Table of Contents

区块链(Blockchain)

区块链（Blockchain）是由节点参与的分布式系统，它的特点是不可更改，不可伪造，也可以将其理解为账簿系统(ledger)。它是比特币的一个重要概念，完整比特币区块链的副本，记录了其代币（token）的每一笔交易。通过这些信息，我们可以找到每一个地址，在历史上任何一点所拥有的价值。

如何简单易懂地介绍区块链？

链接：https://www.zhihu.com/question/37290469/answer/107612456

事情是这样的，最近我的室友Hasaki一直在问我区块链和比特币的事情，我尝试了很多种不通的姿势以求简单通俗形象生动地跟他解释什么是区块链技术，但是最后都失败了。因此我萌生了要写一篇BlockChain for Babies（又名：如何向你的弱智室友解释区块链）的想法，以求能简单直观生动形象地向对区块链技术不了解但是想知道区块链是什么的人介绍区块链技术或者比特币。

因为面向的读者是不想知道具体技术实现只想了解区块链的人群，因此本文避开了一些底层和算法细节，采用比较主观的方式来展示笔者对区块链技术的感性认识。如果你只是对区块链感兴趣，并没有深入学习的打算，或者只是想像我一样在别人问起来的时候装逼，本文应该是一篇很好的“导论”。

总览

区块链本质上是一个去中心化的分布式账本数据库（感谢@程剑宇指出：在与比特币相关的区块链应用中可使用这一术语，但区块链技术可能并不包含“账本”）。其本身是一串使用密码学相关联所产生的数据块，每一个数据块中包含了多次比特币网络交易有效确认的信息。

这是区块链的定义，因此要逐步了解区块链，我们需要一步步了解如下东西。

去中心化

先来考虑一个中心化集中式处理的过程。你要在某宝上买一部手机，交易流程是：你将钱打给支付宝－支付宝收款后通知卖家发货－卖家发货－你确认收货－支付宝把钱打给卖家。

<

img src="https://pic3.zhimg.com/50/fd044856b872644c8629402a034afcf1_hd.jpg" data-rawwidth="588" data-rawheight="483" class="origin_image zh-lightbox-thumb" width="588" data-original="https://pic3.zhimg.com/fd044856b872644c8629402a034afcf1_r.jpg"

>

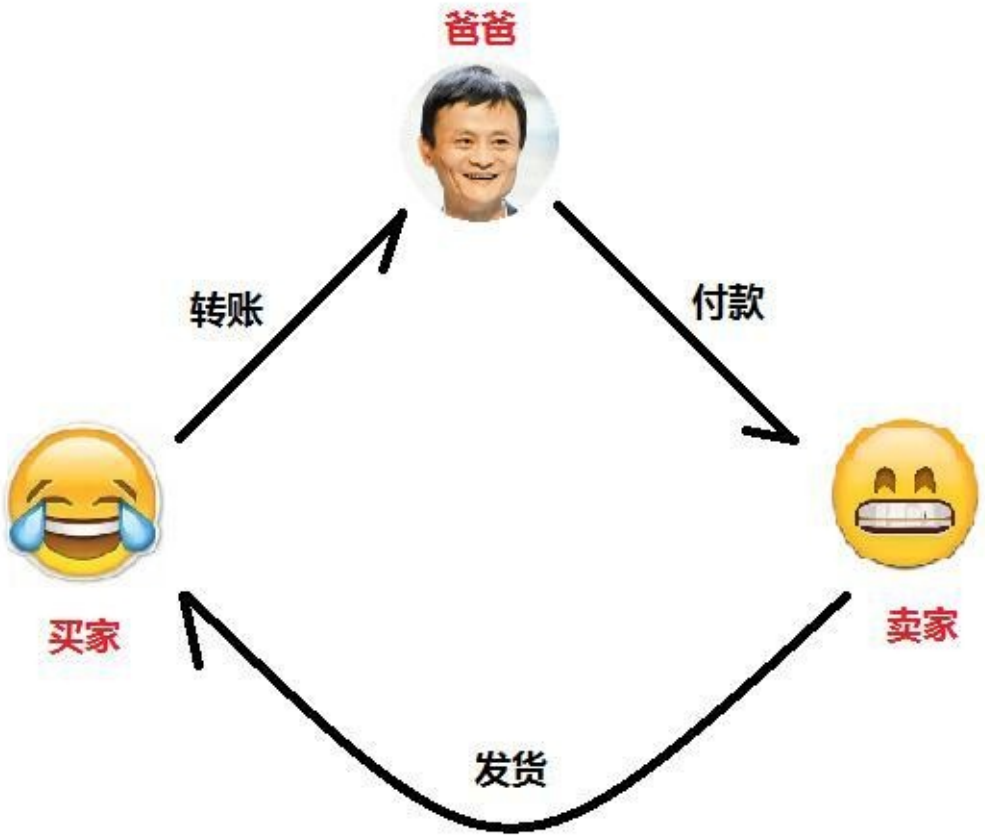


图1: 中心化集中式交易模式

在这个过程中，虽然你是在和卖家交易，但是这笔交易还牵扯到了除了你和卖家的第三方，即支付宝，你和卖家的交易都是围绕支付宝展开。因此，如果支付宝系统出了问题便会造成这笔交易的失败。并且虽然你只是简单的买了一个手机，但是你和卖家都要向第三方提供多余的信息。因此考虑极端情况，如果支付宝跑路了或者是拿了钱不却不承认你的交易或者是支付宝所在的城市因为开G20把所有人都赶走了(?)，那么你就悲剧了。

而去中心化的处理方式就要显得简单很多，你只需要和卖家交换钱和手机，然后双方都声称完成了这笔交易，就OK了。

可以看出在某些特定情况下，去中心化的处理方式会更便捷，同时也无须担心自己的与交易无关的信息泄漏。

其实如果只考虑两个人的交易并不能把去中心化的好处完全展示出来，设想如果有成千上万笔交易在进行，去中心化的处理方式会节约很多资源，使得整个交易自主化、简单化，并且排除了被中心化代理控制的风险。

去中心化是区块链技术的颠覆性特点，它无需中心化代理，实现了一种点对点的直接交互，使得高效率、大规模、无中心化代理的信息交互方式成为了现实。

当然，上述的例子有一个很大的潜在问题：没有了权威的中心化代理，怎样保证每笔交易的准确性和有效性呢？比如：如果没有了权威的中心化代理，张三某一天借了我100块钱，但是不还钱还不承认怎么办？这里就引出了区块链的其它特性。

两个基础难题

在去中心化以后，整个系统中没有了权威的中心化代理，信息的可信度和准确性便会面临问题。

问题1：类两军问题

第一次听说这个问题居然是在TCP的课上，大致说的是有两个相距很远的军队要传递信息，红军派遣一个信使去跟蓝军说："你他娘的把意大利炮拿出来！"。蓝军收到信息后又派了一个信使去红军说："收到指令！"。然后红军又派一个信使去蓝军说："知道你收到指令了！"。然后蓝军又派一个信使去红军说："知道你知道我收到指令了！"。然后红军又派一个信使去蓝军说："知道你知道我知道你收到指令了！".....然后就没完没了了。

<

img src="https://pic1.zhimg.com/50/3b659f83a2ffa35e9bc4f3e71042bf07_hd.jpg" data-rawwidth="596" data-rawheight="393" class="origin_image zh-lightbox-thumb" width="596" data-original="https://pic1.zhimg.com/3b659f83a2ffa35e9bc4f3e71042bf07_r.jpg"

>

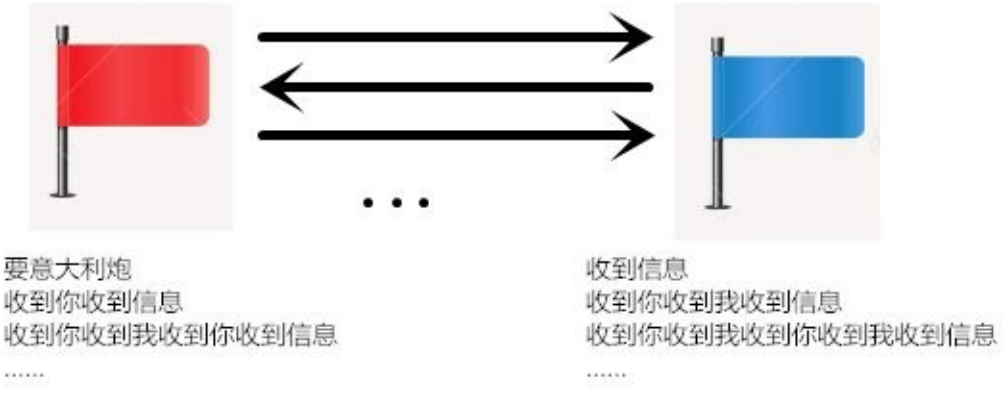


图2：在分布式计算中在异步系统和不可靠的通道上达到一致性是不可能的

在这种情况下，因为是点对点的通信，双方不可能在这种情况下达到信息的一致性。严谨一点，就是“在分布式计算上，试图在异步系统和不可靠的通道上达到一致性是不可能的”。

问题2：拜占庭将军问题

拜占庭罗马帝国在军事行动中，采取将军投票的策略来决定是进攻还是撤退，也就是说如果多数人决定进攻，就上去干。但是军队中如果有奸细（比如将军已经反水故意乱投票，或者传令官叛变擅自修改军令），那怎么保证最后投票的结果真正反映了忠诚的将军的意愿呢？

拜占庭将军问题反映到信息交换领域中来，可以理解为在一个去中心的系统中，有一些节点是坏掉的，它们可能向外界广播错误的信息或者不广播信息，在这种情况下如何验证数据传输的准确性。

区块链技术的诞生

现在让我们来一步一步在去中心化的系统中解决这些问题，见证区块链技术雏形的诞生。

1

我们先来建立一个去中心化的系统，为了方便理解，我们来看一个简单的去中心化借贷模型：如果A借了B 100块钱，这个时候，A在人群中大喊“我是A，我借给了B 100块钱！”，B也在人群中大喊“我是B，A借给了我100块钱！”，此时路人甲乙丙丁都听到了这些消息，因此所有人都在心中默默记下了“A借给了B100块钱”。你看，这个时候一个去中心化的系统就建立起来了，这个系统中不需要银行，也不需要借贷协议和收据，严格来说，甚至不需要人与人长久的信任关系（比如B突然又改口说“我不欠A钱！”，这个时候人民群众就会站出来说“不对，我的小本本上记录了你某天借了A100块钱！”）。

<

img src="https://pic1.zhimg.com/50/17966428c5ce9ea9d60aab703b3a405e_hd.jpg" data-rawwidth="777" data-rawheight="514" class="origin_image zh-lightbox-thumb" width="777" data-original="https://pic1.zhimg.com/17966428c5ce9ea9d60aab703b3a405e_r.jpg"

>

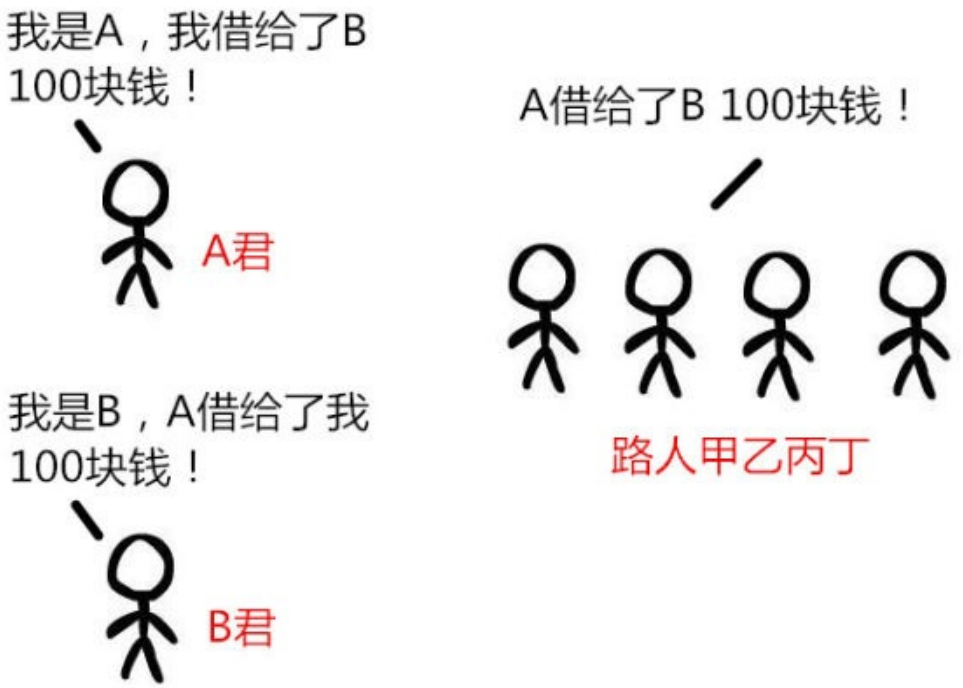


图3：去中心化借贷模型

2

可能你已经发现了，在上述的模型中，所谓的“100块钱”已经不重要了。换句话说，任何东西都可以在这个模型中交换，甚至你可以凭空杜撰一个东西，只要大家承认，你就可以让你杜撰的东西流通。比如：我在人群中高喊一声“我创造了10个查克拉！”，我甚至不需要知道查克拉是什么，也不需要关心世界上是不是真的有查克拉，只要大家都听到，然后在自己的小本本上记下“LaiW3n有10个查克拉”，于是我就真的有100个查克拉了。从此以后，我便可以声称我给了某人1个查克拉，只要路人甲乙丙丁都收到并且承认了这一信息，那我就算完成了这次交易，哪怕世界上没有查克拉。

你现在脑海中是不是浮现出了三个字——“比特币”？由于真正的区块链和比特币比我上述的模型复杂太多，细节也丰富太多，因此以下还是以查克拉举例，毕竟本文是Blockchain for Babies.（笑）

3

假设过了很长一段时间，我凭空创造的查克拉已经在这个系统中流通了起来，大家都开始认可了查克拉。但是这个系统中一共就只有10个查克拉，于是有人动了坏心思，他在人群中高呼“我有10个查克拉！”怎么办？大家是直接在本本上记下他有10个查克拉么，这样不是人人都可以伪造查克拉了么？

为了防止这种现象发生，我决定在我创造查克拉的时候给我的查克拉打上标记（更准确地说，我是给我喊的那句“我创造了10个查克拉”打上标记，比如标记为001），这样以后在每一笔交易的时候，我在高喊“我给了某某1个查克拉！”的时候，会附加上额外的一句话：“这1个查克拉的来源是记为001的那条记录，我的这句话标记为002！”。我们再抽象一点，某人喊话的内容的格式就变成了：“这句话编号xxx，上一句话的编号是yyy，我给了某某1个查克拉！”，这样就解决了伪造的问题。其实上述模型就变成一个简化的中本聪第一版比特币区块链协议：

<

img src="https://pic4.zhimg.com/50/129eca5e26cdaf3de77b9a8cffa296a8_hd.jpg" data-rawwidth="794" data-rawheight="533" class="origin_image zh-lightbox-thumb" width="794" data-original="https://pic4.zhimg.com/129eca5e26cdaf3de77b9a8cffa296a8_r.jpg"

>

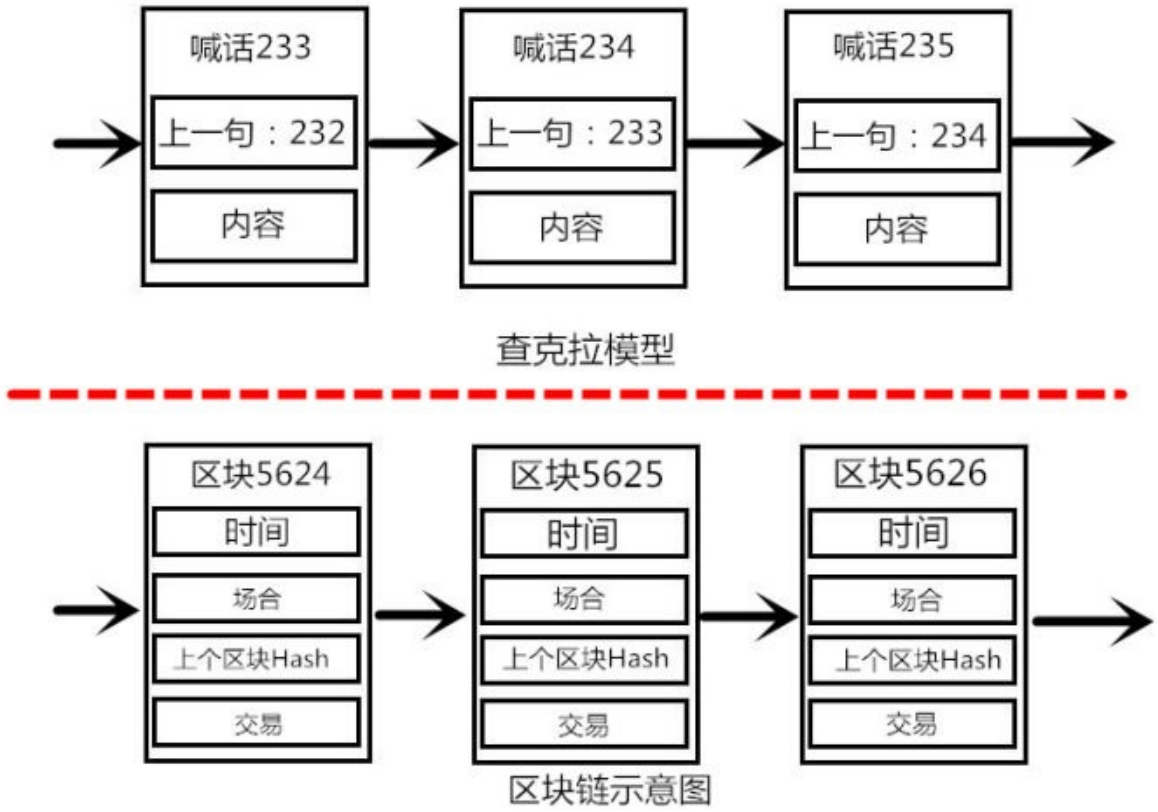


图4：查克拉模型和中本聪第一版区块链协议对比图

好了，看到这里你基本已经能够生动形象又不涉及任何细节地向你的弱智室友解释区块链了。但是也许你的室友是一个有打破沙锅问到底精神求是学子，因此你最好继续准好回答以下这几个问题。

1. “凭啥？”

你室友可能会问：“凭啥你喊一句话我就帮你记？我的小本本不要钱么？”。为了激励大家帮我传话和记账，我决定给第一个听到我喊话并且记录在小本本上的人一些奖励：第一个听到我喊话并记录下来的人，你就凭空得到了1个查克拉，这个查克拉是整个系统对你辛苦记账的报酬，而你记录了这句话之后，要马上告诉其它人你已经记录好了，让别人放弃继续记录这句话，并给你自己的记录编号让别人有据可查，然后你再把我的话加上你的记录编号一起喊出来，供下一个人记账。

当这个规则定下以后，这个系统中一定会出现一批人，他们开始竖着耳朵监听周围发出的声音，以抢占第一个记账的权利。对的，你脑海中是不是又浮现出了“比特币挖矿”的字眼？

值得一提的是，关于比特币挖矿，@玲珑邪僧举了一个很形象的例子：

> 单身汪们要找女票，国民岳母说我有好多女儿，这样吧我给你们出点题目，解出一个就给其中一个姑娘的微信号。
>
> 单身汪们疯狂竞争，想破脑袋去解题。只要其中一只汪解出一道题，就立马得意洋洋地昭告天下，示威全部单身汪，这个姑娘是我的啦，你们放弃吧。其他单身汪们即使不服也没有办法，惆怅懊恼也不是个事儿啊，还是麻溜地立马去解下一道题目吧。这只喜赢姑娘的幸运小汪被岳母认可后还能得到25个货币单位的彩礼，简直人生赢家。

1. “听谁的？”

在这个系统中，如果我和另一个人C几乎同时地喊出一句：“为了艾泽拉斯！”。由于听众所处的位置不同，一定会有人先听到我说的那句话，而另外一些人则先听到C的那句话，如果我们规定只能有一个人说出这句话，那到底这句话是谁说的？

如果不加任何条件，那么上述的情况一定会这样发展：一部分人认为这句话是我说的，在听到这句话之后开始记账，之后他们所做的所有事情都是基于这个事实，并且随着这个信息一次次的传下去，这条信息链会越来越深；而另外一群认为是C先说这句话的人，也会按照这样的趋势发展。这样，原本是一条唯一的信息链，在我们喊出“为了艾泽拉斯”这句话之后，分叉了！？

<

img src="https://pic2.zhimg.com/50/5e564510dae5b6cffa111c38ec271fea_hd.jpg" data-rawwidth="851" data-rawheight="488" class="origin_image zh-lightbox-thumb" width="851" data-original="https://pic2.zhimg.com/5e564510dae5b6cffa111c38ec271fea_r.jpg"

>

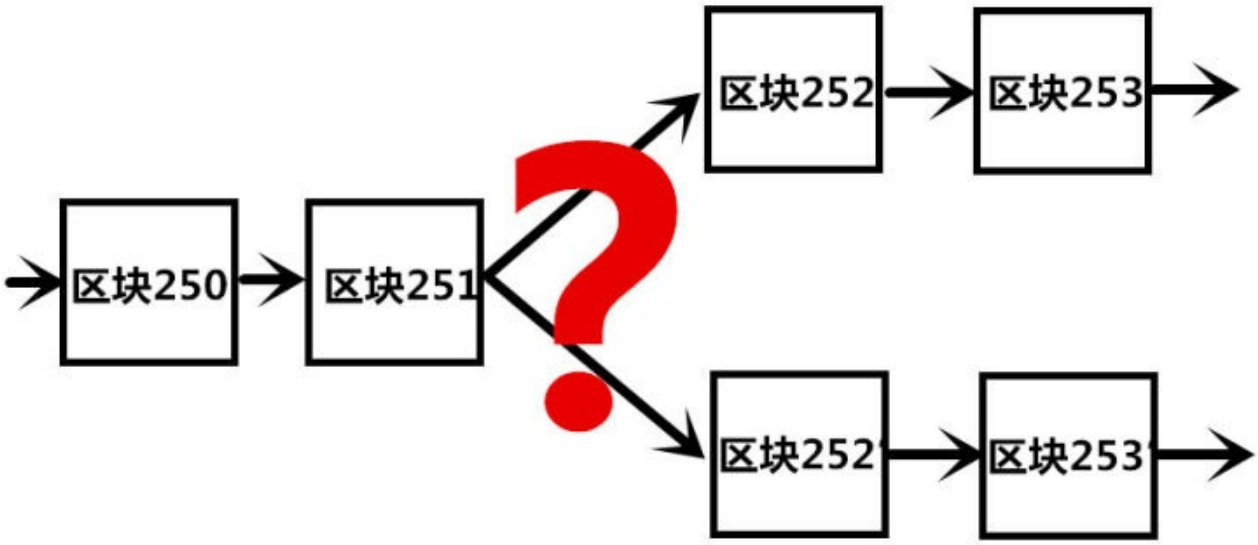


图5：“区块链”分叉

这会导致怎样的情况呢？按照我们的设想，应该每个人的小本本上记录的东西都是一样的，都是一条可以把所有信息串联起来的链条。但是在这一刻，他们小本本上记录的东西不一样了！这还玩毛啊？以后还怎么确定交易和信息的真实性！？

为了解决这个问题，我又追加了新的规则：每个人在记录小本本的时候，需要脱鞋然后用脚拿笔，在小本本上用正楷体书写！有了这个规定，由于用脚写字难度很大，每个人至少需要10分钟才能写完，而且由于每个人用脚写字的熟练度不通，写完这句话所用的时间也不同，因此一定会有人先写完然后高呼“我写完了！那句话是LaiW3n喊的！”，这样其它正在写这句话的人便会停笔，然后在小本本上重新开始写“那句话是来文写的，上一句的编号是xxx”。

如果你对上述我的解决方法感兴趣，你可以对照我上面的比喻去了解以下知识：

“听谁的”——中本聪破解“拜占庭将军问题”的算法

“在小本本上记录”——比特币挖矿

“脱鞋用脚写字”——比特币挖矿难度

“脱鞋写字速度”——算力

“新的规则”——工作量证明链

1. “双花”问题

这个时候你的室友可能又要问：如果我同时宣布我给了A一个查克拉和我给了B一个查克拉，但是我只有一个查克拉，那咋整？是A和B都收到了查克拉还是咋地？

这个时候你只需要托起他的下巴，温柔地看着他的眼睛，用手刮刮他的鼻子，说：“小妖精，你把这种情况带到上面的规则中去试试？”

为了更好的理解比特币和区块链技术，最近整理了部分比特币与区块链核心技术笔记。

一：一些需要知道的术语。

1. 比特币：比特币（Bitcoin，缩写BTC）是一种总量恒定2100万的数字货币，和互联网一样具有去中心化、全球化、匿名性等特性。向地球另一端转账比特币，就像发送电子邮件一样简单，低成本，无任何限制。比特币因此被用于跨境贸易、支付、汇款等领域。
2. 比特币地址：比特币地址可以理解为你的账户身份id,或者可以比喻为银行卡号（例如：12bZrnSvmEusm3KmshobPJiNcZEkJcXGNr）由一串字符和数字组成，以阿拉伯数字“1”开头，不过这数字由密码算法产生。
3. 区块：一个区块就是若干交易数据的集合，它会被标记上时间戳和之前一个区块的独特标记。区块头经过哈希运算后会生成一份工作量证明，从而验证区块中的交易。有效的区块经过全网络的共识后会被追加到主区块链中。
4. 区块链是一串通过验证的区块，当中的每一个区块都与上一个相连，一直连到创世区块。
5. 交易确认：当一项交易被区块收录时，我们可以说它有一次确认。矿工们在此区块之后每再产生一个区块，此项交易的确认数就再加一。当确认数达到六及以上时，通常认为这笔交易比较安全并难以逆转。
6. 交易：简单地说，交易指把比特币从一个地址转到另一个地址。更准确地说，一笔“交易”指一个经过签名运算的，表达价值转移的数据结构。每一笔“交易”都经过比特币网络传输，由矿工节点收集并封包至区块中，永久保存在区块链某处。
7. 难度：整个网络会通过调整“难度”这个变量来控制生成工作量证明所需要的计算力。
8. 难度目标：使整个网络的计算力大致每10分钟产生一个区块所需要的难度数值即为难度目标。
9. 难度调整：整个网络每产生2,106个区块后会根据之前2,106个区块的算力进行难度调整。
10. 矿工费：交易的发起者通常会向网络缴纳一笔矿工费，用以处理这笔交易。大多数的交易需要0.5毫比特币的矿工费。
11. 矿工：指通过不断重复哈希运算来产生工作量证明的各网络节点。
12. 钱包：指保存比特币地址和私钥的软件，可以用它来接受、发送、储存你的比特币。、
13. 比特币网络：是一个由若干节点组成的用以广播交易信息和数据区块的P2P网络。

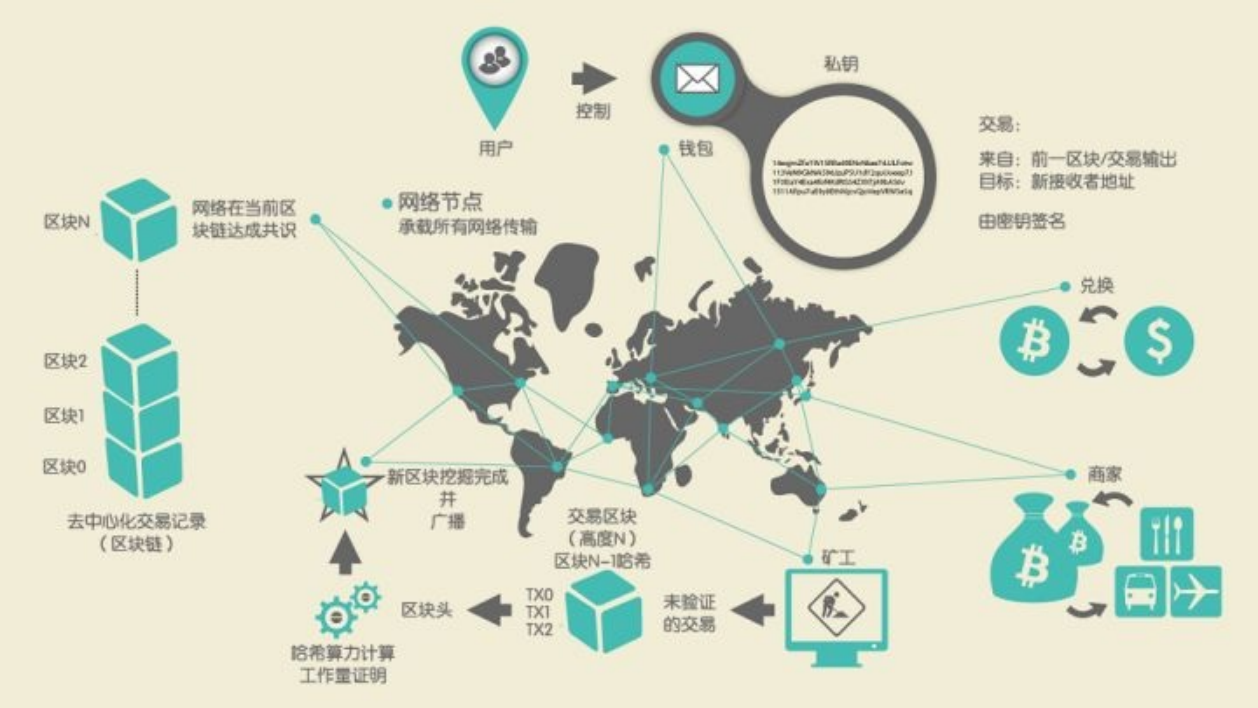
二、基本的介绍

什么是比特币

1. 不同于传统货币，比特币是完全虚拟的。它不但没有实体，本质上也没有一种虚拟物品代表比特币。
2. 比特币隐含在收发币的转账记录中。用户只要有证明其控制权的密钥，用密钥解锁，就可以发送比特币。
3. 这些密钥通常存储在计算机的数字钱包里。拥有密钥是使用比特币的唯一条件，这让控制权完全掌握在每个人手中。
4. 本质上，挖矿把央行的货币发行和结算功能进行分布式，用全球化的算力竞争来取代对

中央发行机构的需求。

5. 比特币系统包含调节挖矿难度的协议。挖矿——在比特币网络中成功写入一个区块交易——的难度是动态调整的，保证不管有多少矿工（多少CPU）挖矿，平均每10分钟只有一个矿工成功。
6. 比特币协议还规定，每四年新币的开采量减半，同时限制比特币的最终开采总量为2,100万枚。这样，流通中的比特币数量非常接近一条曲线，并将在2140年比特币将达到2,100万枚。由于比特币的开采速度随时间递减，从长期来看，比特币是一种通货紧缩货币。此外，不能通过“印刷”新比特币来实现“通货膨胀”。
7. 比特币由这些构成：
8. 一个去中心化的点对点网络（比特币协议）
9. 一个公共的交易账簿（区块链）
10. 一个去中心化的数学的和确定性的货币发行（分布式挖矿）
11. 一个去中心化的交易验证系统（交易脚本）



比特币网络

比特币钱包-客户端

完整客户端：

- 一个完整客户端，或称“全节点”，是存储所有比特币交易的整个交易历史（由每一个用户完成的每一笔交易，曾经所有的每一笔）的客户端，管理用户的钱包，并可以在比特币网络上直接开始交易。
- 类似于一个独立的电子邮件服务器，因为它处理着协议的各个方面，而不依赖于任何其它的服务器或第三方服务。

- 完整客户端目前需要大概145g端空间来存储全部区块数据。

来源：https://bitcoin.org/en/download

轻量级客户端

- 一个轻量级客户端存储用户的钱包，但需要依赖第三方服务器才能进行比特币交易，才能接入比特币网络。
- 轻量级客户端不保存所有交易的完整副本，因此必须信赖第三方的服务器来获取交易确认。
- 这就类似于一个独立的电子邮件客户端，能够通过邮箱服务器来访问一个邮箱，因为它在网络交流中依赖于一个第三方服务器。

在线客户端

- 在线客户端通过网页浏览器在第三方服务器上访问和储存该用户的钱包。
- 这类似于在线邮件，因为它完全依赖于第三方服务器。

移动客户端

- 智能手机的移动客户端，例如基于Android系统，既可以作完整客户端运行，也可作为轻量级客户端或在线客户端。
- 一些移动客户端是与在线客户端或桌面客户端同步的，提供跨多个设备但有一个共同的资金源的多平台钱包。

- 比特币客户端的选择，取决于用户想要管理资金的数目。一个完整的客户端将为用户提供最高级的管理和独立性。这样钱包的备份和安全责任就转移到了用户身上。另一种选择是在线客户端，其设置和使用是最简单的，但在线客户端的取舍还在于需衡量第三方介入的风险，因为安全性和控制权是由用户和网页服务商所共同承担的。如果一个在线钱包服务遭受了损失，就像已发生过的那样，用户们可能会失去所有的资金。反过来看，如果用户的一个完整客户端没有进行适当的备份，他们可能会因为电脑的操作失误而丢失他们的资金。

三、比特币基本原理

概述

1. 从千分之一比特币(1毫比特币）到一亿分之一比特币（1聪比特币），比特币网络可以处理任意小额交易。在本书中，我们将用“比特币”这个术语来表示任意数量的比特币货币，从最小单元（1聪）到可被挖出的所有比特币总数（21,000,000）。
2. 交易包含了每一笔被转移的比特币（输入）的所有权证明，它以所有者的数字签名形式存在，并可以被任何人独立验证。
3. 在比特币术语中，“消费”指的是签署一笔交易：转移一笔以前交易的比特币给以比特币地址所标识的新所有者。

比特币交易

1. 交易告知全网：比特币的持有者已授权把比特币转帐给其他人。而新持有者能够再次授权，转移给该比特币所有权链中的其他人，产生另一笔交易来花掉这些比特币，后面的持有者在花费比特币也是用类似的方式。
2. 交易形式

3. 最常见的交易形式是从一个地址到另一个地址的简单支付，这种交易也常常包含给支付者的“找零”

4. 另一种常见的交易形式是集合多个输入到一个输出的模式。这相当于现实生活中将很多硬币和纸币零钱兑换为一个大额面钞。像这样的交易有时由钱包应用产生来清理许多在支付过程收到的小数额的找零。
5. 最后一种在比特币账簿中常见的交易形式是将一个输入分配给多个输出，即多个接收者（如图2-7）的交易。这类交易有时被商业实体用作分配资金，例如给多个雇员发工资的情形。

6. 交易的构建。比特币交易建立和签名时不用连接比特币网络。只有在执行交易时才需要将交易发送到网络。

7. 完整客户端占太大的硬盘空间，所以大多数钱包使用轻量级的客户端，只保存用户自己的未消费输出。如果钱包客户端没有某一未消费交易输出，它可以通过不同的服务者提供的各种API或完整索引节点的JSON PRC API从比特币网络中拿到这一交易信息。

```
curl 'https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK'

返回：
{
    "unspent_outputs":[

        {
            "tx_hash":"f2c245c38672a5d8fba5a5caa44dcef277a52e916a0603272f91286f2b052706",
            "tx_hash_big_endian":"0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2",
            "tx_index":47854970,
            "tx_output_n": 1,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 8450000,
```

```
            "value_hex": "0080efd0",
            "confirmations":217419
        },

        {
            "tx_hash":"0365fdc169b964ea5ad3219e12747e9478418fdc8abed2f5fe6d0205c96def2
9",
            "tx_hash_big_endian":"29ef6dc905026dfef5d2be8adc8f4178947e74129e21d35aea64
b969c1fd6503",
            "tx_index":71083209,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 100000,
            "value_hex": "0186a0",
            "confirmations":161802
        },

        {
            "tx_hash":"d9717f774daab8d3dd470853204394c82e3c01097479575d6d2ee97d7b3bdfa
1",
            "tx_hash_big_endian":"a1df3b7b7de92e6d5d57797409013c2ec8944320530847ddd3b8
aa4d777f71d9",
            "tx_index":75974855,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 1000000,
            "value_hex": "0f4240",
            "confirmations":153961
        },

        {
            "tx_hash":"3f1df69df90d097981ca9c97ad8b6a32daed345565a433f8c8e472b2dab2ac7
9",
            "tx_hash_big_endian":"79acb2dab272e4c8f833a4655534edda326a8bad979cca817909
0df99df61d3f",
            "tx_index":79887883,
            "tx_output_n": 1,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 719787,
            "value_hex": "0afbab",
            "confirmations":148074
        },

        {
            "tx_hash":"417bdb6f5db3e830407f94d1a82d1667e738b19da3679b7263ebfb913394efd
d",
            "tx_hash_big_endian":"ddef943391fbeb63729b67a39db138e767162da8d1947f4030e8
b35d6fdb7b41",
            "tx_index":170905487,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
```

```
            "confirmations":67883
        },

        {
            "tx_hash":"d049d6039f9d1cb2625bac294d7465b4b1077bd5bc0e30e01e02b184db524c1
f",
            "tx_hash_big_endian":"1f4c52db84b1021ee0300ebcd57b07b1b465744d29ac5b62b21c
9d9f03d649d0",
            "tx_index":174630347,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 11100,
            "value_hex": "2b5c",
            "confirmations":65345
        },

        {
            "tx_hash":"b8a6470c7a38d0983effed00a3f75c74ba371da1387352f35d1df155851ea8d
1",
            "tx_hash_big_endian":"d1a81e8555f11d5df3527338a11d37ba745cf7a300edff3e98d0
387a0c47a6b8",
            "tx_index":175949432,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64439
        },

        {
            "tx_hash":"a2b9570e26e3991fc999c42dc8c6eea7b06514b61814da1a71b56c6ba2ae651
c",
            "tx_hash_big_endian":"1c65aea26b6cb5711ada1418b61465b0a7eec6c82dc499c91f99
e3260e57b9a2",
            "tx_index":175955161,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64430
        },

        {
            "tx_hash":"05230cb8cd8c6a3788ed41433dfdd68a1a608cc8feb3bc1c29d97ce84bec070
e",
            "tx_hash_big_endian":"0e07ec4be87cd9291cbcb3fec88c601a8ad6fd3d4341ed88376a
8ccdb80c2305",
            "tx_index":175955664,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64430
```

```
        }

    ]
}curl 'https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK'

返回：
{
    "unspent_outputscurl 'https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQ
cwXYCpvK8h7FK'

返回：
{
    "unspent_outputs":[

        {
            "tx_hash":"f2c245c38672a5d8fba5a5caa44dcef277a52e916a0603272f91286f2b05270
6",
            "tx_hash_big_endian":"0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a5
7286c345c2f2",
            "tx_index":47854970,
            "tx_output_n": 1,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 8450000,
            "value_hex": "0080efd0",
            "confirmations":217419
        },

        {
            "tx_hash":"0365fdc169b964ea5ad3219e12747e9478418fdc8abed2f5fe6d0205c96def2
9",
            "tx_hash_big_endian":"29ef6dc905026dfef5d2be8adc8f4178947e74129e21d35aea64
b969c1fd6503",
            "tx_index":71083209,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 100000,
            "value_hex": "0186a0",
            "confirmations":161802
        },

        {
            "tx_hash":"d9717f774daab8d3dd470853204394c82e3c01097479575d6d2ee97d7b3bdfa
1",
            "tx_hash_big_endian":"a1df3b7b7de92e6d5d57797409013c2ec8944320530847ddd3b8
aa4d777f71d9",
            "tx_index":75974855,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 1000000,
            "value_hex": "0f4240",
            "confirmations":153961
        },
```

```
        {
            "tx_hash":"3f1df69df90d097981ca9c97ad8b6a32daed345565a433f8c8e472b2dab2ac7
9",
            "tx_hash_big_endian":"79acb2dab272e4c8f833a4655534edda326a8bad979cca817909
0df99df61d3f",
            "tx_index":79887883,
            "tx_output_n": 1,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 719787,
            "value_hex": "0afbab",
            "confirmations":148074
        },

        {
            "tx_hash":"417bdb6f5db3e830407f94d1a82d1667e738b19da3679b7263ebfb913394efd
d",
            "tx_hash_big_endian":"ddef943391fbeb63729b67a39db138e767162da8d1947f4030e8
b35d6fdb7b41",
            "tx_index":170905487,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":67883
        },

        {
            "tx_hash":"d049d6039f9d1cb2625bac294d7465b4b1077bd5bc0e30e01e02b184db524c1
f",
            "tx_hash_big_endian":"1f4c52db84b1021ee0300ebcd57b07b1b465744d29ac5b62b21c
9d9f03d649d0",
            "tx_index":174630347,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 11100,
            "value_hex": "2b5c",
            "confirmations":65345
        },

        {
            "tx_hash":"b8a6470c7a38d0983effed00a3f75c74ba371da1387352f35d1df155851ea8d
1",
            "tx_hash_big_endian":"d1a81e8555f11d5df3527338a11d37ba745cf7a300edff3e98d0
387a0c47a6b8",
            "tx_index":175949432,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64439
        },

        {
```

```
            "tx_hash":"a2b9570e26e3991fc999c42dc8c6eea7b06514b61814da1a71b56c6ba2ae651
c",
            "tx_hash_big_endian":"1c65aea26b6cb5711ada1418b61465b0a7eec6c82dc499c91f99
e3260e57b9a2",
            "tx_index":175955161,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64430
        },

        {
            "tx_hash":"05230cb8cd8c6a3788ed41433dfdd68a1a608cc8feb3bc1c29d97ce84bec070
e",
            "tx_hash_big_endian":"0e07ec4be87cd9291cbcb3fec88c601a8ad6fd3d4341ed88376a
8ccdb80c2305",
            "tx_index":175955664,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000,
            "value_hex": "2710",
            "confirmations":64430
        }

    ]
}
```


创建交易输出

1. 交易的输出会被创建成为一个包含这笔数额的脚本的形式，只能被引入这个脚本的一个解答后才能兑换。
2. 简单点说就是，支付方的交易输出会包含一个脚本，这个脚本说“这个输出谁能拿出一个签名和接收方的公开地址匹配上，就支付给谁”。因为只有接收方钱包的私钥可以匹配这个地址，所以只有接收方的钱包可以提供这个签名以兑换这笔输出。因此支付方式会用需要接收方的签名来包装一个输出。
3. 假如支付方的地址上，金额是0.10比特币的输出形式，而此次交易只需要支付0.015比特币，就需要找0.085比特币的零钱。支付方的钱包将自己的金额分成了两个支付：一个给接收方，一个给自己。她可以在以后的交易里消费这笔零钱输出。
4. 最后，为了让这笔交易尽快地被网络处理，支付方的钱包会多付一小笔费用。这个不是明显地包含在交易中的，而是通过输入和输出的差值所隐含的。这个差值会就被矿工当作交易费放到区块的交易里，最终放进区块链帐薄中。
5. 支付方的钱包应用创建的交易大小为258字节，包含了金额未来所属需要的全部信息。最后的最后，这个交易必须要被传送到比特币网络中以成为分布式账簿（区块链）的一部分。

交易的传送

1. 比特币网络是由参与的比特币客户端联接几个其他比特币客户端组成的P2P网络。比特币网络的目的是将交易和区块传播给所有参与者。
2. 钱包应用可以发送新的交易给其它任意一个已联接到互联网的比特币客户端。支付方的钱包不必直接连着接收方的比特币钱包。任何比特币网络节点（其它客户端）收到一个之前没见过的有效交易时会立刻将它转发给联接到自身的其它节点。因此，这个交易迅速地从P2P网络中传播开来，几秒内就能到达大多数节点。
3. 接受者的钱包接收到支付方发过来的交易节点时，会立即确认交易是一个收入支付，因为它包含能用自己私钥兑换的输出。接收方的钱包应用也能够独立地用之前未消费输入来确认这个交易是正确构建的，并且由于包含足够交易费会被下一个区块包含进去。这时接收方就可以以一个很小的风险假定这个交易会很快被加到区块且被确认。
4. 一个对比特币交易的常见误解是它们必须要等10分钟后被确认加进一个新区块，或等60分钟以得到六次确认后才是有效的。虽然这些确认可以确保交易已被整个网络接受，但对于像一杯咖啡这样的小额商品来说就没有必要等待那么长时间了。一个商家可以免确认来接受比特币小额支付。这样做的风险不比接受一个不是用有效身份证领取或没有签名的信用卡的风险更大，而后者是现在商家常做的事情。

交易确认--挖矿

1. 挖矿在比特币系统中起着两个作用：
2. 挖矿在构建区块时会创造新的比特币，和一个中央银行印发新的纸币很类似。每个区块创造的比特币数量是固定的，随时间会渐渐减少
3. 挖矿创建信任。挖矿确保只有在包含交易的区块上贡献了足够的计算量后，这些交易才被确认。区块越多，花费的计算量越大，意味着更多的信任
4. 工作量证明算法指的用SHA256加密算法不断地对区块头和一个随机数字进行哈希计算，直到出现一个和预设值相匹配的解
5. 网络中产生的一笔交易直到成为整个比特币大账簿——区块链的一部分时才会被确认有效。
6. 平均每10分钟，矿工会将自上一个区块以来发生的所有交易生成一个新的区块。
7. 新交易不断地从用户钱包和应用流入比特币网络。当比特币网络上的节点看到这些交易时，会先将它们放到各自节点维护的一个临时的未经验证的交易池中。当矿工构建一个新区块时，会将这些交易从这个交易池中拿出来放到这个新区块中，然后通过尝试解决一个非常困难的问题（也叫工作量证明）以证明这个新区块的合法性。
8. 交易被加进新区块时，以交易费用高的优先以及其它的一些规则进行排序。
9. 矿工一旦从网络上收到一个新区块时，会意识到在这个区块上的解题竞赛已经输掉了，会马上开始下一个新区块的挖掘工作。它会立刻将一些交易和这个新区块的数字指纹放在一起开始构建下一个新区块，并开始给它计算工作量证明。
10. 每个矿工会在他的区块中包含一个特殊的交易，将新生成的比特币（当前每区块为25比特币）作为报酬支付到他自己的比特币地址。如果他找到了使得新区块有效的解法，他就会得到这笔报酬，因为这个新区块被加入到了总区块链中，他添加的这笔报酬交易也

会变成可消费的。

11. 按惯例来说，一个区块获得六次以上“证明”时就被认为是不可撤销的了，因为要撤销和重建六个区块需要巨量的计算。

四、比特币客户端

1. BitCoin，比特币核心拥有交易账簿（区块链）的一份完整拷贝，里面记录了自2009年比特币网络被发明以来发生在比特币网络上的每一笔交易。
2. bitcoin源码地址： https://github.com/bitcoin/bitcoin
3. 带有rc后缀的是预发行版本，可以用来测试。没有后缀的稳定版本可以直接在产品环境上运行。

其他客户端

1. pycoin 是一款基于Python库，并可以支持比特币密钥的操作和交易的客户端，甚至可以支持编译语言从而处理非标准交易。
2. btcd是一款基于Go语言的全节点比特币工具。目前，它通过使用精准的规则（包括bugs），下载、验证和服务区块链。它同时依靠新发掘出来的区块来维持交易池，同时依赖没有形成区块的单独交易。在缜密的规则以及检查下，确保了每笔独立交易的安全，并且可以过滤基于矿工需求的交易。btcd与bitcoind的一个主要区别是btcd不包含比特币钱包的功能，其实这是一个精心的设计。这意味着你不能直接通过btcd进行比特币交易。然而这项功能可以由正在研发的btcwallet与btcgui两个项目提供。另一个显著的区别是btcd同时支持HTTP POST（比如bitcoind）与推荐使用的Websockets两种通信协议的请求。并且btcd的RPC连接默认设置为TLS-开启。
3. bitcoinj 一款全节点java客户端和程序库。

五、算法与秘钥、地址、钱包

在比特币交易的支付环节，收件人的公钥是通过其数字指纹表示的，称为比特币地址，就像支票上的支付对象的名字（即“收款方”）。一般情况下，比特币地址由一个公钥生成并对应于这个公钥。然而，并非所有比特币地址都是公钥；他们也可以代表其他支付对象，譬如脚本。这样一来，比特币地址把收款方抽象起来了。

椭圆曲线算法：私钥和公钥

比特币或者大部分区块链的核心系统是基于椭圆曲线算法（Elliptic-curve cryptography

）建立起来的，ECC 是一种公开密钥密码学，又称为非对称密码学。在这种密码学中，需要产生一对密钥。其中一个密钥称为私钥，需要保密；另一个密钥称为公钥，是可以公开让别人知道的。私钥和公钥在数学上的关系是不可逆的，也就是通过某个数学函数，我们可以从私钥计算出公钥，但是不能从公钥反向推导出私钥(或者说从计算上是不可行的)。有关此算法的相关可以参考：http://diamond.boisestate.edu/~liljanab/MATH308/GuideToECC.pdf，简单来说就是

1. 一个比特币钱包中包含一系列的密钥对，每个密钥对包括一个私钥和一个公钥。私钥

（k）是一个数字，通常是随机选出的。有了私钥，我们就可以使用 < font color="red" > 椭圆曲线乘法 < /font > 这个单向加密函数产生一个公钥（K）。

2. 有了公钥（K），我们就可以使用一个单向加密哈希函数生成比特币地址（A）
3. 在比特币交易中，私钥用于生成支付比特币所必需的签名以证明资金的所有权。
4. 私钥必须始终保持机密，因为一旦被泄露给第三方，相当于该私钥保护之下的比特币也拱手相让了。
5. 私钥还必须进行备份，以防意外丢失，因为私钥一旦丢失就难以复原，其所保护的比特币也将永远丢失。
6. 私钥一般是256 bit
7. 比特币软件使用操作系统底层的随机数生成器来产生256位的熵（随机性）。通常情况下，操作系统随机数生成器由人工的随机源进行初始化，也可能需要通过几秒钟内不停晃动鼠标等方式进行初始化。
8. 私钥可以是1和n-1之间的任何数字，其中n是一个常数（n=1.158 * 1077，略小于2256），并由比特币所使用的椭圆曲线的阶所定义（见4.1.5 椭圆曲线密码学解释）。要生成这样的一个私钥，我们随机选择一个256位的数字，并检查它是否小于n-1。从编程的角度来看，一般是通过在一个密码学安全的随机源中取出一长串随机字节，对其使用SHA256哈希算法进行运算，这样就可以方便地产生一个256位的数字。如果运算结果小于n-1，我们就有了一个合适的私钥。否则，我们就用另一个随机数再重复一次。
9. 一定不要使用自己写的代码或使用编程语言内建的简易随机数生成器来获得一个随机数。我们建议读者使用密码学安全的伪随机数生成器（CSPRNG），并且需要有一个来自具有足够熵值的源的种子。使用随机数发生器的程序库时，需仔细研读其文档，以确保它是加密安全的。对CSPRNG的正确实现是密钥安全性的关键所在。
10. dumpprivkey命令会把私钥以Base58校验和编码格式显示，这种私钥格式被称为钱包导入格式（WIF，Wallet Import Format）
11. sx newkey - > 生成并显示私钥
12. 比特币地址A = RIPEMD160(SHA256(public key))
13. 通常用户见到的比特币地址是经过“Base58Check”编码的，这种编码使用了58个字符（一种Base58数字系统）和校验码，提高了可读性、避免歧义并有效防止了在地址转录和输入中产生的错误。Base58Check编码也被用于比特币的其它地方，例如比特币地址、私钥、加密的密钥和脚本哈希中，用来提高可读性和录入的正确性。
14. 最全面的比特币Python库是 Vitalik Buterin写的pybitcointools

比特币钱包

1. 钱包是私钥的容器，通常通过有序文件或者简单的数据库实现。另外一种制作私钥的途径是 确定性密钥生成。在这里你可以用原先的私钥，通过单向哈希函数来生成每一个新的私钥，并将新生成的密钥按顺序连接。只要你可以重新创建这个序列，你只需要第一个私钥（称作种子、主私钥）来生成它们。

非确定性(随机)钱包

1. 在最早的一批比特币客户端中，钱包只是随机生成的私钥集合。这种类型的钱包被称作

零型非确定钱包。

2. 举个例子，比特币核心客户端预先生成100个随机私钥，从最开始就生成足够多的私钥并且每把钥匙只使用一次。这种类型的钱包有一个昵称“Just a Bunch Of Keys（一堆私钥）”简称JBOK。
3. 这种钱包现在正在被确定性钱包替换，因为它们难以管理、备份以及导入。
4. 随机钥匙的缺点就是如果你生成很多，你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。每一把钥匙都必须备份，否则一旦钱包不可访问时，钱包所控制的资金就付之东流。
5. 这种情况直接与避免地址重复使用的原则相冲突——每个比特币地址只能用一次交易。地址通过关联多重交易和对方的地址重复使用会减少隐私。

确定性（种子）钱包

1. 确定性，或者“种子”钱包包含通过使用单项离散方程而可从公共的种子生成的私钥。
2. 种子是随机生成的数字。这个数字也含有比如索引号码或者可生成私钥的“链码”
3. 在确定性钱包中，种子足够收回所有的已经产生的私钥，所以只用在初始创建时的一个简单备份就足以搞定。
4. 种子也足够让钱包输入或者输出。这就很容易允许使用者的私钥在钱包之间轻松转移输入。

助记码词汇

1. 助记码词汇是英文单词序列代表（编码）用作种子对应所确定性钱包的随机数。单词的序列足以重新创建种子，并且从种子那里重新创造钱包以及所有私钥。
2. 助记码代码可以让使用者复制钱包更容易一些，因为它们相比较随机数字顺序来说，可以很容易地被读出来并且正确抄写
3. BIP0039定义助记码和种子的创建过程如下：
4. 1.创造一个128到256位的随机顺序（熵）
5. 2.提出SHA256哈希前几位，就可以创造一个随机序列的校验和
6. 3.把校验和加在随机顺序的后面
7. 4.把顺序分解成11位的不同集合，并用这些集合去和一个预先已经定义的2048个单词字典做对应
8. 5.生成一个12至24个词的助记码

分层确定性钱包

1. 分层确定性钱包包含从数结构所生成的钥匙。这种母钥匙可以生成子钥匙的序列。这些子钥匙又可以衍生出孙钥匙，以此无穷类推。

高级密钥和地址

加密私钥（**BIP0038**）

1. 私钥必须保密
2. BIP0038提出了一个通用标准，使用一个口令加密私钥并使用Base58Check对加密的私钥进行编码，这样加密的私钥就可以安全地保存在备份介质里，安全地在钱包间传输，保持密钥在任何可能被暴露情况下的安全性。
3. BIP0038加密方案是：
4. 输入一个比特币私钥，通常使用WIF编码过，base58chek字符串的前缀“5”
5. 输入一个长密码作为口令，通常由多个单词或一段复杂的数字字母字符串组成
6. BIP0038加密方案的输出一个由base58check编码过的加密私钥，前缀为6P
7. 如果你看到一个6P开头的的密钥，这就意味着该密钥是被加密过，并需要一个口令来转换（解码）该密钥回到可被用在任何钱包WIF格式的私钥（前缀为5）。
8. 许多钱包APP现在能够识别BIP0038加密过的私钥，会要求用户提供口令解码并导入密钥。

P2SH (Pay-to-Script Hash)和多重签名地址

1. 传统的比特币地址从数字1开头，来源于公钥，而公钥来源于私钥。虽然任何人都可以将比特币发送到一个1开头的地址，但比特币只能在通过相应的私钥签名和公钥哈希值后才能消费。
2. 以数字3开头的比特币地址是P2SH地址，有时被错误的称谓多重签名或多重签名地址。他们指定比特币交易中受益人作为哈希的脚本，而不是公钥的所有者。
3. 不同于P2PKH交易发送资金到传统1开头的比特币地址，资金被发送到3开头的地址时，需要的不仅仅是一个公钥的哈希值，同时也需要一个私钥签名作为所有者证明。在创建地址的时候，这些要求会被定义在脚本中，所有对地址的输入都会被这些要求阻隔。
4. script hash = RIPEMD160(SHA256(script)) 脚本哈希的结果是由Base58Check编码前缀为5的版本、编码后得到开头为3的编码地址。
5. P2SH函数最常见的实现是用于多重签名地址脚本。顾名思义，底层脚本需要多个签名来证明所有权，此后才能消费资金。设计比特币多重签名特性是需要从总共N个密钥中需要M个签名（也被称为“阈值”），被称为M-N多签名，其中M是等于或小于N。

纸钱包

1. 纸钱包是打印在纸张上的比特币私钥。有时纸钱包为了方便起见也包括对应的比特币地址，但这并不是必要的，因为地址可以从私钥中导出。
2. 纸钱包是一个非常有效的建立备份或者线下存储比特币（即冷钱包）的方式。作为备份机制，一个纸钱包可以提供安全性，以防在电脑硬盘损坏、失窃或意外删除的情况下造成密钥的的丢失。
3. 作为一个冷存储的机制，如果纸钱包密钥在线下生成并永久不在电脑系统中存储，他们在应对黑客攻击，键盘记录器，或其他在线电脑欺骗更有安全性。
4. 一个更复杂的纸钱包存储系统使用BIP0038加密的私钥。打印在纸钱包上的这些私钥被其所有者记住的一个口令保护起来。没有口令，这些被加密过的密钥也是毫无用处的。

5. 虽然你可以多次存款到纸钱包中，但是你最好一次性提款，一次性提取里面所有的资金。因为如果你提取的金额少于其中的金额的话，会生成一个找零地址。并且，你所用的电脑可能被病毒感染，那么就有可能泄露私钥。一次性提款可以减少私钥泄露的风险，如果你所需的金额比较少，那么请把余额找零到另一个纸钱包中。

六、交易

比特币交易的生命周期

1. 一笔比特币交易的生命周期起始于它被创建的那一刻，也就是诞生
2. 比特币交易会被一个或者多个签名加密，这些签名标志着对该交易指向的比特币资金的使用许可。
3. 接下来，比特币交易被广播到比特币网络中。在比特币网络中，每一个节点（比特币交易参与者）验证、并将交易在网络中进行广播，直到这笔交易被网络中大多数节点接收。
4. 最终，比特币交易被一个挖矿节点验证，并被添加到区块链上一个记录着许多比特币交易的区块中。
5. 比特币交易可以被任何人在线上或线下创建，即便创建这笔交易的人不是这个账户的授权签字人。类似于，一个负责应付账款的柜员可以创建比特币交易，然后让CEO对它进行数字签名，从而使之有效。
6. 一张支票是指定一个特定账户作为资金来源的，但是比特币交易指定以往的一笔交易作为其资金来源，而不是一个特定账户。
7. 一旦一笔比特币交易被创建，它会被资金所有者（们）签名。如果它是合法创建并签名的，则该笔交易现在就是有效的，它包含了转移这笔资金所需要的所有信息。
8. 最终，有效的比特币交易必须能接入比特币网络，从而使之能被传送，直至抵达下一个登记在公共总账薄（区块链）的挖矿节点。

广播交易至比特币网络

1. 首先，一笔交易需要传递至比特币网络，才能被传播，也才能加入区块链中。
2. 本质上，一笔比特币交易只是300到400字节的数据，而且它们必须被发送到成千上万个比特币节点中的任意一个。只要发送者能使用多于一个比特币节点来确保这笔交易被传播，那么发送者并不需要信任用来传播该笔交易的单一节点。
3. 比特币交易因此可以通过未加密网络（例如WiFi、蓝牙、NFC、ChirP、条形码或者复制粘贴至一个网页表格）被发送到比特币网络。在一些极端情况下，一笔比特币交易可以通过封包无线电、卫星或“短波、扩频或跳频以避免被侦测或阻塞通信的方式进行传输。一笔比特币交易甚至可被编为文字信息中的表情符号并被发表到在线论坛，或被发送成一条短信或一条Skype聊天信息。因为比特币将金钱变成了一种数据结构，所以在本质上是不可能阻止任何人创建并执行比特币交易的。
4. 一旦一笔比特币交易被发送到任意一个连接至比特币网络的节点，这笔交易将会被该节

点验证。如果交易被验证有效，该节点将会将这笔交易传播到这个节点所连接的其他节点；同时，交易发起者会收到一条表示交易有效并被接受的返回信息。如果这笔交易被验证为无效，这个节点会拒绝接受这笔交易且同时返回给交易发起者一条表示交易被拒绝的信息。

5. 为了避免垃圾信息的滥发、拒绝服务攻击或其他针对比特币系统的恶意攻击，每一个节点在传播每一笔交易之前均进行独立验证。< font color='red' > 一个异常交易所能到达的节点不会超过一个 < /font > 。

交易结构

1. 一笔比特币交易是一个含有输入值和输出值的数据结构，该数据结构植入了将一笔资金从初始点（输入值）转移至目标地址（输出值）的代码信息。
2. 比特币交易的输入值和输出值与账号或者身份信息无关。应该将它们理解成一种被特定秘密信息锁定的一定数量的比特币。只有拥有者或知晓这个秘密信息的人可以解锁。
3. 字段：版本(4bytes，明确这笔交易参照的规则)、输入量(1-9bytes，被包含的输入的数量)、输入(不定bytes，一个或多个交易输入)、输出量(1-9bytes，被包含的输出的数量)、输出(不定bytes，一个或多个交易输出)、时钟时间(4bytes, 一个UNIX时间戳或区块号)

交易的输入和输出

1. 比特币交易的基本单位是未经使用的一个交易输出，简称UTXO。UTXO是不能再分割、被所有者锁住或记录于区块链中的并被整个网络识别成货币单位的一定量的比特币货币。
2. 实际上，并不存在储存比特币地址或账户余额的地点，只有被所有者锁住的、分散的UTXO。
3. “一个用户的比特币余额”，这个概念是一个通过比特币钱包应用创建的派生之物。比特币钱包通过扫描区块链并聚合所有属于该用户的UTXO来计算该用户的余额。
4. < font color='red' > 在比特币的世界里既没有账户，也没有余额，只有分散到区块链里的UTXO < /font > 。一个UTXO可以是一“聪”的任意倍。
5. 尽管UTXO可以是任意值，但只要它被创造出来了，就像不能被切成两半的硬币一样不可再分了。如果一个UTXO比一笔交易所需量大，它仍会被当作一个整体而消耗掉，但同时会在交易中生成零头。大部分比特币交易都会产生找零。
6. 一笔比特币交易可以有任意数值，但必须从用户可用的UTXO中创建出来。用户不能再把UTXO进一步细分，就像不能把一元纸币撕开而继续当货币使用一样。用户的钱包应用通常会从用户可用的UTXO中选取多个可用的个体来拼凑出一个大于或等于一笔交易所需的比特币量。
7. 被交易消耗的UTXO被称为交易输入，由交易创建的UTXO被称为交易输出。通过这种方式，一定量的比特币价值在不同所有者之间转移，并在交易链中消耗和创建UTXO。一笔比特币交易通过使用所有者的签名来解锁UTXO，并通过使用新的所有者的比特币地址来锁定并创建UTXO。
8. 对于输出和输入链来说，有一个例外，它是一种特殊的交易类型，称为Coinbase交易。这是每个区块中的首个交易。这种交易存在的原因是作为对挖矿的奖励而产生全新的可

用于支付的比特币给“赢家”矿工。这也就是为什么比特币可以在挖矿过程中被创造出来。

9. 交易输出包含两部分：
10. 一定量的比特币，被命名为“聪”，是最小的比特币单位；
11. 一个锁定脚本，也被当作是“障碍”，提出支付输出所必须被满足的条件以“锁住”这笔总额
12. 交易输出把用聪表示的一定数量的比特币，和特定的定义了支付输出所必须被满足的条件的障碍，或者叫锁定脚本，关联到了一起。在大多数情况下，锁定脚本会把输出锁在一个特定的比特币地址上，从而把一定数量的比特币的所有权转移到新的所有者上。
13. 交易输入是指向UTXO的指针。它们指向特定的UTXO，并被交易哈希和在区块链中记录UTXO的序列号作为参考。
14. 当用户付款时，他的钱包通过选择可用的UTXO来构造一笔交易。比如说，要支付0.015比特币，钱包应用会选择一个0.01 UTXO和一个0.005 UTXO，使用它们加在一起来得到想要的付款额。
15. 一旦UTXO被选中，钱包会为每个UTXO生成包含签名的解锁脚本，由此让它们变得可以通过满足锁定脚本的条件来被支付。钱包把这些UTXO作为参考，并且连同解锁脚本一起作为输入加到交易中。

交易费用

1. 大多数交易包含交易费，这是为了在网络安全方面给比特币矿工一种补偿。大多数钱包自动计算并计入交易费。但是，如果你编程构造交易，或者使用命令行接口，你必须手动计算并计入这些费用。
2. 交易费可当作是为了包含（挖矿）一笔交易到下一个区块中的一种鼓励，也可当作是对于欺诈交易和任何种类的系统滥用，在每一笔交易上通过征收一笔小成本的税而造成的一种妨碍。交易费被挖出这个区块的矿工得到，并且记录在这个交易的区块链中。
3. 交易费不足或者没有交易费的交易可能会被推迟，基于尽力而为的原则在几个区块之后被处理，甚至可能根本不被处理。交易费不是强制的，而且没有交易费的交易也许最终会被处理，但是，包含交易费将提高处理优先级。
4. 起初，交易费是网络中的一个固定常数。渐渐地，交易费的结构被放宽了，以便被市场基于网络容量和交易量而强制影响。目前最小交易费被固定在每千字节0.0001比特币，或者说是每千字节万分之一比特币，最近一次改变是从千分之一比特币减少到这个数值的。大多数交易少于一千字节，但是那些包含多个输入和输出的交易尺寸可能更大。在未来的比特币协议修订版中，钱包应用预计会使用统计学分析，基于最近的几笔交易的平均费用，来计算最恰当的费用并附在交易上。
5. 交易的数据结构没有交易费的字段。相反地，交易费通过所有输入的总和，以及所有输出的总和之间的差来表示。从所有输入中扣掉所有输出之后的多余的量会被矿工收集走。
6. 如果你忘记了在手动构造的交易中增加找零的输出，系统会把找零当作交易费来处理。举例来说，如果你消耗了一个20比特币的UTXO来完成1比特币的付款，你必须包含一笔19比特币的找零回到你的钱包。否则，那剩下的19比特币会被当作交易费，并且会被挖

出你的交易到一个区块中的矿工收走。尽管你会受到高优先级的处理，并且让一个矿工喜出望外，但这很可能不是你想要的。

7. 高交易费不是因为付的钱很多，而是因为她的交易很复杂并且尺寸很大，交易费是与参加交易的比特币值无关的。

交易链条和孤立交易

1. 当一条交易链被整个网络传送时，他们并不能总是按照相同的顺序到达目的地。有时，子交易在父交易之前到达。在这种情况下，节点会首先收到一个子交易，而不能找到他参考的父交易。节点不会立即抛弃这个子交易，而是放到一个临时池中，并等着接收它的父交易，与此同时广播这个子交易给其他节点。没有父交易的交易池被称作孤立交易池。
2. 内存中储存的孤立交易数量是有限制的，这是为了防止针对比特币节点的拒绝服务攻击（DoS）。这个限制被定义在比特币涉及到的客户端的源代码中的MAX_ORPHAN_TRANSACTIONS。

比特币交易脚本和脚本语言

1. 比特币客户端通过执行一个用类Forth脚本语言编写的脚本验证比特币交易。锁定脚本被写入UTXO，同时它往往包含一个用同种脚本语言编写的签名。
2. 当一笔比特币交易被验证时，每一个输入值中的解锁脚本被与其对应的锁定脚本同时（互不干扰地）执行，从而查看这笔交易是否满足使用条件。
3. 大多数经比特币网络处理的交易是以“Alice付给Bob”的形式存在的。同时，它们是以一种称为“P2PKH”（Pay-to-Public-Key-Hash）脚本为基础的。然而，通过使用脚本来锁定输出和解锁输入意味着通过使用编程语言，比特币交易可以包含无限数量的条件。
4. 比特币交易验证并不基于一个不变的模式，而是通过运行脚本语言来实现。这种语言可以表达出多到数不尽的条件变种。这也是比特币作为一种“可编程的货币”所拥有的权力。

脚本创建（锁定与解锁）

1. 比特币的交易验证引擎依赖于两类脚本来验证比特币交易：一个锁定脚本和一个解锁脚本。
2. 锁定脚本是一个放在一个输出值上的“障碍”，同时它明确了今后花费这笔输出的条件。
3. 解锁脚本是一个“解决”或满足被锁定脚本在一个输出上设定的花费条件的脚本，同时它将允许输出被消费。解锁脚本是每一笔比特币交易输出的一部分，而且往往含有一个被用户的比特币钱包（通过用户的私钥）生成的数字签名。由于解锁脚本常常包含一个数字签名，因此它曾被称作ScriptSig。并非所有解锁脚本都一定会包含签名。
4. 每一个比特币客户端会通过同时执行锁定和解锁脚本来验证一笔交易。对于比特币交易中的每一个输入，验证软件会先检索输入所指向的UTXO。这个UTXO包含一个定义了花费条件的锁定脚本。接下来，验证软件会读取试图花费这个UTXO的输入中所包含的解锁脚本，并执行这两个脚本。
5. 首先，使用堆栈执行引擎执行解锁脚本。如果解锁脚本在执行过程中未报错（没有悬空操作符），主堆栈（非其它堆栈）将被复制，然后脚本将被执行。如果采用从解锁脚本

处复制而来的数据执行锁定脚本的结果为真，那么解锁脚本就成功地满足了锁定脚本所设置的条件，因而，该输入是一个能使用该UTXO的有效授权。如果在执行完组合脚本后的结果不是真，那么输入就不是有效的，因为它并未能满足UTXO中所设置的使用该笔资金的条件。

脚本语言

1. 比特币交易脚本语言，也称为脚本，是一种基于逆波兰表示法的基于堆栈的执行语言。
2. 比特币脚本语言被称为基于栈语言，因为它使用的数据结构被称为栈。
3. 如果堆栈顶部的结果显示为真（标记为{0×01}），即为任何非零值或脚本执行后堆栈为空情形，则交易有效。如果堆栈顶部的结果显示为假（0字节空值，标记为{}）或脚本执行被操作符禁止。

图灵非完备性

1. 比特币脚本语言包含许多操作，但都故意限定为一种重要的方式——没有循环或者复杂流控制功能以外的其他条件的流控制。
2. 这样就保证了脚本语言的图灵非完备性，这意味着脚本的复杂性有限，交易可执行的次数也可预见。
3. 脚本并不是一种通用语言，施加的这些限制确保该语言不被用于创造无限循环或其它类型的逻辑炸弹，这样的炸弹可以植入在一笔交易中，通过引起拒绝服务的方式攻击比特币网络。
4. 受限制的语言能防止交易激活机制被人当作薄弱环节而加以利用。

非主权验证

1. 比特币交易脚本语言是无国家主权的，没有国家能凌驾于脚本之上，也没有国家会在脚本被执行后对其进行保存。所以需要执行脚本的所有信息都已包含在脚本中。
2. 如果您的系统对一个脚本进行验证，可以确信的是每一个比特币网络中的其他系统也将对其进行验证，这意味着一个有效的交易对每个人而言都是有效的，而且每一个人都明白这一点。 < font color='red' > 这种对于结果的可预见性是比特币系统的一项重要良性特征 < /font > 。

标准交易

五大标准脚本分别为**P2PKH**、**P2PK**、**MS**（限**15**个密钥）、**P2SH**和**OP_Return**

P2PKH（Pay-to-Public-Key-Hash）

1. 比特币网络上的大多数交易都是P2PKH交易，此类交易都含有一个锁定脚本，该脚本由公钥哈希实现阻止输出功能，公钥哈希即为广为人知的比特币地址。由P2PKH脚本锁定的输出可以通过键入公钥和由相应私钥创设的数字签名得以解锁。

2. < Cafe Signature > < Cafe Public Key > OP_DUP OP_HASH160 < Cafe Public Key Hash > OP_EQUAL OP_CHECKSIG

P2PK（Pay-to-Public-Key）

1. 与P2PKH相比，P2PK模式更为简单。与P2PKH模式含有公钥哈希的模式不同，在P2PK脚本模式中，公钥本身已经存储在锁定脚本中，而且代码长度也更短。
2. P2PKH是由Satoshi创建的，主要目的一方面为使比特币地址更简短，另一方面也使之更方便使用。P2PK目前在Coinbase交易中最为常见，Coinbase交易由老的采矿软件产生，目前还没更新至P2PKH。
3. < Signature from Private Key A > < Public Key A > OP_CHECKSIG

多重签名

1. 多重签名脚本设置了这样一个条件，假如记录在脚本中的公钥个数为N，则至少需提供其中的M个公钥才可以解锁。这也被称为M-N组合，其中，N是记录在脚本中的公钥总个数，M是使得多重签名生效的公钥数阀值（最少数目）
2. OP_0 < Signature B > < Signature C > 2 < Public Key A > < Public Key B > < Public Key C > 3 OP_CHECKMULTISIG

数据输出（OP_RETURN操作符）

1. 比特币的分发和时间戳账户机制（也即区块链），其潜在运用将大大超越支付领域。许多开发者试图充分发挥交易脚本语言的安全性和可恢复性优势将其运用于电子公证服务、证券认证和智能协议等领域。
2. 比特币脚本语言的早期运用主要包括在区块链上创造出交易输出。
3. 在0.9版的比特币核心客户端上，通过采用OP_Return操作符最终实现了妥协。OP_Return允许开发者在交易输出上增加40字节的非交易数据。然后，与伪交易型的UTXO不同，OP_Return创造了一种明确的可复查的非交易型输出，此类数据无需存储于UTXO集
4. OP_Return输出被记录在区块链上，它们会消耗磁盘空间，也会导致区块链规模的增加，但它们不存储在UTXO集中，因此也不会使得UTXO内存膨胀，更不会以消耗代价高昂的内存为代价使全节点都不堪重负。
5. OP_RETURN不涉及可用于支付的解锁脚本的特点，OP_RETURN不能使用其输出中所锁定的资金，因此它也就没有必要记录在蕴含潜在成本的UTXO集中，所以OP_RETURN实际是没有成本的。OP_RETURN常为一个金额为0的比特币输出，因为任何与该输出相对应的比特币都会永久消失。
6. 假如一笔OP_RETURN遇到脚本验证软件，它将立即导致验证脚本和标记交易的行为无效。如果你碰巧将OP_RETURN的输出作为另一笔交易的输入，则该交易是无效的。
7. 一笔标准交易（通过了isStandard()函数检验的）只能有一个OP_RETURN输出。但是单个OP_RETURN输出能与任意类型的输出交易进行组合。

P2SH（Pay-to-Script-Hash）

1. P2SH旨在使复杂脚本的运用能与直接向比特币地址支付一样简单。在P2SH支付中，复杂的锁定脚本被电子指纹所取代，电子指纹为密码学哈希。当一笔交易试图支付UTXO时，要解锁支付脚本，它必须含有与哈希相匹配的脚本。
2. P2SH的含义是，向与该哈希匹配的脚本支付，当输出被支付时，该脚本将在后续呈现。
3. 在P2SH交易中，锁定脚本由哈希取代，哈希指代的是赎回脚本。因为它在系统中是在赎回时出现而不是以锁定脚本模式出现。
4. P2SH的另一重要特征是它能将脚本哈希编译为一个地址。P2SH地址是基于Base58编码的一个含有20个字节哈希的脚本，就像比特币地址是基于Base58编码的一个含有20个字节的公钥。由于P2SH地址采用5作为前缀，这导致基于Base58编码的地址以“3”开头。
5. 与直接使用复杂脚本以锁定输出的方式相比，P2SH具有以下特点：
6. 在交易输出中，复杂脚本由简短电子指纹取代，使得交易代码变短。
7. 脚本能被编译为地址，支付指令的发出者和支付者的比特币钱包不需要复杂工序就可以执行P2SH。
8. P2SH将构建脚本的重担转移至接收方，而非发送方。
9. P2SH将长脚本数据存储的负担从输出方（存储于UTXO集，影响内存）转移至输入方（仅存储于区块链）。
10. P2SH将长脚本数据存储的重担从当前（支付时）转移至未来（花费时）。
11. P2SH将长脚本的交易费成本从发送方转移至接收方，接收方在使用该笔资金时必须含有赎回脚本。
12. 不能将P2SH植入P2SH赎回脚本，因为P2SH不能自循环。也不能在赎回脚本中使用OP_RETURN，因为OP_RETURN的定义即显示不能赎回。
13. 因为赎回脚本只有在你试图发送一个P2SH输出时才会在比特币网络中出现，假如你将输出与一个无效的交易哈希锁定，则它将会被忽略。你将不能使用该笔资金，因为交易中含有赎回脚本，该脚本因是一个无效的脚本而不能被接受。
14. 这样的处理机制也衍生出一个风险，你能将比特币锁定在一个未来不能被花费的P2SH中。因为比特币网络本身会接受这一P2SH，即便它与无效的赎回脚本所对应（因为该赎回脚本哈希没有对其所表征的脚本给出指令）

七、比特币网络

节点类型及分工

1. 尽管比特币P2P网络中的各个节点相互对等，但是根据所提供的功能不同，各节点可能具有不同的分工。每个比特币节点都是路由、区块链数据库、挖矿、钱包服务的功能集合。
2. 每个节点都参与全网络的路由功能，同时也可能包含其他功能。每个节点都参与验证并传播交易及区块信息，发现并维持与对等节点的连接。
3. 一些节点保有一份完整的、最新的区块链拷贝，这样的节点被称为“全节点”。全节点能够

独立自主地校验所有交易，而不需借由任何外部参照。

4. 一些节点只保留了区块链的一部分，它们通过一种名为“简易支付验证（SPV）”的方式来完成交易验证。这样的节点被称为“SPV节点”，又叫“轻量级节点”。
5. 挖矿节点通过运行在特殊硬件设备上的工作量证明（proof-of-work）算法，以相互竞争的方式创建新的区块。一些挖矿节点同时也是全节点，保有区块链的完整拷贝；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。
6. 用户钱包也可以作为全节点的一部分，这在桌面比特币客户端中比较常见。当前，越来越多的用户钱包都是SPV节点，尤其是运行于诸如智能手机等资源受限设备上的比特币钱包应用；而这正变得越来越普遍。
7. 在比特币P2P协议中，除了这些主要的节点类型之外，还有一些服务器及节点也在运行着其他协议，例如特殊矿池挖矿协议、轻量级客户端访问协议等。
8. 总结：

9. 核心客户端(bitcoin core)：在比特币p2p网络中，包含钱包、矿工、完整区块链数据库、网络路由节点。

10. 完整区块链节点：在p2p网络中，包含完整区块链以及网络路由节点。
11. 独立矿工：包含具有完整区块链副本的挖矿功能、以及比特币p2p网络路由节点
12. 轻量(SPV)钱包：包含不具有区块链的钱包以及比特币p2p网络路由节点。
13. 矿池协议服务器：将运行其他协议的节点(例如矿池挖矿节点、stratum节点)连接至p2p网络的网关路由器
14. 挖矿节点：包含不具有区块链、但具备stratum协议节点(S)或者矿池挖矿协议节点(P)的挖矿功能
15. 轻量(SPV)Stratum钱包：包含不具有区块链的钱包、运行stratum协议的网络节点

网络发现

1. 新的网络节点必须发现至少一个网络中存在的节点并建立连接。节点通常采用TCP协议、使用8333端口（该端口号通常是比特币所使用的，除8333端口外也可以指定使用其他端口）与已知的对等节点建立连接。
2. 在建立连接时，该节点会通过发送一条包含基本认证内容的version消息开始“握手”通信过程，包括以下字段：PROTOCOL_VERSION、nLocalServices(一组该节点支持的本地服务列表，当前仅支持NODE_NETWORK)、nTime(当前时间)、addrYou(当前节点可见的远程节点的IP地址)、addrMe(本地节点所发现的本机IP地址)、subver(指示当前节点运行的软件类型的子版本号,例如：/Satoshi:0.9.2.1/)、BaseHeight(当前节点区块链的区块高度)
3. 虽然比特币网络中没有特殊节点，但是客户端会维持一个列表，那里列出了那些长期稳定运行的节点。这样的节点被称为“种子节点（seed nodes）”。新节点并不一定需要与种子节点建立连接，但连接到种子节点的好处是可以通过种子节点来快速发现网络中的其他节点。在比特币核心客户端中，是否使用种子节点是通过“-dnsseed”控制的。默认情况下，该选项设为1，即意味着使用种子节点。
4. 另一种方式是，起始时将至少一个比特币节点的IP地址提供给正在启动的节点（该节点

不包含任何比特币网络的组成信息）。在这之后，启动节点可以通过后续指令建立新的连接。用户可以使用命令行参数“-seednode”把启动节点“引荐”并连接到一个节点，并将该节点用作DNS种子。

5. 当建立一个或多个连接后，新节点将一条包含自身IP地址的addr消息发送给其相邻节点。相邻节点再将此条addr消息依次转发给它们各自的相邻节点，从而保证新节点信息被多个节点所接收、保证连接更稳定。另外，新接入的节点可以向它的相邻节点发送getaddr消息，要求它们返回其已知对等节点的IP地址列表。通过这种方式，节点可以找到需连接到的对等节点，并向网络发布它的消息以便其他节点查找。
6. 节点必须连接到若干不同的对等节点才能在比特币网络中建立通向比特币网络的种类各异的路径（path）
7. 由于节点可以随时加入和离开，通讯路径是不可靠的。因此，节点必须持续进行两项工作

8. 在失去已有连接时发现新节点

9. 并在其他节点启动时为其提供帮助

10. 节点启动时只需要一个连接，因为第一个节点可以将它引荐给它的对等节点，而这些节点又会进一步提供引荐。一个节点，如果连接到大量的其他对等节点，这既没必要，也是对网络资源的浪费。在启动完成后，节点会记住它最近成功连接的对等节点；因此，当重新启动后它可以迅速与先前的对等节点网络重新建立连接。如果先前的网络的对等节点对连接请求无应答，该节点可以使用种子节点进行重启动。

11. bitcoin-cli getpeerinfo 列出对等节点的连接信息
12. 如果节点持续某个连接长达90分钟没有任何通信，它会被认为已经从网络中断开，网络将开始查找一个新的对等节点。因此，比特币网络会随时根据变化的节点及网络问题进行动态调整，不需经过中心化的控制即可进行规模增、减的有机调整。

全节点

1. 完整区块链节点保有完整的、最新的包含全部交易信息的比特币区块链拷贝，这样的节点可以独立地进行建立并校验区块链，从第一区块（创世区块）一直建立到网络中最新的区块。完整区块链节点可以独立自主地校验任何交易信息，而不需要借助任何其他节点或其他信息来源。

交换“库存清单”

1. 一个全节点连接到对等节点之后，第一件要做的事情就是构建完整的区块链。如果该节点是一个全新节点，那么它就不包含任何区块链信息，< font color='red' > 它只知道一个区块——静态植入在客户端软件中的创世区块 < /font > 。新节点需要下载从0号区块（创世区块）开始的数十万区块的全部内容，才能跟网络同步、并重建全区块链。
2. 通过分摊工作量的方式防止单一对等节点被批量请求所压垮。该节点会追踪记录其每个对等节点连接上“正在传输”（指那些它已经发出了请求但还没有接收到）的区块数量，且检查该数量有没有超过上限（MAX_BLOCKS_IN_TRANSIT_PER_PEER）。如果一个

节点需要更新大量区块，它会在上一请求完成后才发送对新区块的请求，从而允许对等节点控制更新速度，不至于压垮网络。

3. 一个节点上线，会从发送getblocks消息开始，收到一个inv响应，接着开始下载缺失的区块库存清单和区块广播协议。

简易支付验证（SPV）节点

1. SPV节点只需下载区块头，而不用下载包含在每个区块中的交易信息。由此产生的不含交易信息的区块链，大小只有完整区块链的1/1000。SPV节点不能构建所有可用于消费的UTXO的全貌，这是由于它们并不知道网络上所有交易的完整信息。
2. 简易支付验证是通过参考交易在区块链中的深度，而不是高度，来验证它们。一个拥有完整区块链的节点会构造一条验证链，这条链是由沿着区块链按时间倒序一直追溯到创世区块的数千区块及交易组成。而一个SPV节点会验证所有区块的链（但不是所有的交易），并且把区块链和有关交易链接起来。
3. 一个全节点要检查第300,000号区块中的某个交易，它会把从该区块开始一直回溯到创世区块的300,000个区块全部都链接起来，并建立一个完整的UTXO数据库，通过确认该UTXO是否还未被支付来证实交易的有效性。
4. SPV节点则不能验证UTXO是否还未被支付。相反地，SPV节点会在该交易信息和它所在区块之间用merkle路径（见“7.7 Merkle 树”）建立一条链接。然后SPV节点一直等待，直到序号从300,001到300,006的六个区块堆叠在该交易所在的区块之上，并通过确立交易的深度是在第300,006区块~第300,001区块之下来验证交易的有效性。
5. SPV节点毫无疑问可以证实某个交易的存在性，但它不能验证某个交易（譬如同一个UTXO的双重支付）不存在，这是因为SPV节点没有一份关于所有交易的记录。
6. 完整的区块链节点是通过检查整个链中在它之下的数千个区块来保证这个UTXO没有被支付，从而验证交易。而SPV节点是通过检查在其上面的区块将它压在下面的深度来验证交易。
7. SPV节点对特定数据的请求可能无意中透露了钱包里的地址信息。例如，监控网络的第三方可以跟踪某个SPV节点上的钱包所请求的全部交易信息，并且利用这些交易信息把比特币地址和钱包的用户关联起来，从而损害了用户的隐私。

Bloom过滤器

1. Bloom过滤器是一个允许用户描述特定的关键词组合而不必精确表述的基于概率的过滤方法。它能让用户在有效搜索关键词的同时保护他们的隐私。在SPV节点里，这一方法被用来向对等节点发送交易信息查询请求，同时交易地址不会被暴露。
2. Bloom过滤器被用来过滤SPV节点从对等节点里收到的交易信息。SPV会建立一个只能和SPV节点钱包里的地址匹配的过滤器。随后，SPV节点会向对等节点发送一条包含需在该连接中使用的过滤器的filterload消息。当过滤器建好之后，对等节点将每个交易的输出值代入过滤器中验证。那些匹配的交易会被传送回SPV节点。
3. 为回应来自SPV节点的getdata信息，对等节点会发出一条只含有和过滤器匹配的区块的区块头信息，以及与之相匹配的交易的merkle树。这一对等节点还会发出一条相匹配的交易的tx消息。

交易池

1. 比特币网络中几乎每个节点都会维护一份未确认交易的临时列表，被称为内存池或交易池。节点们利用这个池来追踪记录那些被网络所知晓、但还未被区块链所包含的交易。保存用户钱包的节点会利用这个交易池来记录那些网络已经接收但还未被确认的、属于该用户钱包的预支付信息。
2. 有些节点的实现还维护一个单独的孤立交易池。如果一个交易的输入与某未知的交易有关，如与缺失的父交易相关，该孤立交易就会被暂时储存在孤立交易池中直到父交易的信息到达。
3. 当一个交易被添加到交易池中，会同时检查孤立交易池，看是否有某个孤立交易引用了此交易的输出（子交易）。任何匹配的孤立交易会被进行验证。如果验证有效，它们会从孤立交易池中删除，并添加到交易池中，使以其父交易开始的链变得完整。对新加入交易池的交易来说，它不再是孤立交易。前述过程重复递归寻找进一步的后代，直至所有的后代都被找到。通过这一过程，一个父交易的到达把整条链中的孤立交易和它们的父级交易重新结合在一起，从而触发了整条独立交易链进行级联重构。
4. 有些比特币客户端的实现还维护一个UTXO数据库，也称UTXO池，是区块链中所有未支付交易输出的集合。“UTXO池”的名字听上去与交易池相似，但它代表了不同的数据集。UTXO池不同于交易池和孤立交易池的地方在于，它在初始化时不为空，而是包含了数以百万计的未支付交易输出条目，有些条目的历史甚至可以追溯至2009年。UTXO池可能会被安置在本地内存，或者作为一个包含索引的数据库表安置在永久性存储设备中。
5. 交易池和孤立交易池代表的是单个节点的本地视角。取决于节点的启动时间或重启时间，不同节点的两池内容可能有很大差别。相反地，UTXO池代表的是网络的突显共识，因此，不同节点间UTXO池的内容差别不大。此外，交易池和孤立交易池只包含未确认交易，而UTXO池之只包含已确认交易。

警告消息

1. 警告消息是比特币的“紧急广播系统”，比特币核心开发人员可以借此功能给所有比特币节点发送紧急文本消息。这一功能是为了让核心开发团队将比特币网络的严重问题通知所有的比特币用户，例如一个需要用户采取措施的的严重bug。警告系统迄今为止只被用过几次，最严重的一次是在2013年，一个关键的数据库缺陷导致比特币区块链中出现了一个多区块分叉。
2. 警告通过公钥进行加密签名。对应的私钥是由核心开发团队的一些特定成员所持有。这样的数字签名可以确保虚假警告不会在网络中传播。
3. 收到警告消息的节点会验证该消息，检查是否过期，并传播给其所有对等节点，从而保证了整个网络中的快速传播。

八、区块链

简介

1. 比特币核心客户端使用Google的LevelDB数据库存储区块链元数据。区块被从后向前有序地链接在这个链条里，每个区块都指向前一个区块。区块链经常被视为一个垂直的栈，第一个区块作为栈底的首区块，随后每个区块都被放置在其他区块之上。
2. “高度”表示区块与首区块之间的距离；“顶部”或“顶端”表示最新添加的区块。
3. 对每个区块头进行SHA256加密哈希，可生成一个哈希值。通过这个哈希值，可以识别出区块链中的对应区块。同时，每一个区块都可以通过其区块头的“父区块哈希值”字段引用前一区块（父区块）
4. 虽然每个区块只有一个父区块，但可以暂时拥有多个子区块。每个子区块都将同一区块作为其父区块，并且在“父区块哈希值”字段中具有相同的（父区块）哈希值。一个区块出现多个子区块的情况被称为“区块链分叉”。区块链分叉只是暂时状态，只有当多个不同区块几乎同时被不同的矿工发现时才会发生。
5. 尽管一个区块可能会有不止一个子区块，但每一区块只有一个父区块，这是因为一个区块只有一个“父区块哈希值”字段可以指向它的唯一父区块。一个长区块链的存在可以让区块链的历史不可改变，这也是比特币安全性的一个关键特征。
6. 区块结构：
7. 区块大小 4字节 用字节表示的该字段之后的区块大小
8. 区块头 80字节 组成区块头的几个字段
9. 交易计数器 1-9（可变整数）字节 交易的数量
10. 交易 可变的 记录在区块里的交易信息

区块头

1. 区块头由三组区块元数据组成：
2. 首先是一组引用父区块哈希值的数据，这组元数据用于将该区块与区块链中前一区块相连接
3. 第二组元数据，即难度、时间戳和nonce，与挖矿竞争相关
4. 第三组元数据是merkle树根，一种用来有效地总结区块中所有交易的数据结构
5. 区块标识符：区块头哈希值和区块高度。区块主标识符是它的加密哈希值，一个通过SHA256算法对区块头进行二次哈希计算而得到的数字指纹。产生的32字节哈希值被称为区块哈希值，但是更准确的名称是：区块头哈希值，因为只有区块头被用于计算。
6. 区块哈希值实际上并不包含在区块的数据结构里，不管是该区块在网络上传输时，抑或是它作为区块链的一部分被存储在某节点的永久性存储设备上时。相反，区块哈希值是当该区块从网络被接收时由每个节点计算出来的。区块的哈希值可能会作为区块元数据的一部分被存储在一个独立的数据库表中，以便于索引和更快地从磁盘检索区块。
7. 第二种识别区块的方式是通过该区块在区块链中的位置，即“区块高度（block height）”。第一个区块，其区块高度为0，和之前哈希值000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f所引用的

区块为同一个区块。

8. 一个区块的区块哈希值总是能唯一地识别出一个特定区块。一个区块也总是有特定的区块高度。但是，一个特定的区块高度并不一定总是能唯一地识别出一个特定区块。更确切地说，两个或者更多数量的区块也许会为了区块链中的一个位置而竞争。

区块

1. 因为创世区块被编入到比特币客户端软件里，所以每一个节点都始于至少包含一个区块的区块链，这能确保创世区块不会被改变。每一个节点都“知道”创世区块的哈希值、结构、被创建的时间和里面的一个交易。因此，每个节点都把该区块作为区块链的首区块，从而构建了一个安全的、可信的区块链的根。

Merkle 树

1. 在比特币网络中，Merkle树被用来归纳一个区块中的所有交易，同时生成整个交易集合的数字指纹，且提供了一种校验区块是否存在某交易的高效途径。
2. 生成一棵完整的Merkle树需要递归地对哈希节点对进行哈希，并将新生成的哈希节点插入到Merkle树中，直到只剩一个哈希节点，该节点就是Merkle树的根。
3. 因为Merkle树是二叉树，所以它需要偶数个叶子节点。如果仅有奇数个交易需要归纳，那最后的交易就会被复制一份以构成偶数个叶子节点，这种偶数个叶子节点的树也被称为平衡树。

挖矿与共识

1. 挖矿是一种将结算所去中心化的过程，每个结算所对处理的交易进行验证和结算。挖矿保护了比特币系统的安全，并且实现了在没有中心机构的情况下，也能使整个比特币网络达成共识。
2. 比特币以一个确定的但不断减慢的速率被铸造出来。大约每十分钟产生一个新区块，每一个新区块都伴随着一定数量从无到有的全新比特币。
3. 总量有限并且发行速度递减创造了一种抗通胀的货币供应模式。法币可被中央银行无限制地印刷出来，而比特币永远不会因超额印发而出现通胀。
4. 最重要并且最有争议的一个结论是一种事先确定的发行速率递减的货币发行模式会导致货币通货紧缩（简称通缩）。通缩是一种由于货币的供应和需求不匹配导致的货币增值的现象。它与通胀相反，价格通缩意味着货币随着时间有越来越强的购买力。

去中心化共识

1. 共识是数以千计的独立节点遵守了简单的规则通过异步交互自发形成的产物。所有的比特币属性，包括货币、交易、支付以及不依靠中心机构和信任的安全模型等都是这个机制的衍生物。比特币的去中心化共识由所有网络节点的4种独立过程相互作用而产生：

2. 每个全节点依据综合标准对每个交易进行独立验证

3. 通过完成工作量证明算法的验算，挖矿节点将交易记录独立打包进新区块
4. 每个节点独立的对新区块进行校验并组装进区块链

5. 每个节点对区块链进行独立选择，在工作量证明机制下 < font color='red' > 选择累计工作量最大的区块链 < /font >

6. 交易的优先级是由交易输入所花费的UTXO的“块龄”决定，交易输入值高、“块龄”大的交易比那些新的、输入值小的交易拥有更高的优先级。如果区块中有足够的空间，高优先级的交易行为将不需要矿工费。

7. 交易输入的值是由比特币单位“聪”（1亿分之1个比特币）来表示的。UTXO的“块龄”是自该UTXO被记录到区块链为止所经历过的区块数，即这个UTXO在区块链中的深度。
8. 一个交易想要成为“较高优先级”，需满足的条件：优先值大于57,600,000，相当于一个比特币（即1亿聪），年龄为一天（144个区块），交易的大小为250个字节。
9. “挖矿节点在填充这50K字节的时候，会优先考虑这些最高优先级的交易，不管它们是否包含了矿工费。这种机制使得高优先级交易即便是零矿工费，也可以优先被处理。然后，挖矿节点会选出那些包含最小矿工费的交易，并按照“每千字节矿工费”进行排序，优先选择矿工费高的交易来填充剩下的区块，区块大小上限为MAX_BLOCK_SIZE。
10. 比特币交易中没有过期、超时的概念，一笔交易现在有效，那么它就永远有效。然而，如果一笔交易只在全网广播了一次，那么它只会保存在一个挖矿节点的内存中。因为内存池是以未持久化的方式保存在挖矿节点存储器中的，所以一旦这个节点重新启动，内存池中的数据就会被完全擦除。而且，即便一笔有效交易被传播到了全网，如果它长时间未处理，它将从挖矿节点的内存池中消失。如果交易本应该在一段时间内被处理而实际没有，那么钱包软件应该重新发送交易或重新支付更高的矿工费。
11. 节点需要填充难度目标值，为了使得该区块有效，这个字段定义了所需满足的工作量证明的难度。难度在区块中以“尾数-指数”的格式，编码并存储，这种格式称作“难度位”。这种编码的首字节表示指数，后面的3字节表示尾数(系数)。以区块277316为例，难度位的值为0x1903a30c，0x19是指数的十六进制格式，后半部0x03a30c是系数。
12. 挖矿的目标是找到一个使区块头哈希值小于难度目标的nonce。挖矿节点通常需要尝试数十亿甚至数万亿个不同的nonce取值，直到找到一个满足条件的nonce值。
13. 难度的调整是在每个完整节点中独立自动发生的。每2016个区块中的所有节点都会调整难度。难度的调整公式是由最新2,016个区块的花费时长与20160分钟（两周，即这些区块以10分钟一个速率所期望花费的时长）比较得出的。难度是根据实际时长与期望时长的比值进行相应调整的（或变难或变易）。简单来说，如果网络发现区块产生速率比10分钟要快时会增加难度。如果发现比10分钟慢时则降低难度。
14. 为了防止难度的变化过快，每个周期的调整幅度必须小于一个因子（值为4）。如果要调整的幅度大于4倍，则按4倍调整。由于在下一个2,016区块的周期不平衡的情况会继续存在，所以进一步的难度调整会在下一周期进行。因此平衡哈希计算能力和难度的巨大差异有可能需要花费几个2,016区块周期才会完成。

区块链的组装与选择

1. 节点维护三种区块：第一种是连接到主链上的，第二种是从主链上产生分支的（备用链），最后一种是在已知链中没有找到已知父区块的。在验证过程中，一旦发现有不符合标准的地方，验证就会失败，这样区块会被节点拒绝，所以也不会加入到任何一条链

中。

2. 任何时候，主链都是累计了最多难度的区块链。在一般情况下，主链也是包含最多区块的那个链，除非有两个等长的链并且其中一个有更多的工作量证。主链也会有一些分支，这些分支中的区块与主链上的区块互为“兄弟”区块。这些区块是有效的，但不是主链的一部分。 保留这些分支的目的是如果在未来的某个时刻它们中的一个延长了并在难度值上超过了主链，那么后续的区块就会引用它们。

区块链分叉

1. 每一个节点总是选择并尝试延长代表累计了最大工作量证明的区块链，也就是最长的或最大累计难度的链。节点通过将记录在每个区块中的难度加总起来，得到建立这个链所要付出的工作量证明的总量。只要所有的节点选择最长累计难度的区块链，整个比特币网络最终会收敛到一致的状态。分叉即在不同区块链间发生的临时差异，当更多的区块添加到了某个分叉中，这个问题便会迎刃而解。
2. 两个区块的分叉是有可能的，这种情况发生在因先前分叉而相互对立起来的矿工，又几乎同时发现了两个不同区块的解。然而，这种情况发生的几率是很低的。单区块分叉每周都会发生，而双块分叉则非常罕见。

矿池

1. 矿池通过专用挖矿协议协调成百上千的矿工。个人矿工在建立矿池账号后，设置他们的矿机连接到矿池服务器。他们的挖矿设备在挖矿时保持和矿池服务器的连接，和其他矿工同步各自的工作。这样，矿池中的矿工分享挖矿任务，之后分享奖励。
2. 成功出块的奖励支付到矿池的比特币地址，而不是单个矿工的。一旦奖励达到一个特定的阈值，矿池服务器便会定期支付奖励到矿工的比特币地址。通常情况下，矿池服务器会为提供矿池服务收取一个百分比的费用。
3. 大部分矿池是“托管的”，意思是有一个公司或者个人经营一个矿池服务器。矿池服务器的所有者叫矿池管理员，同时他从矿工的收入中收取一个百分比的费用。
4. 托管矿池存在管理人作弊的可能，管理人可以利用矿池进行双重支付或使区块无效。此外，中心化的矿池服务器代表着单点故障。如果因为拒绝服务攻击服务器挂了或者被减慢，池中矿工就不能采矿。
5. P2Pool通过将矿池服务器的功能去中心化，实现一个并行的类似区块链的系统，名叫份额链。P2Pool采矿方式比在矿池中采矿要复杂的多，因为它要求矿工运行空间、内存、带宽充足的专用计算机来支持一个比特币的完整节点和P2Pool节点软件。P2Pool矿工连接他们的采矿硬件到本地P2Pool节点，它通过发送区块模板到矿机来模拟一个矿池服务器的功能。
6. 最近，在集中式矿池已经接近产生51%攻击的担忧下，P2Pool的份额增长显著。

共识攻击

1. 比特币的共识机制指的是，被矿工（或矿池）试图使用自己的算力实行欺骗或破坏的难度很大，至少理论上是这样。就像我们前面讲的，比特币的共识机制依赖于这样一个前提，那就是绝大多数的矿工，出于自己利益最大化的考虑，都会通过诚实地挖矿来维持

整个比特币系统。然而，当一个或者一群拥有了整个系统中大量算力的矿工出现之后，他们就可以通过攻击比特币的共识机制来达到破坏比特币网络的安全性和可靠性的目的。

2. 共识攻击只能影响整个区块链未来的共识，或者说，最多能影响不久的过去几个区块的共识（最多影响过去10个块）。而且随着时间的推移，整个比特币块链被篡改的可能性越来越低。
3. 共识攻击也不会影响用户的私钥以及加密算法（ECDSA）。共识攻击也不能从其他的钱包那里偷到比特币、不签名地支付比特币、重新分配比特币、改变过去的交易或者改变比特币持有纪录。共识攻击能够造成的唯一影响是影响最近的区块（最多10个）并且通过拒绝服务来影响未来区块的生成。
4. 共识攻击的一个典型场景就是“51%攻击”。
5. 双重支付可以有两种方式：要么是在交易被确认之前，要么攻击者通过块链分叉来完成。进行51%攻击的人，可以取消在旧分叉上的交易记录，然后在新分叉上重新生成一个同样金额的交易，从而实现双重支付。
6. 为了避免这类攻击，售卖大宗商品的商家应该在交易得到全网的6个确认之后再交付商品。或者，商家应该使用第三方的多方签名的账户进行交易，并且也要等到交易账户获得全网多个确认之后再交付商品。一条交易的确认数越多，越难被攻击者通过51%攻击篡改。
7. 共识攻击中除了“双重支付”攻击，还有一种攻击场景就是拒绝对某个特定的比特币地址提供服务。一个拥有了系统中绝大多数算力的攻击者，可以轻易地忽略某一笔特定的交易。如果这笔交易存在于另一个矿工所产生的区块中，该攻击者可以故意分叉，然后重新产生这个区块，并且把想忽略的交易从这个区块中移除。这种攻击造成的结果就是，只要这名攻击者拥有系统中的绝大多数算力，那么他就可以持续地干预某一个或某一批特定钱包地址产生的所有交易，从而达到拒绝为这些地址服务的目的。
8. 一些安全研究组织利用统计模型得出的结论是，算力达到全网的30%就足以发动51%攻击了。

九、竞争币、竞争块链

1. 比特币的核心：一种基于工作量证明的去中心化的一致性机制。
2. 竞争块链通过实现一致性和分布式账簿机制来给诸如合同、名字注册和其他一些应用提供服务。竞争块链使用的是和比特币一样的创建块的机制，有时也会采用货币或代币的支付机制，但它们的主要目的不是为了维持一个货币系统。

元币平台

1. 元币和元块链是在比特币之上实现的软件层，也可以认为是覆盖在比特币系统之上的平台/协议，或者是在一个币中币的实现。元币的第一个实现利用了大量的 hack 技巧把元数据添加到比特币块链中，比如使用比特币地址编码数据，或者利用空白的交易字段存放新协议层增加的这些元数据。
2. 染色币：一种在少量比特币上存储信息的一种元协议。一个“被染色的”币，是一定数额的重新用于表达另一种资产的比特币。

3. 染色币由特殊的钱包管理，这类钱包存储和解析依附在染色币上的元信息。用户在使用这类钱包的时候，可以通过增加有着某种特殊含义的标签的方式，将一般的比特币“染色”为染色币这种标签的内容可以表示股票证明、优惠券信息、实际财产、商品或者可收集的代币等等。如何书写和解读这类标签，完全取决于给这枚比特币“染色”的人，他可以决定附着在这部分比特币上的元信息属性。比如信息类型、能不能再分割、某种符号或描述，或者其他的相关信息。这部分比特币一旦被染色，这些币可以用来交易、分割、合并和获取利息等。被染色的比特币也可用通过删除附着信息的方式，也能将“被染色的”比特币恢复为普通比特币。
4. 万事达币是另一个建立在比特币之上的协议，该协议支持多个平台对比特币系统的扩展。
5. 合约币是另一个建立在比特币系统之上的协议层。合约币拥有用户货币、可交易代币、金融手段、去中心化财产交易和其他一些功能。合约币利用比特币脚本语言中的OP_RETURE操作符记录元信息来增加比特币交易的额外信息。合约币使用名为XCP的代币维持整个系统的运行。
6. 绝大多数的山寨币都来自比特币源代码的克隆，少数则没有使用比特币的任何源码，仅仅是借鉴了块链的模型后自己实现。竞争币或竞争块链（下一节会讲到）都是运行在自己块链上的独立的块链实现。之所以以命名区分，主要是因为竞争币主要用做货币，而竞争块链则不是。
7. 莱特币
8. 莱特币使用了其他工作量证明算法（scrypt）的加密货币，这种算法起初是为了防止密码遭暴力破解而设计的，目标是通过使用这种消耗内存的算法来实现一种不依赖GPU和ASIC芯片的电子货币。
9. 把新块产生的时间从比特币的10分钟缩短为2分半钟。
10. 如果把比特币看作电子货币中的金币的话，那么莱特币的愿景就是当电子货币系统中的银币，谋求成为比特币的一种轻量的替代货币。
11. 考虑到莱特币8,400万的货币总量和相对更快的确认速度，很多莱特币的拥趸相信与比特币相比，莱特币更适合零售业的交易
12. 竞争币区别于比特币的三点主要不同：
13. 货币策略不同
14. 基于工作量证明的一致性机制不同
15. 一些特殊的功能，比如更强的匿名性等等
16. 评估竞争币的价值
17. 这款竞争币有没有引入重大的创新？
18. 如果有，那么这项创新是不是足够吸引使用比特币的用户转移过来？
19. 这款竞争币是不是致力于某一细分领域或应用？
20. 这款竞争币可以吸引到足够多的矿工来抵御一致性攻击吗？

21. 这款竞争币的市场总值是多少？
22. 整个系统的用户/钱包规模大概是多少？
23. 接受其支付的商家有多少？
24. 整个系统每日的交易数是多少？
25. 交易总量是多少？

26. < font color='red' > 比特币的工作量证明机制只有一个目的：维护比特币系统的安全。 < /font > 跟维护一个传统货币系统比起来，挖矿的成本并不高。然而，某些批评者认为挖矿这一行为是一种浪费。

非货币性竞争区块链

1. 域名币是比特币源代码的首个克隆产物，它是一种使用区块链的去中心化平台，用来注册和转让键－值对。域名币支持全球的域－名注册，类似因特网上的域－名注册系统。目前域名币作为根域名.bit的替代性域名服务（DNS）使用。也可以用来注册其他命名空间下的名称和键－值对，例如存储邮件地址、密钥、SSL证书、文件签名、投票系统和股票凭证之类，以及许多其他应用。
2. Bitmessage是一个实现了去中心化安全消息服务的比特币竞争币区块链，其本质上是一个无服务器的加密电子邮件系统。Bitmessage可以让用户通过一个Bitmessage地址来编写和发送消息。这些消息的运作方式与比特币交易大致相同，但区别在于消息是短暂瞬态的——如果超过两天还没被传送至目的节点，消息将会丢失。发送方和接收方都是假名，除了一个bitmessage地址外，他们没有其他的身份标识。
3. 以太坊

4. 以太坊是一种图灵完备的平台，基于区块链账簿，用于合约的处理和执行。它不是比特币的一个克隆，而是完完全全独立的一种设计和实现。

5. 以太坊内置一种叫做ether的货币，该货币是付合约执行之费用所必须的。以太坊区块链记录的东西叫做合约，所谓合约，就是一种低级二进制码，也是一种图灵完备语言。本质上，合约其实是运行在以太坊系统中各个节点上的程序。
6. 这些程序可以存储数据、支付及收取、存储ether以及执行无穷范围（因此才叫图灵完备）的计算行为，在系统中充当去中心化的自治软件代理。

十、比特币安全

安全准则

1. 核心准则是去中心化，这一点对安全性具有重要意义。网络的安全性是基于工作量证明而非访问控制，比特币网络可以对所有人开放，也无需对比特币流量进行加密。
2. 一笔比特币交易只授权向指定接收方发送一个指定数额，并且不能被修改或伪造。它不会透露任何个人信息，例如当事人的身份，也不能用于权限外的支付。因此，比特币的支付网络并不需要加密或防窃听保护。

3. 比特币的去中心化安全模型很大程度上将权力移交到用户手上，随之而来的是用户们保管好密钥的责任。
4. 在比特币里，共识系统创建了一个可信的完全去中心化的公开账本，一个正确验证过的区块使用创世块作为信任的根源，建立一条直至当前区块的可信任链。比特币系统可以并应该使用区块链作为它们的信任根源。在设计一个多系统服务机制的比特币应用时，你应该仔细确认安全体系，以确保对它的信任能有据可依。最终，唯一可确信无疑的是一条完全有效的区块链。如果你的应用程序或明或暗地信赖于区块链以外的东西，就该引起重视，因为它可能会引入漏洞。一个不错的方法评估你应用程序的安全体系：单独考量每个组件，设想该组件被完全攻破并被坏人掌控的场景。依次取出应用程序的每个组件，并评估它被攻破时对整体安全的影响。如果你的应用程序的安全性在该组件沦陷后大打折扣，那就说明你已经对这些组件过度信任了。
5. 一个没有漏洞的比特币应用程序应该只受限于比特币的共识机制，这意味着其安全体系的信任源于比特币最坚固的部分。

用户最佳安全实践

1. 比特币物理存储
2. 硬件钱包：只提供了有限的接口，从而可以给非专业用户提供近乎万无一失的安全等级
3. 平衡风险：为了防止被盗窃，其主人曾之前采取了一系列复杂的操作去加密备份。结果他们不慎丢失了加密的密钥，使得备份变得毫无价值，白白失去了一大笔财富。
4. 分散风险：将风险分散到不同类型的比特币钱包。审慎的用户应该只留一小部分（或许低于5%）的比特币在一个在线的或手机钱包，就像零用钱一样，其余的部分应该采用不同存储机制分散开来，诸如电脑钱包和离线（冷存储）钱包。
5. 多重签名管理： 多重签名比特币地址需要多个签名才能支付，从而保证资金的安全
6. 存活能力：一个非常重要却又常常被忽视的安全性考虑是可用性，尤其是在密钥持有者丧失工作能力或死亡的情况下。

区块链的根本属性是去中心化，而去中心化的依托是共识机制。

在了解共识机制之前，先来看两个古老的引入问题：类两军问题、拜占庭将军问题。

类两军问题：

> 古代有两个相距很远的军队要传递信息，
>
> 蓝军派遣一个信使去跟红军说：有本事把意大利炮拿过来！
>
> 红军收到后回复蓝军说：收到指令。
>
> 蓝军要给出确认答复：知道你收到指令了！
>
> 红军继续给出答复：知道你知道我知道指令了！
>
> ….

拜占庭将军问题：

> 拜占庭罗马帝国在军事行动中，采取将军投票策略来决定进攻还是撤退，即如果多数人决定进攻，就整体确定进攻策略。但是军队中如果有奸细（将军可能反水、传令官可能误传），如何保证最后投票真实反映忠诚将军的决策？
>
> 拜占庭帝国周围有10个小国，它们饱受拜占庭欺压，却只有同一时间有6个以上国家进攻才有可能打败拜占庭帝国，非则一定战败。
>
> 难点在于：古时候军队之间的通信完全依赖于人，如果军队中有奸细，无论是将军反水还是传令官误传，都会是另外9个国家收到假消息，从而造成作战失败。如果你是国王，该如何判断一定会有另外5个以上国家与你并肩作战？毕竟一不小心，就亡国了。

由于类似于以上这样的问题存在，共识的必要性浮现出来。

九种共识机制比较

区块链上的共识机制有多种，但任何一种都不是完美无缺，或者说适用于所有应用场景的。

1. 工作量证明(POW)

工作量证明（Proof of Work，简称PoW）通常只能从结果证明，因为监测工作过程通常是繁琐且低效的。

比特币在区块的生成过程种使用了PoW机制，一个符合要求的区块哈希值由N个前导零构成，零的个数取决于网络的难度值。要得到合理的区块哈希值需要经过大量的尝试计算，计算时间取决于机器的哈希运算速度。当某个节点提供出一个合理的区块哈希值，说明该节点确实经过了大量的尝试计算，但是并不能得出计算次数，因为寻找合理的哈希值是一个概率事件。当节点拥有占全网n%的算力时，该节点既有n%的概率找到区块哈希值。

PoW依赖机器进行数学运算来获取记账权，资源消耗大、共识机制高、可监管性弱，同时每次达成共识需要全网共同参与运算，性能效率比较低，容错性方便允许全网50%节点出错。

PoW的优点：完全去中心化，节点自由进出。

PoW的缺点：目前比特币已经吸引全球大部分的算力，其他再使用PoW共识机制的区块链应用很难获得相同的算力来保障自身安全；挖矿造成大量的资源浪费；共识达成的周期较长。

使用PoW的项目有：比特币、以太坊的前三个阶段（Frontier前沿、Homestead家园、Metropolis大都会）。以太坊的第四个阶段 Serenity宁静 将采用权益证明机制（**POS**）。

2. 权益证明（P0S）

权益证明（Proof of Stake，简称PoS）由Quantum Mechanic2011年在比特币论坛讲座上首先提出，后经Peercoin（点点币）和NXT（未来币）以不同思路实现。

PoS的主要理念是节点记账权的获得难度与节点持有的权益成反比，相比PoW，其在一定程度上减少了数学运算带来的资源消耗，性能也得到了相应的提升，但依然是基于哈希运算，竞争获取记账权的方式，可监管性弱。该共识机制的容错性和PoW相同。它是PoW的一种升级，根据每个节点所占代币的比例和时间，等比例地降低挖矿难度，从而加快找到随机数的速度。

在PoW中，一个用户可能拿1000美元来购买计算机，并加入网络来挖矿以此产生新区块，从而得到奖励。而在PoS中，用户可以拿1000美元购买等价的代币，并把这些代币当作押金放入PoS机制中，这样用户就有机会产生新区块而得到奖励。

总体而言，这个系统中存在一个持币人的集合，他们把手中的代币放入PoS机制中，这样他们就变成验证者。比如对区块链最前面的一个区块而言，PoS算法在验证者中随机选择一个（选择验证者的权重依据他们投入的代币量，比如一个投入押金为1W代币的验证者被选择的概率是一个投入1K代币验证者的10倍），给他权利产生下一个区块。如果在一定时间内，这个验证者没有产生一个区块，则选出第二个验证者代替产生新区块。与PoW一样，PoS以最长的链为准。

随着规模经济（指扩大生产规模引起经济效益增加的现象）的消失，中心化所带来的风险减小了。价值1000万美元的代币带来的回报不多不少，是价值100万美元代币的10倍，不会有人因为负担得起大规模生产工具而得不到成比例的额外回报。

PoS的优点：在一定程度上缩短了共识达成的时间；不再需要大量消耗能源去挖矿。

PoS的缺点：还是需要挖矿，本质上没有解决商业应用的痛点；所有的确认都只是一个概率上的表达，而不是一个确定性的事情，理论上有可能存在其他攻击影响，例如以太坊的DAO攻击事件造成以太坊硬分叉，而ETC随之出现，事实上证明了此次硬分叉的失败。

3. 股份授权证明（DPOS）

BitShares（比特股）社区首先提出了股份授权证明（简称DPoS）机制，它与PoS的主要区别在于节点选举若干代理人，由代理人验证和记账，但其合规监管、性能、资源消耗和容错性与PoS相似。类似于董事会投票，持币者投出一定数量的节点，进行代理验证和记账。

DPoS的工作原理如下：每个股东按其持股比例拥有相应的影响力，51%股东投票的结果将是不可逆且有约束力的，其挑战是通过及时而高效的方法达到“51%批准”；

为了达到这个目标，每个股东可以将其投票授予一名代表。获票数最多的前100位代表按既定时间表轮流产生区块。每位代表分配到一个时间段来生产区块。

所有的代表将收到等同于一个平均水平的区块所含交易费的10%作为报酬。如果一个平均水平的区块用100股作为交易费，一位代表将获得一股作为报酬。

网络延迟有可能使某些代表没能及时广播他们的区块，而这将导致区块链分叉。然而，这不太可能发生，因为制造该区块的代表可以与制造该区块前后的区块的代表建立直接连接。建立这种与你之后的代表（也许也包括其后的那名代表）的直接连接是为了确保你能得到报酬。

DPoS的投票模式可以每30秒产生一个新区块，并且在正常的网络条件下，区块链分叉的可能性极其小，即使发生也可以在几分钟内得到解决。执行该模式的基本步骤如下：

1. 成为代表。成为一位代表，你必须在网络上注册你的公钥，并获得一个32位的特有标识符。该标识符会被每笔交易数据的“头部”引用。
2. 授权投票。每个钱包有一个参数设置窗口，在该窗口里用户可以选择一位或更多的代表，并将其分级。一经设定，用户所做的每笔交易将把选票从“输入代表”转移至“输出代

表”。一般情况下，用户不会创建专门以投票为目的的交易，因为那将耗费他们一笔交易费。但是在紧急情况下，某些用户可能觉得通过支付费用这一更积极的方式来改变他们的投票是值得的。

3. 保持代表忠诚。每个钱包将显示一个状态指示器，让用户知道他们的代表表现如何。如果他们错过了太多的区块，那么系统将会推荐用户更换一位新的代表。如果任何代表被发现签发了一个无效的区块，那么所有标准钱包将在每个钱包进行更多交易前要求选出一位新代表。
4. 抵抗攻击。在抵抗攻击上，前100位代表所获得的权利是相同的，即每位代表都有一项平等的投票权，因此，无法通过获得超过1%的选票而将权利集中到单一代表上。由于只有100位代表，不难想象一个攻击者可以对每位轮到其生产区块的代表依次进行拒绝服务攻击。幸运的是，由于每位代表的标识是其公钥而非IP地址，这种特定攻击的威胁很容易被减轻。这将使确定DDoS（分布式拒绝服务）攻击目标更为困难。而代表之间的潜在连接将使妨碍他们生产区块变得更为困难。

DPoS的优点：大幅缩小参与验证和记账节点的数量，可以达到秒级的共识验证。

DPoS的缺点：整个共识机制还是依赖于代币，而很多商业应用是不需要代币的。

4. 投注共识

投注共识是以太坊下一代的共识机制Casper（鬼马小精灵）引入的一个全新概念，属于PoS。Casper的共识是按区块达成的，而不像PoS那样按链达成。

为了防止验证人在不同的世界中提供不同的投注，我们还有一个简单严格的条款：如果你两次的投注序号一样，或者说你提交了一个无法让Casper依照合约处理的投注，你将失去所有保证金。从这一点我们可以看出，Casper与传统的PoS不同的是，Casper有惩罚机制，这样非法节点通过恶意攻击网络不仅得不到交易费，而且还面临着保证金被没收的风险。

Casper协议下的验证人需要完成出块和投注两个活动。具体如下：

出块是一个独立于其他所有时间而发生的过程，验证人收集交易，当轮到他们的出块时间时，他们就制造一个区块，并签名，然后发送到网络上。投注的过程更为复杂一些，目前Casper默认的验证人策略被设计为模仿传统的拜占庭容错共识：观察其他的验证人如何投注，取33%处的值，向0或1进一步移动。

而客户端确认当前状态的过程是这样的：一开始先下载所有的区块和投注，然后用上面的算法来形成自己的意见，但是不公布意见；它只是简单地按顺序在每个高度进行观察，如果一个区块的概率高于0.5就处理它，否则就跳过它。在处理所有的区块之后，所得到的状态就可以显示为区块链的“当前状态”。客户端还可以给出对于“最终确定”的主观看法：如果高度k之前的每个区块形成的意见高于99.999%或者低于0.001%，那么客户端可以认为前k个区块已经最终确定。

5. 瑞波共识机制（Ripple Consensus）

瑞波共识算法使一组节点能够基于特殊节点列表形成共识。初始特殊节点列表就像一个俱乐部，要接纳一个新成员，必须由该俱乐部51%的会员投票通过。共识遵循这些核心成员的“51%权利”，外部人员则没有影响力。由于该俱乐部由中心化开始，它将一直是中心化的，而如果它开始腐化，股东们什么也做不了。与比特币及Peercoin一样，瑞波系统将股东们与其投票权隔开，因此，它比其他系统更中心化。

6. Pool验证池

基于传统的分布式一致性技术以及数据验证机制，Pool（联营）验证池是目前行业内大范围使用的共识机制。它的优缺点如下：

优点：不需要代币也可以工作，在成熟的分布式一致性算法（Paxos、Raft）的基础上，实现秒级共识验证。

缺点：去中心化程度不如比特币，更适合多方参与的多中心商业模式。

7. 实用拜占庭容错

在分布式计算上，不同的计算机通过信息交换尝试达成共识，但有时候，系统中的协调计算机或者成员计算机可能因系统错误，而交换错误信息，以致影响最终的系统一致性。对于拜占庭将军问题，若根据错误计算机的数量，寻找可能的解决办法，这其实无法找到一个绝对的答案，只可以用来验证一个机制的有效程度。

而拜占庭将军问题的可能解决方法为：在N≥3F+1的情况下，一致性是可能实现的（N为计算机总数，F为有问题的计算机总数）。信息在计算机间互相交换后，各计算机列出所有得到的信息，以大多数的结果作为解决办法。

最早由卡斯特罗和利斯科夫在1999年提出的使用拜占庭容错（PBFT）是第一个得到广泛应用的拜占庭算法。只要系统中有2/3的节点是正常工作的，就可以保证一致性。

使用拜占庭容错算法的总体过程如下：客户端向主节点发送请求调用服务操作，如“<REQUEST,o,t,c>”，这里客户端c请求执行操作o，时间戳t用来保证客户端请求只会执行一次。每个由副本节点发给客户端的消息都包含了当前的视图编号，使得客户端能够追踪视图编号，从而进一步推算出当前主节点的编号。客户端通过点对点消息向它自己认为的主节点发送请求，然后主节点自动将该请求向所有备份节点进行广播。

视图编号是连续编号的整数，主节点由公式p=v mod |R|计算得到，这里v是视图编号，p是副本编号，|R|是副本集合的个数。

副本发给客户单的响应为“<REPLY,v,t,c,i,r>”，v是视图编号，t是时间戳，i是副本的编号，r是请求执行的结果。

主节点通过广播将请求发送给其他副本，然后就开始执行三个阶段的任务。

1. 预准备阶段。主节点分配一个序列号n给收到的请求，然后向所有备份节点群发预准备消息，预准备消息格式为“ < < PRE-PREPARE, v, n, d > , m > ”，这里v是视图编号，m是客户端发送的请求消息，d是请求消息m的摘要。
2. 准备阶段。如果备份节点i接受了预准备消息，则进入准备阶段。在准备的同时，该节点向所有副本节点发送准备消息“ < PREPARE, v, n, d, i > ”，并且将预准备消息和准备消息写入自己的消息日志。
3. 确认阶段。当“(m, v, n, i)”条件为真的时候，副本i将“ < COMMIT, v, n, D(m), i > ”向其他副本节点广播，于是就进入了确认阶段。所有副本都执行请求并将结果发回客户端。客户端需要等待不同副本节点发回相同的结果，作为整个操作的最终结果。

如果客户端没有在有限时间内收到回复，请求将向所有副本节点进行广播；
如果该请求已经在副本节点处理过了，副本就向客户端重发一遍执行结果；
如果请求没有在副本节点处理过，该副本节点将把请求转发给主节点；
如果主节点没有将该请求进行广播，那么就认为主节点失效；
如果有足够多的副本节点认为主节点失效，则会触发一次视图变更。

图2-85展示了在没有发生主节点失效的情况下算法的正常执行流程，其中副本0是主节点，副本3是失效节点，而c是客户端。

使用拜占庭容错机制是一种采用“许可投票、少数服从多数”来选举领导者并进行记账的共识机制，该共识机制允许拜占庭容错，允许强监督节点参与，具备权限分级能力，性能更高，耗能更低，而且每轮记账都会由全网节点共同选举领导者，允许33%的节点作恶，容错率为33%。

由于特别适合联盟链的应用场景，实用拜占庭容错机制及其改进算法为目前使用最多的联盟链共识算法，其改进算法为目前使用最多的联盟链共识算法，其改进算法在以下方面进行了调整：修改底层网络拓扑的要求，使用P2P网络；可以动态地调整节点数量；减少协议使用的消息数量。

8. 授权拜占庭容错

2016年4月，小蚁公司发布共识算法白皮书，描述了一种通用共识机制——授权拜占庭容错，提出了一种改进的拜占庭容错算法，使其能够适用于区块链系统。授权拜占庭容错算法在使用拜占庭容错算法的基础上，进行了以下改进：

1. 将C/S架构的请求响应模式改进为适合P2P网络的对等节点模式；
2. 将静态的共识参与节点改进为可动态进入、退出的共识参与节点；
3. 为共识参与节点的产生设计了一套基于持有权益比例的投票机制，通过投票决定共识参

与节点（记账节点）；

4. 在区块链中引入数字证书，解决了投票中对记账节点真实身份的认证问题。

授权拜占庭容错机制的优点：专业化的记账人；可以容忍任何类型的错误；记账由多人协同完成；每一个区块都有最终性，不会分叉；算法的可靠性有严格的数字证明。

授权拜占庭容错机制的缺点：当1/3及以上的记账人停止工作后，系统将无法提供服务；当1/3及以上的记账人联合作恶，且其他所有的记账人被恰好分割为两个网络孤岛时，恶意记账人可以使系统出现分叉，但是会留下密码学证据。

总而言之，授权拜占庭容错机制最核心的一点，就是最大限度地确保系统的最终性，使区块链能够适用于真正的金融应用场景。

9. Paxos算法

这是一种传统的分布式一致性算法，是一种基于选举领导者的共识机制。领导者节点拥有绝对权限，并允许强监督节点参与，其性能高，资源消耗低。所有节点一般有线下准入机制，但选举过程中不允许有作恶节点，不具备容错性。