$ 次方。

我们可以看到，要通过这样的参数来计算出符合条件的值，基本上也就只能靠暴力计算匹配了，这种不断的执行 SHA256 计算的过程很消耗算力，也因此这个过程被形象的称之为是挖矿，简单的说，挖矿就是重复计算区块头的哈希值，不断修改该参数，直到与难度目标值匹配的一个过程。

一旦匹配成功，就可以广播一个新的区块，其他客户端会验证接收到的新区块是否合法，如果验证通过，就会写入到自己的区块链账本数据中。那么，挖矿的奖励在哪儿呢，不是说矿工成功出一个区块就能得到多少多少的比特币奖励的么？那么这里奖励在哪呢？这个奖励其实是作为一条交易事务包含在区块的交易事务中的，相当于系统给矿工转账了一笔比特币，这种交易事务由于特殊性，通常称之为是 coinbase 交易，这个交易一般是位于区块中的第一条，比特币系统也正是通过这种挖矿奖励的方式发行新的比特币，就像央行发行新钞一样。

这个奖励不是无限的，从 2009 年 1 月创建出第一个区块，每个区块奖励 50 个比特币，然后每 21 万个区块大约 4 年产量减半，到 2012 年 11 月减半为每个区块奖励 25 个比特币，然后在 2016 年 7 月减半为每个新区块奖励 12.5 个比特币。基于这个公式，比特币挖矿奖励逐步减少，直到 2140 年，所有的比特币 (20,999,999.98) 将全部发行完毕，到那个时候挖矿就只能收入一些交易手续费了，彼时，比特币网络是否还能保持运行，我们目前也只能持保留意见，矿工在没有明显的激励情况下，是否还愿意持续保持挖矿以承担区块打包的责任，现在也很难说。

比特币中的挖矿计算基本就是这个过程了，其实还是很简单的，本质上就是利用了 SHA256 计算，有朋友可能有疑问，那第一个区块也就是创世区块是怎么挖出来的？呵呵，很简单，创世区块是硬编码直接写进去的，在比特币的源码中，通过 CreateGenesisBlock 这个方法写入，并且还留

下了一段话："The Times 03/Jan/2009 Chancellor on brink of second bailout for banks";当时正是英国的财政大臣达林被迫考虑第二次出手缓解银行危机的时刻，这句话是泰晤士报当天的头版文章标题。

3. 区块广播

矿工挖出区块后，就进行网络广播，传递给相邻的节点，节点接收到新的区块后会进行一系列的验证，比如区块数据格式是否正确；区块头的哈希值小于目标难度；区块时间戳是否在允许范围之内；区块中第一个交易（且只有第一个）是 `coinbase` 交易；区块中的交易事务是否有效等，总之就是一连串的检测，全部校验通过就把新的区块数据纳入到自己的区块链账本中。如果是挖矿节点接收到信息，就会立即停止当前的挖矿计算，转而进行下一区块的竞争。

比特币的挖矿过程说到这里，不知道有没有朋友会有个疑惑，那就是挖矿算法虽然能够提供工作量证明，表明矿工确实是投入了相当的算力的，但是却不能保证只能是一个矿工能挖到啊，如果在同一时间内多个矿工都计算出了符合条件的值，都拥有了打包权，那以谁的为准呢？当初在想到这个问题的时候，苦思冥想了好久，在考虑着能有什么样机制来进行选择呢？良久过后，无解，看了一下比特币中的解决方案，竟然是那么的简单，人家没用什么复杂的算法，就是让节点自己选择，最终传播最广，处于最长的链中的区块将被保留，因此到底谁的区块会被保留下来，可能还真得看看运气了。这里实际上隐含着一个计算机中的理论的，即 **FLP** 原理，先看下定义：在网络可靠，存在节点失效（即使只有一个）的最小化异步模型系统中，不存在一个可以解决一致性问题的确定性算法。这个就是 **FLP** 原理了。这个其实也很好理解，比如三个人在不同房间投票，虽然三个人彼此之间是可以通电话沟通的，但是经常会有人时不时的睡着，比如 A 投票 0，B 投票 1，C 收到了然后睡着了（类比节点失效了），则 A 和 B 永远无法在有限时间内和 C 共同获得最终的结果。看到这里，我们也就明白了挖矿的作用了，除了发行新的比特币外，主要就是维持网络共识，让每个节点对区块链的数据保持一个最终一致性。

4. 挖矿方式

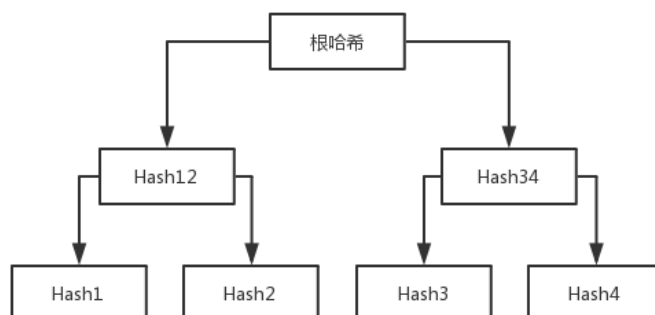
比特币的挖矿过程我们已经了解了，现在给大家介绍下挖矿方式，我们知道，挖矿算法在执行过程中，为了抢夺区块打包权，就得拼命的去算出那个符合难度目标的值，大家都在不断的升级自己的算力，难度也就越来越大，挖矿程序本身不复杂，关键是这个过程非常依赖计算机的算力资源，可以说得算力者得天下，也因为这个原因，挖矿的方式在多年来不断的进化，一切都围绕着为了得到更高的算力来进化。

我们先从硬件类型来说，早期的时候，还没多少人挖矿，难度值也还不大，使用普通的个人电脑就能进行挖矿了，这个时期的硬件设施主要是普通 CPU 挖矿，随着有更多的矿工加入，难度越来越大，使用普通 CPU 的算力，效率开始不够用，于是出现了 GPU 挖矿，利用显卡来进行挖矿计算，GPU 对于 SHA256 的计算性能更高，曾经一段时间，市面上的显卡销量猛增，就是被买去搭建显卡挖矿的，2017 年上半年，另外一种数字加密货币以太坊价格暴涨，也一度引发了市面上的“一卡难求”，显卡尤其是高端显卡的 GPU 计算在一些挖矿算法上的性能表现确实相当不错，然而，对于算力的追求是无止境的，接着又出现了 FPGA（Field-Programmable Gate Array，现场可编程门阵列）和 ASIC（Application Specific Integrated Circuit，特定应用集成电路）这两种挖矿设备，这两类是属于集成电路的装备了，尤其是 ASIC，基本是目前顶级性能的矿机了，专门为了挖矿而设计，只为挖矿而生！

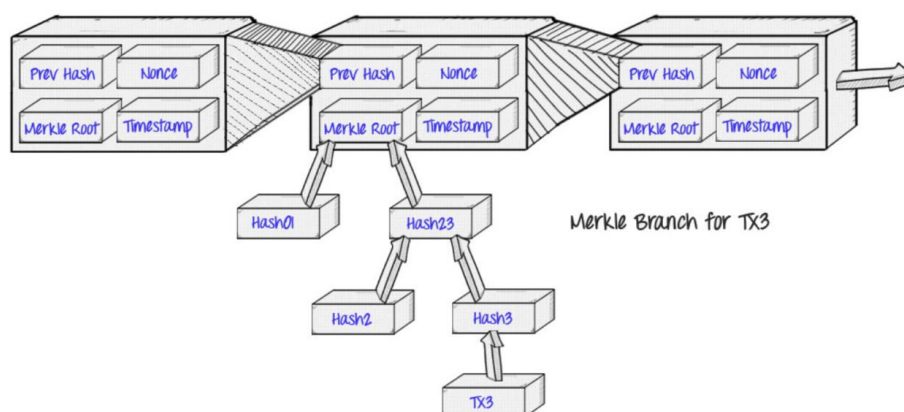
说完了挖矿的装备，我们再来说说挖矿节点的类型，最简单的挖矿节点类型，就是 solo 挖矿也就是个体矿工，自己搞个挖矿装备然后默默的开挖，守株待兔般等待着挖矿成功，如今，在挖矿难度大幅度提升的时代，个人挖矿几乎是一点机会都没有，那么现在流行的挖矿节点是什么类型的呢？那就是矿池，矿池通过挖矿协议协调众多的矿工，相当于就是大家联合起来，每个人都贡献自己的算力，形成一个整体，大大的增强整个挖矿节点网络的算力，个人矿工也可以加入到矿池，他们的挖矿设备在挖矿时保持和矿池服务器的连接，和其他矿工共同分享挖矿任务，之后分享奖励。

1.4.4 数据完整性证明-梅克尔树

在介绍比特币挖矿过程的时候，我们提到过一个概念，叫梅克尔根，是保存在区块头的一个值，并且说这个值是通过区块中的交易事务计算而来的，那么我们就来看看它到底是怎么得到的，我们来看一个示意图：



这幅图是什么意思呢，我们看到，首先这是一个树结构，在底部有 4 个哈希值，我们假设某个区块中一共有 4 条交易事务，那么每条交易事务都计算一个哈希值，分别对应这里的 Hash1 到 Hash4，然后再两两结对再次计算哈希值，以此类推，直到计算出最后一个哈希值，也就是根哈希。这样的一棵树结构就称之为是梅克尔树（merkle tree），而这个根哈希就是梅克尔根（merkle root）。我们再来看一个示意图：



可以看到，每一个区块都是具有一棵梅克尔树结构的，同时可以发现，梅克尔树中的每一个节点都是一个哈希值，因此也可以称之为是哈希树，而比特币中的梅克尔树是通过交易事务的哈希值两两哈希计算而成，所以这样梅克尔树就称之为是二叉梅克尔树，那么这样的一个树结构有什么作用呢？

比特币是一个分布式的网络结构，当一个节点需要同步自己的区块链账本数据时，并没有一个明确的服务器来下载，而是通过与其他节点进行通信实现的，在下载区块数据的时候，难免会有部分数据会损坏，对于这些一条条的交易事务，如何去校验有没有问题呢？这个时候，梅克尔树就能发挥作用了，由于哈希算法的特点，只要参与计算的数据发生一点点的变更，计算出的哈希值就会变掉。我们以第一个示意图来说明，假设 A 通过 B 来同步区块数据，同步完成后，发现计算出的梅克尔根与 B 不一致，也就是有数据发生了损坏，此时先比较 Hash12 和 Hash34 哪个不一致，假如是 Hash12 不一致，则再比较 Hash1 和 Hash2 哪个不一致，如果是 Hash2 不一致，则只要重新下载交易事务 2 就行了，重新下载后，再计算出 Hash12 并与 Hash34 共同计算出新的梅克尔根，如果一致，则说明数据完整。我们发现，通过梅克尔树，可以很快的找到出问题的数据块，而且本来一大块的区块数据可以被切分成小块处理。

梅克尔树的特性还支持实现比特币的轻量级钱包，也就是 SPV(Simplified Payment Verification, 简单支付验证)钱包，具体原理我们在“比特币钱包”一节中介绍。

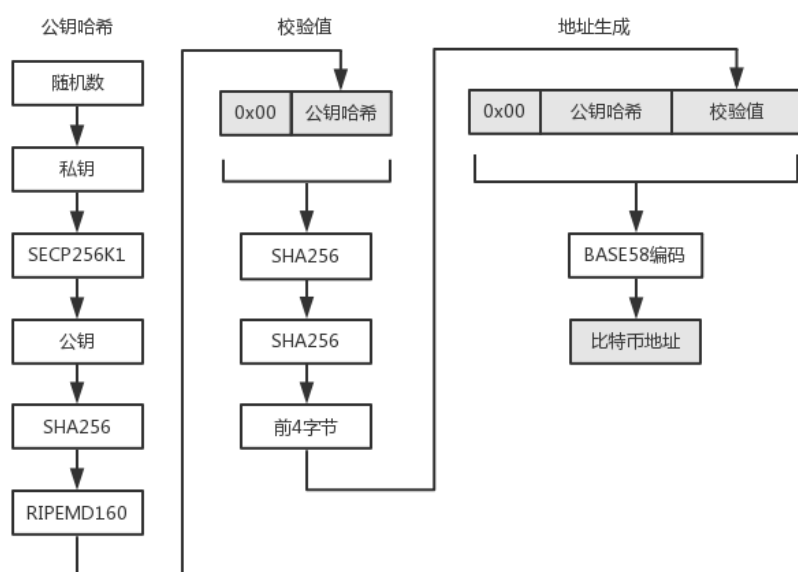
1.4.5 比特币钱包—核心钱包与轻钱包

钱包,是属于比特币系统中的一个前端工具,其最基本的功能就是用来管理用户的比特币地址、发起转账交易、查看交易记录等,在这方面与我们生活中使用的钱包是类似的,一开始的比特币钱包是跟比特币核心客户端一起发布的,在 1.1.2 节中我们介绍比特币核心客户端的时候已经初步做了个了解,这个钱包是比特币核心钱包,其使用过程必须要配合完整的区块链数据副本,因此一般也只适合在桌面端使用。

我们在使用比特币钱包的时候,经常会遇到一个名词:比特币地址。通过钱包转账就是将比特币从一个地址转移到另外一个地址,暂且不论这个转移的过程是什么样的,就这个地址到底是什么意思呢?它又是通过什么来产生的呢?我们先来看一组名词关键字:私钥、公钥和钱包地址。

私钥与公钥是什么意思呢?这是来自于公开密钥算法的概念,我们常说比特币是一种加密数字货币,之所以这么说,是因为比特币的系统设计中巧妙地使用了现代加密算法,而其中一个运用就是生成比特币地址,比特币地址的生成与公开密钥算法密切相关,什么叫公开密钥算法呢?传统的加密算法,其加密和解密方法是对称的,比如凯撒密码,通过将字母移位来加密,比如字母 a 替换成 c、b 替换成 d、d 替换成 f 这样,本来是 abc 的单词就变成了 cdf,然而这种加密算法一旦泄露,别人也就知道了解密算法,换句话说只有一个密钥。针对这种问题,公开密钥算法就应运而生,公开密钥算法是属于一种不对称加密算法,拥有两个密钥,一个是私钥,一个是公钥,公钥可以公开给别人看到,私钥必须要妥善保管,使用私钥加密(通常习惯上将私钥加密称之为是私钥签名)的数据可以用公钥解密,而使用公钥加密的数据可以用私钥解密,两者是互相匹配的。目前使用比较广泛的公开密钥算法主要有 RSA 算法和椭圆曲线加密算法(ECC),RSA 是利用了素数分解难度的原理, ECC 是利用了椭圆曲线离散对数的计算难度,比特币中使用的是椭圆曲线加密算法。

接下来,我们就来看下比特币地址是怎么生成的,为了直观展示,我们看一幅示意图:



这便是比特币地址的生成过程，过程大致是这样的：

1) 首先使用随机数发生器生成一个私钥，私钥在比特币中的作用非常重要，可以用来证明用户的身份，也可以签发交易事务。

2) 私钥经过 **SECP256K1** 算法处理生成了公钥，**SECP256K1** 是一种特定的椭圆曲线算法，需要注意的是，通过算法可以从私钥生成公钥，但是却无法反向的从公钥生成私钥，这也是公钥为什么可以公开的原因。

3) 公钥接下来先使用 **SHA256** 哈希算法计算，再使用 **RIPEMD160** 哈希算法计算，计算出公钥哈希，比特币的代码中通过两次哈希来计算地址值，这样能进一步确保哈希后数值的唯一性，进一步降低不同的数据进行哈希后出现相同的概率。与 **SHA256** 一样，**RIPEMD160** 也是一种 Hash 算法。

4) 将一个地址版本号连接到公钥哈希（比特币主网版本号为“0x00”），然后对其进行两次 **SHA256** 运算，将计算得到的结果取前面 4 字节作为公钥哈希的校验值。

5) 将 0x00 版本号与公钥哈希以及校验值连接起来，然后进行 **BASE58** 编码转换，最终得到了比特币地址。

以上便是比特币地址的生成过程了，我们可以发现比特币的地址其实就是通过公钥转化而来的，将上图简化一下，就是下面这么一个过程：



所以，实际上，在比特币系统中，本质上并没有一个叫做“地址”的东西，因为“地址”是可以通过公钥转化而来的，可以理解为就是公钥的另外一种形式，而公钥又是可以通过私钥计算出来的，因此在比特币钱包中，真正需要妥善保管的是生成的私钥数据，这玩意可千万不能弄丢了，一旦丢失，那可比忘记银行卡密码还麻烦。比特币钱包的主要功能就是保管私钥。

比特币的核心钱包是跟核心客户端在一起的，可以完成创建钱包地址、收发比特币、加密钱包、备份钱包等功能，由于核心钱包是与核心客户端在一起使用的，因此在进行转账交易时，可以进行完整的交易验证，当然付出的代价就是必须得带上那么一大坨的账本数据，到 2017 年 8 月份这份数据已经超过了 130G，而且还在持续不断的增长中，因此并不方便用户的实际使用，实际上除了这一点不方便外，在私钥管理上也有麻烦的地方，通过官方的核心钱包可以无限制的创建自己所需数量的钱包地址，然而这些地址对应的私钥管理也就成了问题，如果不小心损坏了某一个私钥数据，那也就找不回来了，基于这些问题，发展出了新的解决方案。

很多时候，我们在进行支付的时候，只是想通过一个支付验证，知道支付已经成功发起就可以了，对于完整的交易验证（需要在完整的账本数据上校验，比如是否包含足够的余额，是否双花等等）可以交给核心节点，这样就可以将钱包功能部分剥离出来，由此产生了 SPV (Simplified Payment Verification, 简单支付验证) 钱包，事实上这个概念在比特币白皮书中就介绍过了，我们来看下它的原理是什么，SPV 钱包的大致过程如下所示。

- 1) 首先下载完整的区块头数据，注意是区块头不是所有的区块链数据，这样可以大大减少需要获取的账本数据量，区块头中包含有区块的梅克尔根，SPV 方式主要就是靠它来实现的。
- 2) 如果是想要验证某笔支付交易，则计算出这笔交易事务的哈希值 txHash。
- 3) 找到 txHash 所在的区块，验证一下所在区块的区块头是否包含在账本数据中。
- 4) 获得所在区块中计算梅克尔根所需要的哈希值。
- 5) 计算出梅克尔根。
- 6) 若计算结果与所在区块的梅克尔根相等，则支付交易是存在的。
- 7) 根据该区块所处的高度位置，还可以确定该交易得到了多少个确认

我们看到了，SPV 原理的钱包就是使用了梅克尔树来验证支付是否已经发生，这也是为什么称之为是简单支付验证的原因，不过我们也可以发现，支付验证所做的事情很少，仅仅只能看到当

前的支付交易是否被发起而已，并不能保证这笔交易事务最终会进入到主链中，也就是说还需要等待核心节点进行全面的交易验证并且矿工打包到区块后进入主链，在这个过程中是有可能发生失败的，所以 SPV 钱包虽然是带来了便捷性但也是牺牲了安全性的。时至今日，已经出现了各种各样的比特币钱包，在 bitcoin.org 网站上我们可以一见端倪：



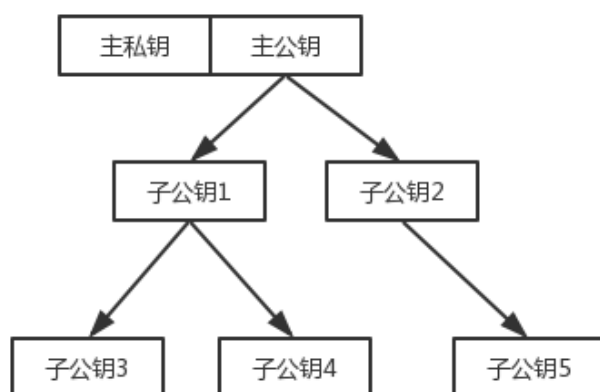
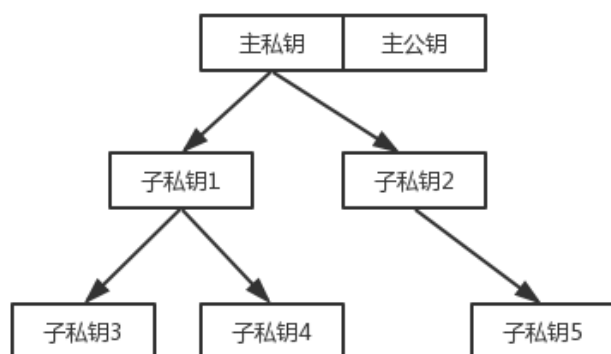
我们可以看到有各种类型的钱包可以使用，通常我们把不需要携带完整账本数据使用的钱包称之为是轻量级钱包，大家在选用自己的钱包时，务必了解清楚钱包的功能和来源，以免遭受不必要的损失。

接下来我们再来介绍一种管理多个私钥的钱包技术，称之为是分层确定性钱包，术语是 Hierarchical Deterministic Wallets，有时也简称为 HD Wallets，这个是在比特币开发的 BIP32 中专门的建议论述（BIP 全称为 Bitcoin Improvement Proposal，是“比特币改进建议”的意思，这是一个面向比特币社区的一个公开的意见簿，BIP32 就是第 32 号建议的意思）。简单的说，分层确定性钱包是具有如下的特点：

- 1) 用一个随机数来生成根私钥，这部分和任何一个比特币钱包生成任何一个私钥没有区别；
- 2) 用一个确定的，不可逆的算法，基于根私钥生成任意数量的子私钥。

比如比特币中使用的 SHA256 就是一个确定不可逆的算法，可以很容易的使用 SHA256 设计出一个 HD 模型：SHA256 (seed+n)，这个就算是类型 1 确定性钱包了。实际上，分层确定性钱包是确定性钱包的一种，目前分层确定性钱包有 Type1、Type2，还有 BIP32 规范几种类型，这些都是为了实现这同一个目的而制定的不同实现方法，基本原理都是类似的。

所谓的分层，除了私钥有主私钥来生成逐层的私钥以外，公钥也一样，通过主公钥生成所有的子公钥，实际上，生成的密钥本身，无论是私钥还是公钥，都还可以作为根来继续生成子密钥，这就是所谓的分层了。注意这里的通过公钥生成子公钥，不需要私钥的参与，无论是主私钥还是子私钥都不需要参与。我们来看一下图示，如下：



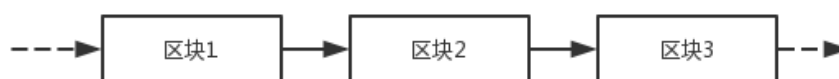
如图所示，通过主私钥和主公钥，可以按规律生成一棵密钥树。

这个特性是非常有用的，在一定程度上，隔离了私钥和公钥，可以带来不少的便捷性，如下。

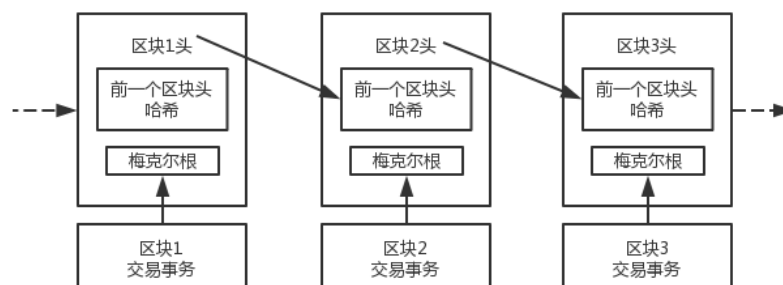
- 1) 备份只需要备份主私钥就行了，新增地址无需再次备份私钥。
- 2) 可以保证主私钥的冷存储，无论增加多少个地址，只需要主公钥就可以了。
- 3) 方便审计，只需要提供主公钥或者某个分支的子公钥，就可以查看下级的数据而又保证不能被交易。
- 4) 有了这棵树，还可以配合权限，设定不同层级的权限，能查看余额还是能交易等。当然啦，便捷性往往都是要牺牲安全性的，缺点很明显，这种钱包，由于私钥之间是具有固定关系的，不那么随机了，因此只要暴露任何一个其中的私钥，再加上主公钥做关联分析，就很有可能使整个树状密钥结构都泄露。

1.4.6 比特币账本结构：区块链

我们在介绍比特币的时候，一直都在提到区块链，通常大家在谈论区块链的时候，往往都是特指这一类技术，比如基于区块链的加密数字货币，基于区块链的分布式交易系统，基于区块链的支付系统等，事实上这个概念是通过比特币带进我们的视野的，它特指一种特有的数据存储结构。区块链，就是“区块+链”，所谓的区块就是指数据块的意思，每一个数据块之间通过某个标志连接起来，从而形成一条链，我们看下示意图：



大家看到这样的格式，可以发现在生活中有很多相似的记录方式，比如工厂仓库的出入库记录或者会计记账，每个月会计将记账凭证汇总为账簿并且月结，每个月都汇总一次，这样一段时间下来，就形成了一个月一个月的连续的账簿，每个月的数据就相当于区块，区块与区块之间通过年月串联起来，比特币中是大约平均每十分钟一个区块，区块中主要包含了打包的交易事务以及区块摘要信息，我们看下比特币中区块链数据的组成示意图：



通过上图我们可以看到比特币的区块数据在逻辑上分成了区块头和区块体，每个区块头中通过梅克尔根关联了区块中众多的交易事务，而每个区块之间通过区块头哈希值串联了起来。这是一个很有趣的数据格式，它将连续不断的發生的数据分成了一个一个的数据块，这样每个区块就都有自己的身份证号（区块头哈希），在下载同步这些数据的时候，可以并行的从各个节点来获得，无论数据先后，到达本地后再根据身份证号组装起来就行，另外，这种格式是一种链条的格式，链条最大的特点就是一环扣一环，很难从中间去破坏，比如有人篡改了中间的2号区块，那么就得同时把2号区块后续的所有区块都要更改掉，这个难度就大了，因为自己更改的数据要发到网络中的其他

节点接受验证，而其他节点是不会验证通过一个被篡改的数据的，这也是区块链数据不可篡改的其中一个原因。

这种格式还有个巧妙的地方，如果这个数据总是由一个人来记录的，那自然也没什么，但是如果放到网络中，大家共同来记录这个数据，那就有点意思了，每个区块数据由谁来记录或者说打包，是有一个规则的，比如说掷骰子，大家约定谁能连续 3 次掷出 6，那就让他来记下一个区块的数据，为了补偿一下他的劳动投入，奖励给他一些收益，比特币正是使用了这样的原理来不断的发行新的比特币出来，奖励给打包的那个人的比特币就是新发行的比特币。

1.4.7 比特币账户模型：UTXO

在这个小节，我们来认识一下比特币中交易事务的数据结构，首先看这个名词 UTXO，这是一个简称，展开后的全称是 Unspent Transaction Output，也就是“未花费事务输出”的意思，老实说，第一次看到这个术语的时候，一时之间真是有些懵，如果说是未花费的余额还能理解，我钱包里有 100，花了 200，还有 800 未花费，这是很符合通常的理解逻辑的，可这个未花费的“事务输出”是个什么意思，实际上，这与比特币中的交易事务结构是很有关系的。

为了让大家更容易理解，我们暂且先不来解析这个交易数据结构，让我们进入到一个仓库，我们知道仓库的主要业务就是进和出，仓库也会把日常的进出流水账记录下来，为了查询统计方便，除了流水账通常还会汇总一份库存表出来，举例如下：

日期	方向	品名	数量	流水编号
2017-8-1	入	毛笔	10	0001
2017-8-2	入	毛笔	20	0002
2017-8-3	入	墨水	15	0003
2017-8-3	出	毛笔	15	0004
2017-8-4	出	墨水	5	0005
2017-8-5	入	毛笔	10	0006

以上图为例，这是从 2017-8-1 到 2017-8-5 之间，仓库记录的出入流水账，为了统计方便，仓库还汇总保存了一份每天的库存日报表，如下：

日期	品名	库存	品名	库存
2017-8-1	毛笔	10	墨水	0

2017-8-2	毛笔	30	墨水	0
2017-8-3	毛笔	15	墨水	15
2017-8-4	毛笔	15	墨水	10
2017-8-5	毛笔	25	墨水	10

每天仓库在需要出库的时候，只要查看一下库存日报表就知道数量是否足够了，比如 2017-8-3 需要出库 15 支毛笔，此时查看库存表发现毛笔的库存量有 30 支，足够发出，于是就将库存表中的毛笔数量减掉 15，并且将出库明细记录在流水账中，然而，这里有一个问题，库存日报表是另外编制保存的，那就有可能发生数据不一致的情况，比如 2017-8-2 时毛笔的库存本来是 30 却误写为 20，这样导致后续的账务就都是错的了，因此在有些系统中，为了防止出现这样的不一致，索性不再另外保存库存表，而只是出一张视图统计（逻辑统计，并非实际去保存这样一个统计表）。

比特币中的交易事务过程与上述的库存进出是很相像的，某个钱包地址中转入了一笔比特币，然后这个地址又向其他钱包地址转出了一笔比特币，这些不断发生的入和出跟仓库的进出是异曲同工的，然而，在比特币中并没有去保存一份“库存表”，每当在“出库”的时候也并不是去“库存表”中进行扣除，而是直接消耗“入库记录”，也就是说在出库的时候就去找有没有之前的入库记录拿来扣除，比如 2017-8-3 时需要出库 15 支毛笔，此时系统就会去搜索之前的入库记录，发现有 2017-8-1 和 2017-8-2 分别有一笔数量为 10 和 20 的入库记录，为了满足 15 的发出数量，首先可以消耗掉 10 的这一笔，然后从 20 的这一笔再消耗掉 5 支，判断成功后，系统会直接产生一条数量为 10 的出库记录和数量为 5 的出库记录，提供这样的方法，将每一笔入和出都对应了起来。

在比特币的交易事务结构中，“入”就是指金额转入，“出”就是指金额转出，为了让大家对这种金额转入转出有一个更加通俗的理解，我们来看一幅示意图：



上图展示了比特币中的交易事务结构，在比特币的交易事务数据中，存储的就是这样的输入和输出，相当于仓库中的进出流水账，并且“输入”和“输出”彼此对应，或者更准确的说，“输入”就是指向之前的“输出”，我们解释一下图中发生的交易事务：

1) 001 号交易为 Coinbase 交易，也就是挖矿交易，在这个交易中，“输入”部分没有对应的“输出”，而是由系统直接奖励发行比特币，矿工 Alice 得到了 12.5 个比特币的奖励，放在 001 号交易的“输出”部分。此时，对于 Alice 来说，拥有了这 12.5 个比特币的支配权，这 12.5 个比特币的输出可以作为下一笔交易的“输入”，所以，顾名思义，这笔“输出”就称之为是 Alice 的未花费输出，也就是 Alice 的 UTXO 的意思。

2) 002 号交易中，Alice 转账 6 比特币到 Bob 的地址，Alice 找到了自己的 UTXO（如果 Alice 不止一笔 UTXO，可以根据一定的规则去选用，比如将小金额的先花费掉），由于只是需要转账 6 比特币，可是 UTXO 中却有 12.5 个，因此需要找零 6.5 个到自己的地址中，由此产生了 002 号中的交易输出，注意，在 002 号交易输出中的 Alice 地址是可以和 001 号中的 Alice 地址不一样的，只要都是属于 Alice 自己的钱包地址就可以。

3) 003 号交易中，Bob 转账了 2 比特币到 Lily 的地址，过程与 002 号交易相同，就不再赘述了。

相信大家看到这里，已经基本理解了所谓的 UTXO 是什么意思，我们再来总结一下：

-
- 1) 比特币的交易中不是通过账户的增减来实现的，而是一笔笔关联的输入输出交易事务。
 - 2) 每一笔的交易都要花费“输入”，然后产生“输出”，这个产生的“输出”就是所谓的“未花费过的交易输出”也就是 UTXO，每一笔交易事务都有一个唯一的编号，称之为是交易事务 ID，这是通过哈希算法计算而来的，当需要引用某一笔交易事务中的“输出”时，主要提供交易事务 ID 和所处“输出”列表中的序号就可以了。
 - 3) 由于没有一个账户的概念，因此当“输入”部分的金额大于所需的“输出”时，必须给自己找零，这个找零也是作为交易的一部分包含在“输出”中。

有朋友会问：这个 UTXO 的意思是明白了，可是就这么一条条的“输入”和“输出”，怎么证明哪一条 UTXO 是属于谁的呢？在比特币中，是使用输入脚本和输出脚本程序实现的，有时候也称为“锁定脚本”和“解锁脚本”，简单的说，就是通过“锁定脚本”，利用私钥签名解锁自己的某一条 UTXO（也就是之前的“输出”），然后使用对方的公钥锁定新的“输出”，成功后，这笔新的“输出”就成为了对方的 UTXO，对方也可以同样的使用“锁定脚本”和“解锁脚本”来实现转账。这个脚本程序其实本质上就可以看成是比特币中的数字合约，这也是为什么比特币被称为是可编程数字货币的原因，它的转入转出或者说输入输出是通过脚本程序的组合来自动实现的，实现过程中还使用到了私钥和公钥，也就是公开密钥算法，所以比特币还称为可编程加密数字货币。

1.4.8 动手编译比特币源码

如果有人一直在跟你说有个煎饼多好吃，芝麻有多香，鸡蛋有多金黄，你肯定希望去看一看；如果有人一直在跟你说有首歌曲多动人，旋律有多美，歌词有多感人，你肯定希望去听一听；如果.....是的，咱们说了那么多的概念，技术名词，界面也瞧过了，可是这么一个软件到底是怎么编译出来的呢？无论您是不是程序员，都可以感受一下这个过程，看看这个设计巧妙的软件是怎么个通过源代码生成可执行程序的。

比特币的源码是公开的，并且维护在 GitHub 网站上，GitHub 是一个源码托管平台，因为是以 git 作为版本库格式进行托管的，所以名为 GitHub。比特币的源码就托管在 <https://github.com/bitcoin/bitcoin> 这个地址下，目前这个源码由比特币基金会进行维护。版权类型是 MIT，这是一个很松散的版权协议，也因此每一个对比特币源码感兴趣的人都可以自由的去复制、修改，以进行学习研究。打开网页后，可以看到有详细的程序源码以及附带的文档说明，我们就从这里下载源码进行编译。在说明编译步骤之前，先介绍些概要前提吧，烹调大餐前得先看个菜单不是。首先，比特币的源码是使用 C++ 语言开发的，因此想要深入研究源码的朋友们，最好要有不

错的 C++ 基础; 其次, 源码中使用了其他很多开源库, 比如 `libssl-dev`、`libevent-dev`、`libboost-all-dev` 等, 因此编译的时候也需要先安装这些第三方的依赖; 另外, 比特币源码在 Linux 系统上进行编译最方便, 很多依赖库都是先天开发在 Linux 平台的, 当然其他系统上也可以进行编译。

好了, 接下来, 我们就开始这道大餐吧

1. 准备操作系统环境

这里我们使用 Ubuntu16.04LTS 桌面版, 关于 Ubuntu 的安装就不在这里赘述啦, 物理安装或者用虚拟机加载安装都可以, 装好系统后, 首先更新一下系统的软件源。

```
sudo apt-get update
```

2. 获得源码

先来看下获得源码的命令:

```
sudo apt-get install git  
  
mkdir ~/bitcoinsource  
  
git clone https://github.com/bitcoin/bitcoin.git "~/bitcoinsource"
```

1) 第 1 条命令是安装 `git` 命令工具, 这个 `git` 工具是用来从 `github` 上下载源码的, 事实上, 使用 `git` 工具不但可以下载源码, 也可以在本机创建自己的版本库;

2) 第 2 条命令是在当前用户的目录下创建一个文件夹, 用以保存即将下载的比特币源码, 读者朋友具体操作时, 可以自行决定路径和文件夹名称;

3) 第 3 条命令就是从 `GitHub` 上下载比特币的源码到创建的 `bitcoinsource` 目录中。这里有个问题需要注意, 如果在 `git clone` 过程中终止了, 当再次进行 `clone` 时会出错, 一般会有这样的提示:

```
git clone:GnuTLS recv error (-9):A TLS packet with unexpected length was received
```

这个错的原因是因为 `git clone` 并不支持断续下载, 删除目录后重新创建一个新目录再 `clone` 就可以了。

除了上述的 `git clone` 命令方法外, 实际上, 我们可以在 `GitHub` 上直接下载源码压缩包, 下载下来的文件名一般为 `bitcoin-master.zip`, 然后解压缩即可:

```
unzip bitcoin-master.zip
```

解压缩后, 将当前工作目录 `cd` 到 `bitcoin-master` 中, 至此就可以开始着手编译了。

3. 安装依赖库

工欲善其事必先利其器，比特币源码中使用了很多第三方的功能库，这些都是必须的依赖，正所谓一个好汉三个帮，一个篱笆三个桩，没有这些可以自由方便使用的库，使用 C++ 开发比特币软件就要复杂不少。

比如，以下 3 行命令主要安装 C++ 编译器和 make 工具：

```
sudo apt-get install make  
  
sudo apt-get install gcc  
  
sudo apt-get install g++
```

比如，以下命令主要是安装依赖库：

```
sudo apt-get install build-essential  
  
sudo apt-get install libtool  
  
sudo apt-get install autotools-dev  
  
sudo apt-get install autoconf  
  
sudo apt-get install pkg-config  
  
sudo apt-get install libssl-dev  
  
sudo apt-get install libevent-dev  
  
sudo apt-get install libboost-all-dev  
  
sudo apt-get install libminiupnpc-dev  
  
sudo apt-get install libqt4-dev  
  
sudo apt-get install libprotobuf-dev  
  
sudo apt-get install protobuf-compiler  
  
sudo apt-get install libqrencode-dev
```

libevent-dev 是一个网络库，实现网络通信功能的；**libssl-dev** 是一个密码算法库，提供了随机数生成，椭圆曲线密码算法等功能；**libboost-all-dev** 是一个 C++ 工具库，提供各种 C++ 调用的基础功能库如多线程调用以及一些有用的数据结构等；**libqt4-dev** 是一个跨平台的 C++ 库，用于实现跨平台运行的软件界面；以及其他等等，都是比特币源码中需要用到的功能依赖库。值得一提的是，这些依赖库也都是开源的，也就是说，比特币源码不但本身是自由开源的，使用的其他的依赖库也是自由开源的，这样就方便了那些希望对比特币源码进行深入研究的朋友，可以对每一个实现细节

细嚼慢咽的，尽情的去学习研究。

这两行命令主要安装比特币需要用到的数据存储驱动，其使用的类型是 **Berkeley DB**，是一种开源的文件数据库，到这里为止，就万事俱备只欠东风啦，该准备的材料都准备好了。

```
sudo apt-get install libdb-dev  
sudo apt-get install libdb++-dev
```

4. 编译准备

这两个步骤是使用 **make** 工具进行编译的准备工作。

```
./autogen.sh  
./configure
```

需要注意的是，在执行 **./configure** 的时候，有可能会看到这样的提示，如下：

```
configure: error: Found Berkeley DB other than 4.8, required for portable wallets  
(--with-incompatible-bdb to ignore or --disable-wallet to disable wallet  
functionality)
```

看提示是 **configure** 命令执行时出的问题，大概的意思就是发现 **Berkeldy DB** 的版本高于了 4.8，我们在安装 **Berkeley DB** 的时候，命令下载安装的是最新版本，这个其实就是个警告而已，没什么影响，提示中也给出了解决方法，在 **configure** 的命令后面加上一个参数 **--with-incompatible-bdb** 就可以了。

```
./configure --with-incompatible-bdb
```

执行完毕就可以了，接下来的工作就简单啦，直接 **make** 编译安装即可。

5. 编译安装

```
make  
sudo make install
```

执行完毕后，就大功告成啦，接下来我们就可以运行比特币客户端程序啦，我们可以运行带界面的程序试试，经过这个步骤，在源码目录 **src/qt/**下生成了可执行程序，同时安装到了 **/usr/local/bin** 目录下。

6. 运行测试

输入以下命令：

```
bitcoin-qt
```

激动人心的时刻就来临啦！我们可以看到比特币的界面显示出来了，当然了，也可以去尝试运行 `bitcoind` 程序。至此，在 Ubuntu 操作系统上编译比特币源码就结束了。限于篇幅，在其他操作系统比如 Mac、Windows 上的编译过程就不再赘述了，读者朋友如果感兴趣，也可以参考比特币源码中，`doc` 文件夹下面的 `build-osx.md` 和 `build-windows.md` 的文件说明，分别是尝试在 Windows 和 MacOS 系统上的编译。

7. 使用 IDE 管理源码

按理说到这里也没什么可说的了，编译完成了，运行也可以了，不过有木有觉得哪里不太爽呢？对了，缺少一个 IDE（Integrated Development Environment，集成开发管理）啊，这么多的文件，用文本编辑器一个个看，要看花眼了。好，接下来我们就安装一个 IDE 工具来管理这些源码，比特币系统是使用 C++ 开发的，图形界面部分使用的又是 qt 组件，那就选择 QtCreator 吧，本身也开源，而且跨平台，对 C++ 的编译支持也非常好。由于上述的源码编译是在 Ubuntu 下进行的，因此，我们仍然在 Ubuntu 下进行安装设置，咱们还是按照步骤来一步步说明吧。

（1）准备 QtCreator

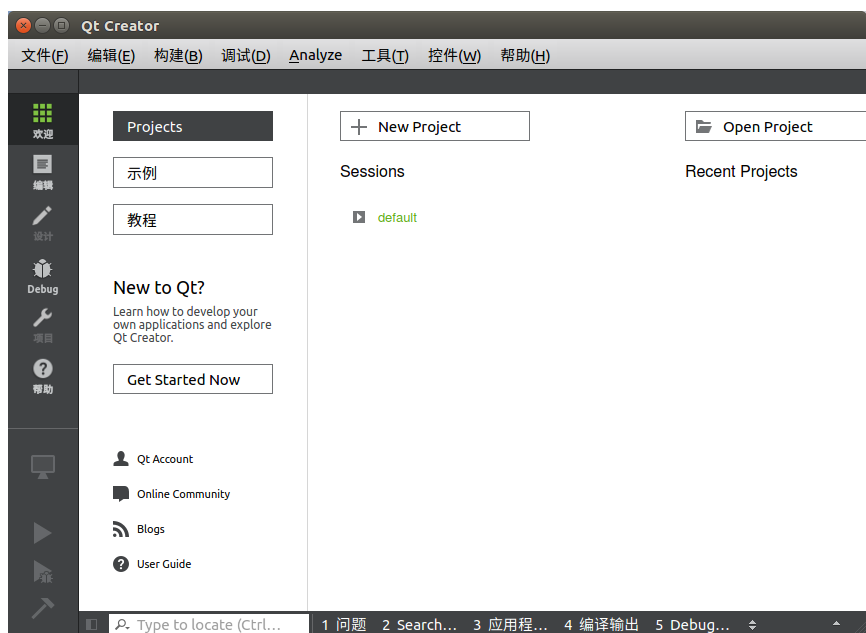
可以直接到 QtCreator 官网下载，Qt 分为商业版和开源版本，我们使用开源版本即可，下载后得到一个文件 `qt-opensource-linux-x64-5.6.2.run`，读者朋友自己下载的时候，还可以选择在线安装版和离线安装版，这里下载的是离线安装版，进入到文件所在的目录，执行如下命令

```
chmod +x qt-opensource-linux-x64-5.6.2.run  
./qt-opensource-linux-x64-5.6.2.run
```

第 1 行命令是给安装文件赋上一个执行权限。

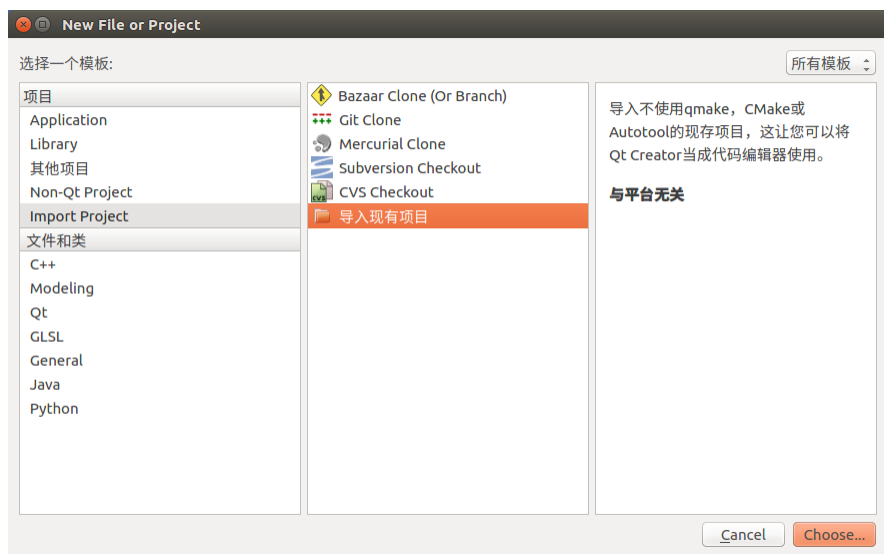
第 2 行命令是执行安装。

安装完毕后，可以打开 QtCreator，见到如下界面



(2) 导入源码项目

在 QtCreator 的菜单栏，点击“文件”→“新建文件或项目”，会弹出一个向导窗体，选择其中的 Import Project，并选中右侧的“导入现有项目”，如下图所示。



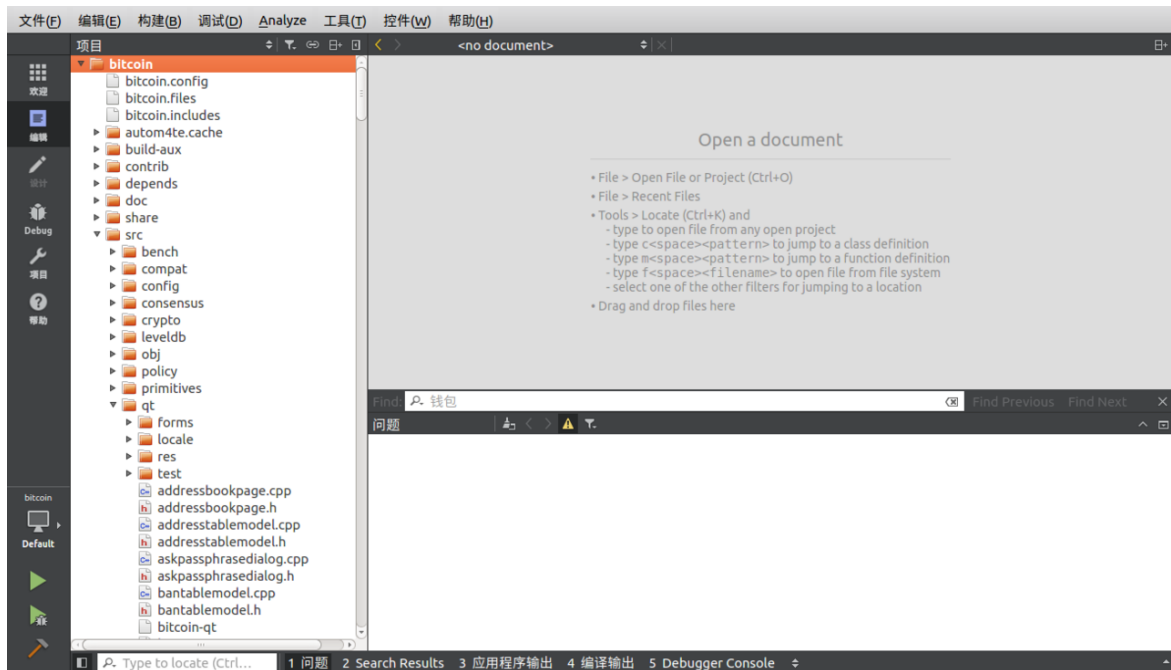
接下来就是选择我们的比特币源码所在目录，也就是需要导入的项目。



图中的“项目名称”可以任意起名，“位置”就是比特币源码所在的目录。选择完毕后继续。



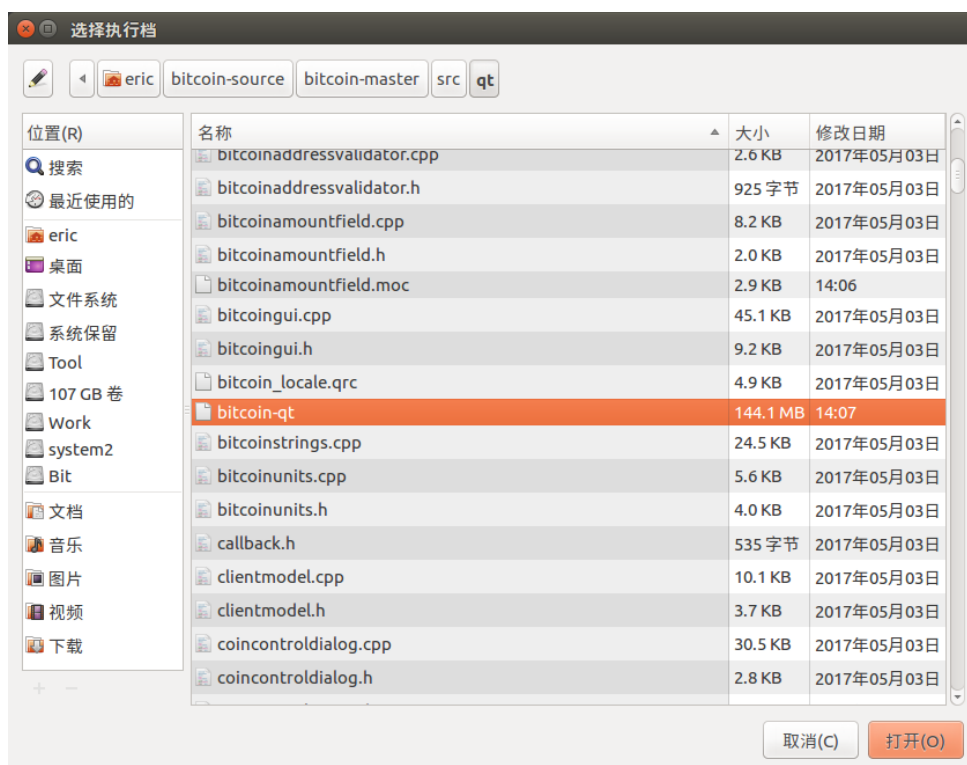
这个界面主要是用于选择一个源码版本控制系统，可以根据自己需要选用，这里只是演示，因此不作选择，直接就可以完成操作，源码导入完毕后，在 QtCreator 中的展现如下



可以看到，在左侧已经列出了源码的文件列表，src 目录下的就是所有的代码文件了，可以看到，根据不同的代码功能，划分了不同的目录，具体细节这里就不赘述了。到了这一步，可以运行一下试一试，点击运行按钮，咣当，咋回事，弹出个啥？



这是要选择个执行程序，比如 bitcoin-qt、bitcoind 等，我们通过这个对话框选择 bitcoin-qt，如下图所示



注意，这里选择的执行程序是在 `src/qt/` 目录下，该目录下的执行程序是通过源码编译直接生成的。

选择后，点击运行，可以看到熟悉的界面又出来了，这样我们就使用 QtCreator 将比特币的源码管理起来了，通过 IDE 工具查看源码要方便许多，感兴趣的朋友也可以尝试着修改其中的界面文件或者源码文件，体会一把编译调试的乐趣。

小提示：

- ① 我们在使用 QtCreator 引入的源码目录，是之前已经经过了一系列步骤编译过的，因此依赖库都已经具备了，执行程序也已经生成了，QtCreator 就像一个外壳，只是做了一个导入集成。
- ② 比特币是一个一直在发展的开源项目，在参照以上步骤进行操作的时候，一定要注意选择的版本是否一致或者兼容，就本文中，选用的操作系统是 Ubuntu16.04 LTS 桌面版，下载的比特币源码版本是 v0.14，使用的 QtCreator 是 4.0.3。

1.5 区块链的技术意义

要了解区块链的技术意义，我们只要来整理一下区块链系统的特点就行了，实际上这些特点在比特币中也已经体现了出来，我们来一一说明一下吧：

1. 数据不可篡改性

为什么数据不可篡改呢？首先区块链系统不是一个中心化的软件设施，比如说比特币，如果是一个单机软件，或者说是被某一个人某一家机构控制的，那肯定是谈不上数据不可篡改的，至少在技术上是不能保证的，而比特币是一个 P2P 的对等网络结构软件，没有服务器，数据是每个节点各自存储一份，自己最多把自己节点上的数据改掉，然而只是这样的话是得不到整个网络的承认的，无法被其他节点验证通过，修改后的数据也就无法被打包到区块中了（51%攻击什么的暂且不提，只是另外一个话题了），不但如此，如果一个数据被打包进区块后，后续又连续确认了多个区块，比如数据是被打包进 5 号区块的，现在整个区块链账本的区块高度是 10 号，那么想要更改掉当时的数据的话就更难了，因为这个时候不单单是要改掉 5 号区块的数据，后续的区块都要变动，因为区块之间是通过区块哈希连接起来的，更改了某个区块的数据后，后续的区块就都要更改了，因此想要篡改的难度就更大了。有这个特点，在很多领域内就很有意义了，比如说金融行业的业务数据、公众政务数据、审计数据等，这些行业的数据都是要有严格的防篡改要求的。

2. 分布式存储

对于一个软件系统，一个需要保存数据的软件系统，最担心的是什么？毫无疑问是数据丢失。传统的软件设计架构，再怎么考虑数据备份或者说数据库集群等，也总是不能很好的保证数据的安全，要么就是需要运营者投入大量数据备份设备或者数据库集群设施等，无论如何也不是一个廉价简约的方案。在区块链系统中，每个运行的节点都拥有一份完整的数据副本，这样的设计不但是使得数据存储避免了单故障点的问题，还可以让每个节点能够独立的验证和检索数据，大大增加了整个系统的可靠性，节点之间的数据副本还可以互相保持同步并使用类似梅克尔树这样的技术结构保证数据的完整性和一致性。这种分布式的结构很适合用在那些面向公众的服务型软件设施上，避免集中而昂贵的专用服务器配备，也具备相当良好的数据安全性。

3. 匿名性

我们在使用传统的服务软件时，通常都是需要注册一个用户名，绑定手机号、邮箱什么的，为了加强用户识别的准确性，还会要求进行实名认证、视频认证之类，然而在区块链系统中，目前几乎所有的区块链产品，都是使用所谓的地址来标识用户的，仅此而已，不再需要提供其他任何能标识出用户身份的信息，地址通常是通过公开密钥算法生成的公钥转换而来的，这通常就是一串如乱

码一般的字符串，因此，虽然比特币、以太坊等这些公链系统的数据是完全公开透明的，可我们却并不能知道背后的操作者是谁，不但如此，每个使用者还可以创建任意数量的地址，只要你愿意，可以每一次都使用不同的地址来进行转账等各种操作，也可以将自己的资产分散在众多的地址上，这就实现了一种用户身份的匿名性。那么，匿名性有什么用途呢？我们知道，在很多场合，比如转账支付，比如收款，比如创建链上资产等等，这些都是比较隐私的行为，匿名性在很大程度上可以满足这些隐私安全的需求，不但是对个人，对于企业这样的商业环境，也是有同样的需求的。

4. 价值传递

这大概是区块链系统中最重要的一個特性了，所谓价值，就是泛指各种资产，比如货币资产、信用资产、版权资产以及各种实物资产如黄金等，所有这些资产在本质上其实都是一种信用或者说是信任，比如货币，我们之所以愿意通过劳动来获取货币，是因为相信使用这些货币可以交换到需要的商品，这种信任是由政府担保的，也因为有这种信任的担保，出现了各种本身不具备太多价值但是却附带有信用价值的资产形式比如纸币、支票、汇票等，再来说版权，一首歌曲、一幅画、一本书等等，都具有版权，版权也是一种资产，是具有价值的，这个价值从何而来，版权的背后是创作者的劳动投入，这个就是价值，国家通过法律保护这种价值，因此就能够在市场上流转传递了。我们可以发现，在这些价值资产的转移过程中，都需要一个重要的条件，那就是信用的保证，我们需要政府，需要银行，需要担保公司等等这些第三方的机构来提供信用保证，只有在这些信用保证的前提下，我们才能完成各种交易，即便是互联网发展起来后，虽然可以通过互联网方便的传输各种数据（图片、视频、音乐等），但是这些数据的价值却仍然需要这些第三方来保证，互联网本身并不具备价值保护的机制，而如果要在全世界范围内进行交易那就更麻烦了，涉及不同的法律，不同的价值认定规则，不同的支付方式等等，那就得需要更多的机构来提供交易的保证，代价昂贵且相当麻烦。

区块链能改变这种情况么？我们先看比特币这个例子，比特币是一种数字资产，它是由比特币软件组成的网络来维护的，在这个网络中，不需要其他的第三方，自己可以根据规则发行比特币，并且能确保发行的比特币是具有价值的（工作量证明），而这种价值的认定是通过网络中所有的节点来自动进行验证的，节点之间达成共识就算是认可了，整个过程都是自成一个体系来运行的，人们在转账交易比特币的时候，价值就发生了传递，而如果将其他的资产比如股票、合同、版权、债权等锚定比特币，那么其他的资产也就能方便的进行传递转移了。我们可以发现，区块链系统是可以自己创造信任机制的，在这样一个无需第三方的信任环境中，可以大大简化各种资产交易的过程，

降低交易成本，并且由于区块链系统是一个分布式的系统，节点可以遍布全球，那就可以实现无边界的价值传递了，这是一个多么令人惊叹的一个特性，全球各地的人们都可以在这样一个系统中方便快捷的进行各种资产交易，这才是真正的全球经济一体化呀！

5. 自动网络共识

日常生活中，我们有很多事情需要双方或者多方达成共识，比如签订一份买卖合同，买入一笔债权，担保一份交易或者购房按期还贷、众筹资金管理等，在传统的模式中，这些需求是如何提供服务的呢？比如签订合同，那就需要双方签名，必要时还需要律师审阅，公证处公正；比如担保交易，除了签名外还需要提供资产余额证明；比如购房还贷，需要有收入证明同时也需要还贷者签名；再比如众筹的资金管理，就更复杂了，需要记录每个参与者的资金项，还需要跟踪众筹资金的流向。凡此等等，这些事情在达成共识的过程中，都需要做各种确认，如果有注册过公司的朋友，还会有一个很深的印象，在办理过程中，需要不断的跑工商、跑银行、跑税务局，银行还要跑人民银行和开户行，税务局还要跑国税和地税，在这些窗口办理时，他们都需要一件事情，那就是填写你的身份信息，公司信息，因为他们都要证明你的身份以此才能达成一个双方的共识。

这种共识可以通过一个网络来自动的进行吗？如果可以，那该省多少事啊。我们还是来看比特币的例子，比特币从发行到转账交易，都是由网络中的节点自动进行身份认证和一系列的检查的，检查通过后就达成了网络共识，一笔交易就算是确定了，各个不同的节点之间达成共识的过程不再需要我们去签名，去按指纹或者去打一份什么证明了，因为每个节点都遵守一份共同的约定规则，只要一项交易符合所有的约定规则那就能被确认，每个节点都确认，大家就一致认同了。那么，除了比特币这种转账交易可以自动达成共识外，其他的事务也可以吗？当然是可以的，上述提到的各种商业或者金融活动，都可以通过区块链上的智能合约来实现，区块链系统中的各个节点独立的验证智能合约，共同达成共识，如果能将这种机制应用到商业、金融、政务等领域，那将提高多少效率啊。

6. 可编程合约

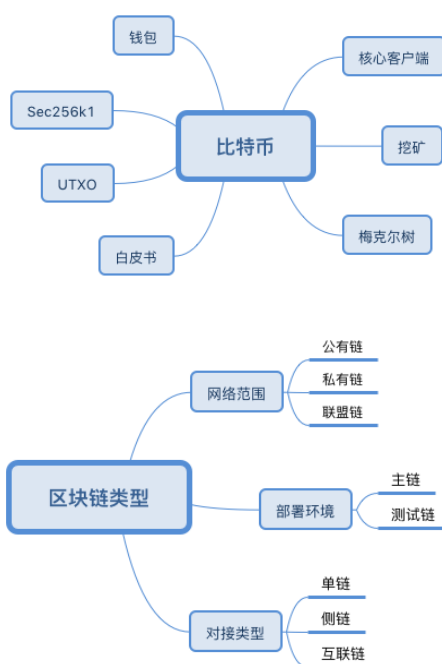
可编程合约，也就是智能合约的意思，以比特币为例，如果用一种更加技术的称呼来描述比特币的话，可以叫做可编程加密数字货币，这个可编程是什么意思呢？在比特币系统中，并不是像银行账户一样，将金额存储在某个账户下就表明一笔资产是某个账户拥有的，而是使用了一种脚本程

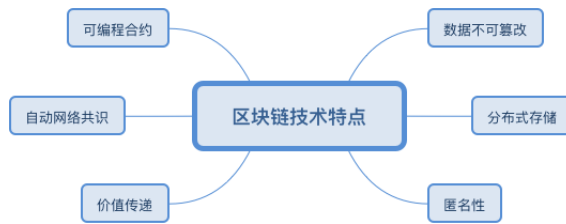
序，通过脚本程序解锁和锁定一笔资产，简单的说，就是让资产具备更强的编程可控制能力，拥有密钥的用户可以提交自己的签名信息让脚本来验证身份，以证明自己对资产的所有权，并且可以通过程序设定对资产的管理方式，比如设定一笔资产需要多个人共同签名才能被转移或者需要达到某个条件的时候才能被使用等，这种可配置可控制的思想就是可编程合约的思想，比特币展示了这种新颖的思路，在后续的发展中，以太坊扩展了这个思路，使可编程合约进一步发扬光大，不但支持加密数字货币，还支持更复杂的金融及商业合约编程，比如众筹、担保等，这种合约使用脚本语言进行开发，部署到区块链后就很难被更改了，也就是所谓的代码即法律，区块链系统具有数据的不可篡改性、价值传递能力，加上可编程合约，就能完全的支持商业环境下的各种合约需求，无论合约中有哪些条条框框，写在纸上不如写在代码中，部署在区块链上，公正透明而且能够刚性执行，更主要的是，这样的合约可以覆盖全世界，以为脚本编写的合约是不分国界的。

1.6 知识点导图

比特币是区块链技术的一种应用，而区块链的概念也是通过比特币带出来的，因此要理解区块链技术，可以从理解比特币开始，后续的各种其他区块链基本可以看作是在比特币基础上的衍生扩展，比特币被设计为是一种数字货币，但是对于一类技术而言，区块链技术的应用场景远不止是数字货币，我们可以结合技术特点，思考在各个领域中可能的应用。

我们来看下本章的思维导图，作为给大家的总结。





参考资料

[1] <https://bitcoin.org/en/developer-documentation>

[2] <https://blockchain.info>