以太坊（Ethereum ）:下一代智能合约和去中心化应用平台

翻译：巨蟹 、少平

译者注：中文读者可以到以太坊爱好者社区获取最新的以太坊信息。

当中本聪在 2009 年 1 月启动比特币区块链时，他同时向世界引入了两种未经测试的革命性的新概念。第一种就是比特币（bitcoin），一种去中心化的点对点的网上货币，在没有任何资产担保、内在价值或者中心发行者的情况下维持着价值。到目前为止，比特币已经吸引了大量的公众注意力, 就政治方面而言它是一种没有中央银行的货币并且有着剧烈的价格波动。然而，中本聪的伟大试验还有与比特币同等重要的一部分：基于工作量证明的区块链概念使得人们可以就交易顺序达成共识。作为应用的比特币可以被描述为一个先申请（first-to-file）系统：如果某人有 50BTC 并且同时向 A 和 B 发送这 50BTC，只有被首先被确认的交易才会生效。没有固有方法可以决定两笔交易哪一笔先到，这个问题阻碍了去中心化数字货币的发展许多年。中本聪的区块链是第一个可靠的去中心化解决办法。现在，开发者们的注意力开始迅速地转向比特币技术的第二部分，区块链怎样应用于货币以外的领域。

常被提及的应用包括使用链上数字资产来代表定制货币和金融工具（彩色币），某种基础物理设备的所有权（智能资产），如域名一样的没有可替代性的资产（域名币）以及如去中心化交易所，金融衍生品，点到点赌博和链上身份和信誉系统等更高级的应用。另一个常被问询的重要领域是"智能合约"- 根据事先任意制订的规则来自动转移数字资产的系统。例如，一个人可能有一个存储合约，形式为"A 可以每天最多提现 X 个币，B 每天最多 Y 个，A 和 B 一起可以随意提取，A 可以停掉 B 的提现权"。这种合约的符合逻辑的扩展就是去中心化自治组织（DAOs）-长期的包含一个组织的资产并把组织的规则编码的智能合约。以太坊的目标就是提供一个带有内置的成熟的图灵完备语言的区块链，用这种语言可以创建合约来编码任意状态转换功能，用户只要简单地用几行代码来实现逻辑，就能够创建以上提及的所有系统以及许多我们还想象不到的的其它系统。

目录

        - 金融衍生品
        - 身份和信誉系统
        - 去中心化文件存储
        - 去中心化自治组织
        - 进一步的应用
    - 杂项和关注
        - 改进版幽灵协议的实施
        - 费用
        - 计算和图灵完备
        - 货币和发行
        - 挖矿的中心化
        - 扩展性
    - 综述：去中心化应用
    - 结论
    - 注解和进阶阅读

历史

去中心化的数字货币概念，正如财产登记这样的替代应用一样，早在几十年以前就被提出来了。1980 和 1990 年代的匿名电子现金协议，大部分是以乔姆盲签技术（Chaumian blinding）为基础的。这些电子现金协议提供具有高度隐私性的货币，但是这些协议都没有流行起来，因为它们都依赖于一个中心化的中介机构。1998 年，戴伟（Wei Dai）的 b-money 首次引入了通过解决计算难题和去中心化共识创造货币的思想，但是该建议并未给出如何实现去中心化共识的具体方法。2005 年，芬尼（Hal Finney）引入了"可重复使用的工作量证明机制"（reusable proofs of work）概念，它同时使用 b-money 的思想和 Adam Back 提出的计算困难的哈希现金（Hashcash）难题来创造密码学货币。但是，这种概念再次迷失于理想化，因为它依赖于可信任的计算作为后端。

因为货币是一个先申请应用，交易的顺序至关重要，所以去中心化的货币需要找到实现去中心化共识的方法。比特币以前的所有电子货币协议所遇到的主要障碍是，尽管对如何创建安全的拜占庭问题容错（Byzantine-fault-tolerant）多方共识系统的研究已经历时多年，但是上述协议只解决了问题的一半。这些协议假设系统的所有参与者是已知的，并产生如"如果有 N 方参与到系统中，那么系统可以容忍 N/4 的恶意参与者"这样形式的安全边界。然而这个假设的问题在于，在匿名的情况下，系统设置的安全边界容易遭受女巫攻击，因为一个攻击者可以在一台服务器或者僵尸网络上创建数以千计的节点，从而单方面确保拥有多数份额。

中本聪的创新是引入这样一个理念：将一个非常简单的基于节点的去中心化共识协议与工作量证明机制结合在一起。节点通过工作量证明机制获得参与到系统的权利，每十分钟将交易打包到"区块"中，从而创建出不断增长的区块链。拥有大量算力的节点有更大的影响力，但获得比整个网络更多的算力比创建一百万个节点困难得多。尽管比特币区块链模型非常简陋，但是实践证明它已经足够好用了，在未来五年，它将成为全世界两百个以上的货币和协议的基石。

作为状态转换系统的比特币

从技术角度讲，比特币账本可以被认为是一个状态转换系统，该系统包括所有现存的比特币所有权状态和“状态转换函数”。状态转换函数以当前状态和交易为输入，输出新的状态。例如，在标准的银行系统中，状态就是一个资产负债表，一个从 A 账户向 B 账户转账 X 美元的请求是一笔交易，状态转换函数将从 A 账户中减去 X 美元，向 B 账户增加 X 美元。如果 A 账户的余额小于 X 美元，状态转换函数就会返回错误提示。所以我们可以如下定义状态转换函数：

```
APPLY(S,TX) > S' or ERROR
```

在上面提到的银行系统中，状态转换函数如下：

```
APPLY({ Alice: $50, Bob: $50 },"send $20 from Alice to Bob") = { Alice: $30,Bob: $70 }
```

但是：

```
APPLY({ Alice: $50, Bob: $50 },"send $70 from Alice to Bob") = ERROR
```

比特币系统的“状态”是所有已经被挖出的、没有花费的比特币（技术上称为“未花费的交易输出，unspent transaction outputs 或 UTXO”）的集合。每个 UTXO 都有一个面值和所有者（由 20 个字节的本质上是密码学公钥的地址所定义[1]）。一笔交易包括一个或多个输入和一个或多个输出。每个输入包含一个对现有 UTXO 的引用和由与所有者地址相对应的私钥创建的密码学签名。每个输出包含一个新的加入到状态中的 UTXO。

在比特币系统中，状态转换函数 `APPLY(S,TX)->S’`大体上可以如下定义：

1. 交易的每个输入：
    - 如果引用的 UTXO 不存在于现在的状态中（s），返回错误提示
    - 如果签名与 UTXO 所有者的签名不一致，返回错误提示
2. 如果所有的 UTXO 输入面值总额小于所有的 UTXO 输出面值总额，返回错误提示
3. 返回新状态 s’,新状态 s 中移除了所有的输入 UTXO，增加了所有的输出 UTXO。

第一步的第一部分防止交易的发送者花费不存在的比特币，第二部分防止交易的发送者花费其他人的比特币。第二步确保价值守恒。比特币的支付协议如下。假设 Alice 想给 Bob 发送 11.7BTC。事实上，Alice 不可能正好有 11.7BTC。假设，她能得到的最小数额比特币的方式是：6+4+2=12。所以，她可以创建一笔有 3 个输入，2 个输出的交易。第一个输出的面值是 11.7BTC，所有者是 Bob（Bob 的比特币地址），第二个输出的面值是 0.3BTC，所有者是 Alice 自己，也就是找零。

挖矿

如果我们拥有可信任的中心化服务机构，状态转换系统可以很容易地实现，可以简单地将上述功能准确编码。然而，我们想把比特币系统建成为去中心化的货币系统，为了确保每个人都同意交易的顺序，我们需要将状态转换系统与一个共识系统结合起来。比特币的去中心化共识进程要求网络中的节点不断尝试将交易打包成“区块”。网络被设计为大约每十

分钟产生一个区块，每个区块包含一个时间戳、一个随机数、一个对上一个区块的引用（即哈希）和上一区块生成以来发生的所有交易列表。这样随着时间流逝就创建出了一个持续增长的区块链，它不断地更新，从而能够代表比特币账本的最新状态。

依照这个范式，检查一个区块是否有效的算法如下：

1. 检查区块引用的上一个区块是否存在且有效。
2. 检查区块的时间戳是否晚于以前的区块的时间戳，而且早于未来 2 小时[2]。
3. 检查区块的工作量证明是否有效。
4. 将上一个区块的最终状态赋于 `S[0]`。
5. 假设 TX 是区块的交易列表，包含 n 笔交易。对于属于 0……n-1 的所有 i,进行状态转换 `S[i+1] = APPLY(S[i],TX[i])`。如果任何一笔交易 i 在状态转换中出错，退出程序，返回错误。
6. 返回正确，状态 `S[n]`是这一区块的最终状态。

本质上，区块中的每笔交易必须提供一个正确的状态转换，要注意的是，“状态”并不是编码到区块的。它纯粹只是被校验节点记住的抽象概念，对于任意区块都可以从创世状态开始，按顺序加上每一个区块的每一笔交易，（妥妥地）计算出当前的状态。另外，需要注意矿工将交易收录进区块的顺序。如果一个区块中有 A、B 两笔交易，B 花费的是 A 创建的 UTXO，如果 A 在 B 以前，这个区块是有效的，否则，这个区块是无效的。

区块验证算法的有趣部分是“工作量证明”概念：对每个区块进行 SHA256 哈希处理，将得到的哈希视为长度为 256 比特的数值，该数值必须小于不断动态调整的目标数值，本书写作时目标数值大约是 2^190。工作量证明的目的是使区块的创建变得困难，从而阻止女巫攻击者恶意重新生成区块链。因为 SHA256 是完全不可预测的伪随机函数，创建有效区块的唯一方法就是简单地不断试错，不断地增加随机数的数值，查看新的哈希数值是否小于目标数值。如果当前的目标数值是 2^192，就意味着平均需要尝试 2^64 次才能生成有效的区块。一般而言，比特币网络每隔 2016 个区块重新设定目标数值，保证平均每十分钟生成一个区块。为了对矿工的计算工作进行奖励，每一个成功生成区块的矿工有权在区块中包含一笔凭空发给他们自己 25BTC 的交易。另外，如果交易的输入大于输出，差额部分就作为“交易费用”付给矿工。顺便提一下，对矿工的奖励是比特币发行的唯一机制，创世状态中并没有比特币。

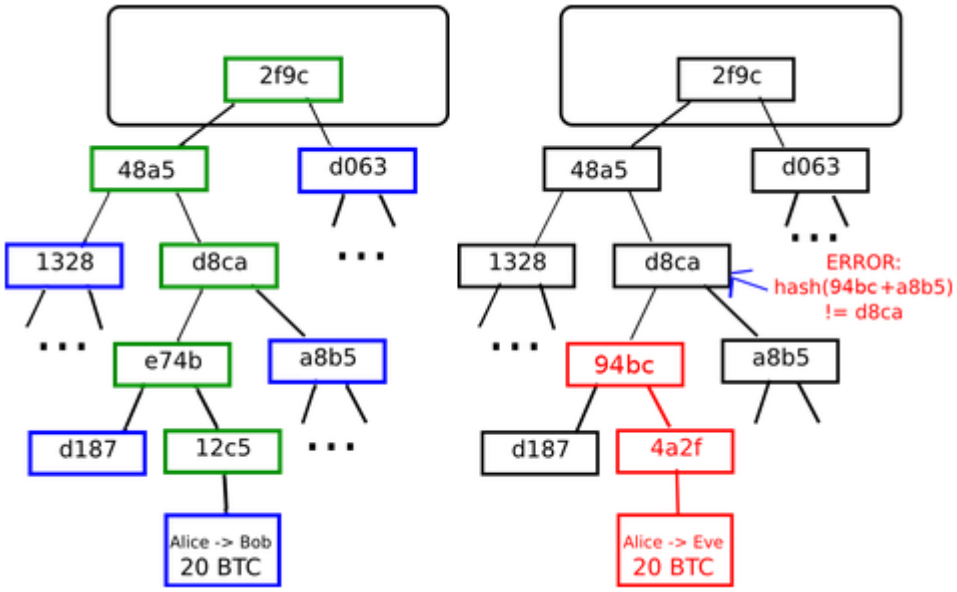
为了更好地理解挖矿的目的，让我们分析比特币网络出现恶意攻击者时会发生什么。因为比特币的密码学基础是非常安全的，所以攻击者会选择攻击没有被密码学直接保护的部分：交易顺序。攻击者的策略非常简单：

1. 向卖家发送 100BTC 购买商品（尤其是无需邮寄的电子商品）。
2. 等待直至商品发出。
3. 创建另一笔交易，将相同的 100BTC 发送给自己的账户。
4. 使比特币网络相信发送给自己账户的交易是最先发出的。

一旦步骤（1）发生，几分钟后矿工将把这笔交易打包到区块，假设是第 270000 个区块。大约一个小时以后，在此区块后面将会有五个区块，每个区块间接地指向这笔交易，从而确认这笔交易。这时卖家收到货款，并向买家发货。因为我们假设这是数字商品，攻击者可以即时收到货。现在，攻击者创建另一笔交易，将相同的 100BTC 发送到自己的账户。

如果攻击者只是向全网广播这一消息，这一笔交易不会被处理。矿工会运行状态转换函数 APPLY(S,TX)，发现这笔交易将花费已经不在状态中的 UTXO。所以，攻击者会对区块链进行分叉，将第 269999 个区块作为父区块重新生成第 270000 个区块，在此区块中用新的交易取代旧的交易。因为区块数据是不同的，这要求重新进行工作量证明。另外，因为攻击者生成的新的第 270000 个区块有不同的哈希，所以原来的第 270001 到第 270005 的区块不指向它，因此原有的区块链和攻击者的新区块是完全分离的。在发生区块链分叉时，区块链长的分支被认为是诚实的区块链，合法的的矿工将会沿着原有的第 270005 区块后挖矿，只有攻击者一人在新的第 270000 区块后挖矿。攻击者为了使得他的区块链最长，他需要拥有比除了他以外的全网更多的算力来追赶（即 51%攻击）。

默克尔树



左：仅提供默克尔树（Merkle tree）上的少量节点已经足够给出分支的合法证明。
右：任何对于默克尔树的任何部分进行改变的尝试都会最终导致链上某处的不一致。

比特币系统的一个重要的可扩展特性是：它的区块存储在多层次的数据结构中。一个区块的哈希实际上只是区块头的哈希，区块头是包含时间戳、随机数、上个区块哈希和存储了所有的区块交易的默克尔树的根哈希的长度大约为 200 字节的一段数据。

默克尔树是一种二叉树，由一组叶节点、一组中间节点和一个根节点构成。最下面的大量的叶节点包含基础数据，每个中间节点是它的两个子节点的哈希，根节点也是由它的两个子节点的哈希，代表了默克尔树的顶部。默克尔树的目的是允许区块的数据可以零散地传送：节点可以从一个源下载区块头，从另外的源下载与其有关的树的其它部分，而依然能够确认所有的数据都是正确的。之所以如此是因为哈希向上的扩散：如果一个恶意用户尝试在树的下部加入一个伪造的交易，所引起的改动将导致树的上层节点的改动，以及更上层节点的改动，最终导致根节点的改动以及区块哈希的改动，这样协议就会将其记录为一个完全不同的区块（几乎可以肯定是带着不正确的工作量证明的）。

默克尔树协议对比特币的长期持续性可以说是至关重要的。在 2014 年 4 月，比特币网络中的一个全节点-存储和处理所有区块的全部数据的节点-需要占用 15GB 的内存空间，而且还以每个月超过 1GB 的速度增长。目前，这一存储空间对台式计算机来说尚可接受，但

是手机已经负载不了如此巨大的数据了。未来只有商业机构和爱好者才会充当完整节点。简化支付确认（SPV)协议允许另一种节点存在，这样的节点被成为“轻节点”，它下载区块头，使用区块头确认工作量证明，然后只下载与其交易相关的默克尔树“分支”。这使得轻节点只要下载整个区块链的一小部分就可以安全地确定任何一笔比特币交易的状态和账户的当前余额。

其它的区块链应用

将区块链的思想应用到其它领域的想法早就出现了。在 2005 年，尼克萨博提出了“用所有权为财产冠名”的概念，文中描述了复制数据库技术的发展如何使基于区块链的系统可以应用于登记土地所有权，创建包括例如房产权、违法侵占和乔治亚州土地税等概念的详细框架。然而，不幸的是在那时还没有实用的复制数据库系统，所以这个协议被没有被付诸实践。不过，自 2009 年比特币系统的去中心化共识开发成功以来，许多区块链的其它应用开始快速出现。

- **域名币（namecoin）**- 创建于 2010 年，被称为去中心化的名称注册数据库。像 Tor、Bitcoin 和 BitMessage 这样的去中心化协议，需要一些确认账户的方法，这样其他人才能够与用户进行交互。但是，在所有的现存的解决方案中仅有的可用的身份标识是象 `1LW79wp5ZBqaHW1jL5TciBCrhQYtHagUWy` 这样的伪随机哈希。理想的情况下，人们希望拥有一个带有象“george”这样的名称的账户。然而，问题是如果有人可以创建“george”账户，那么其他人同样也可以创建“george”账户来假扮。唯一的解决方法是先申请原则（first-to-file），只有第一个注册者可以成功注册，第二个不能再次注册同一个账户。这一问题就可以利用比特币的共识协议。域名币是利用区块链实现名称注册系统的最早的、最成功的系统。
- **彩色币（Colored coins）**- 彩色币的目的是为人们在比特币区块链上创建自己的数字货币，或者，在更重要的一般意义上的货币 – 数字令牌提供服务。依照彩色币协议，人们可以通过为某一特别的比特币 UTXO 指定颜色，发行新的货币。该协议递归地将其它 UTXO 定义为与交易输入 UTXO 相同的颜色。这就允许用户保持只包含某一特定颜色的 UTXO，发送这些 UTXO 就像发送普通的比特币一样，通过回溯全部的区块链判断收到的 UTXO 颜色。
- **元币（Metacoins）**- 元币的理念是在比特币区块链上创建新的协议，利用比特币的交易保存元币的交易，但是采用了不同的状态转换函数 APPLY'。因为元币协议不能阻止比特币区块链上的无效的元币交易，所以增加一个规则如果 APPLY'(S,TX) 返回错误，这一协议将默认 APPLY'(S,TX) = S。这为创建任意的、先进的不能在比特币系统中实现的密码学货币协议提供了一个简单的解决方法，而且开发成本非常低，因为挖矿和网络的问题已经由比特币协议处理好了。

因此，一般而言，建立共识协议有两种方法：建立一个独立的网络和在比特币网络上建立协议。虽然像域名币这样的应用使用第一种方法已经获得了成功，但是该方法的实施非常困难，因为每一个应用需要创建独立的区块链和建立、测试所有状态转换和网络代码。另外，我们预测去中心化共识技术的应用将会服从幂律分布，大多数的应用太小不足以保证自由区块链的安全，我们还注意到大量的去中心化应用，尤其是去中心化自治组织，需要进行应用之间的交互。

另一方面，基于比特币的方法存在缺点，它没有继承比特币可以进行简化确认支付（SPV)的特性。比特币可以实现简化确认支付，因为比特币可以将区块链深度作为有效性确认代

理。在某一点上，一旦一笔交易的祖先们距离现在足够远时，就可以认为它们是合法状态的一部分。与之相反，基于比特币区块链的元币协议不能强迫区块链不包括不符合元币协议的交易。因此，安全的元币协议的简化支付确认需要后向扫描所有的区块，直到区块链的初始点，以确认某一交易是否有效。目前，所有基于比特币的元币协议的"轻"实施都依赖可信任的服务器提供数据，这对主要目的之一是消除信任需要的密码学货币而言，只是一个相当次优的结果。

脚本

即使不对比特币协议进行扩展，它也能在一定程度上实现"智能合约"。比特币的 UTXO 可以被不只被一个公钥拥有，也可以被用基于堆栈的编程语言所编写的更加复杂的脚本所拥有。在这一模式下，花费这样的 UTXO，必须提供满足脚本的数据。事实上，基本的公钥所有权机制也是通过脚本实现的：脚本将椭圆曲线签名作为输入，验证交易和拥有这一 UTXO 的地址，如果验证成功，返回 1，否则返回 0。更加复杂的脚本用于其它不同的应用情况。例如，人们可以创建要求集齐三把私钥中的两把才能进行交易确认的脚本（多重签名），对公司账户、储蓄账户和某些商业代理来说，这种脚本是非常有用的。脚本也能用来对解决计算问题的用户发送奖励。人们甚至可以创建这样的脚本"如果你能够提供你已经发送一定数额的的狗币给我的简化确认支付证明，这一比特币 UTXO 就是你的了"，本质上，比特币系统允许不同的密码学货币进行去中心化的兑换。

然而，比特币系统的脚本语言存在一些严重的限制：

- **缺少图灵完备性** – 这就是说，尽管比特币脚本语言可以支持多种计算，但是它不能支持所有的计算。最主要的缺失是循环语句。不支持循环语句的目的是避免交易确认时出现无限循环。理论上，对于脚本程序员来说，这是可以克服的障碍，因为任何循环都可以用多次重复 if 语句的方式来模拟，但是这样做会导致脚本空间利用上的低效率，例如，实施一个替代的椭圆曲线签名算法可能将需要 256 次重复的乘法，而每次都需要单独编码。
- **价值盲（Value-blindness）**。UTXO 脚本不能为账户的取款额度提供精细的的控制。例如，预言机合约（oracle contract）的一个强大应用是对冲合约，A 和 B 各自向对冲合约中发送价值 1000 美元的比特币，30 天以后，脚本向 A 发送价值 1000 美元的比特币，向 B 发送剩余的比特币。虽然实现对冲合约需要一个预言机（oracle）决定一比特币值多少美元，但是与现在完全中心化的解决方案相比，这一机制已经在减少信任和基础设施方面有了巨大的进步。然而，因为 UTXO 是不可分割的，为实现此合约，唯一的方法是非常低效地采用许多有不同面值的 UTXO（例如对应于最大为 30 的每个 k，有一个 2^k 的 UTXO)并使预言机挑出正确的 UTXO 发送给 A 和 B。
- **缺少状态** – UTXO 只能是已花费或者未花费状态，这就没有给需要任何其它内部状态的多阶段合约或者脚本留出生存空间。这使得实现多阶段期权合约、去中心化的交换要约或者两阶段加密承诺协议（对确保计算奖励非常必要）非常困难。这也意味着 UTXO 只能用于建立简单的、一次性的合约，而不是例如去中心化组织这样的有着更加复杂的状态的合约，使得元协议难以实现。二元状态与价值盲结合在一起意味着另一个重要的应用-取款限额-是不可能实现的。

- **区块链盲（Blockchain-blindness）**- UTXO 看不到区块链的数据，例如随机数和上一个区块的哈希。这一缺陷剥夺了脚本语言所拥有的基于随机性的潜在价值，严重地限制了博彩等其它领域应用。

我们已经考察了在密码学货币上建立高级应用的三种方法：建立一个新的区块链，在比特币区块链上使用脚本，在比特币区块链上建立元币协议。建立新区块链的方法可以自由地实现任意的特性，成本是开发时间和培育努力。使用脚本的方法非常容易实现和标准化，但是它的能力有限。元币协议尽管非常容易实现，但是存在扩展性差的缺陷。在以太坊系统中，我们的目的是建立一个能够同时具有这三种模式的所有优势的通用框架。

以太坊

以太坊的目的是基于脚本、竞争币和链上元协议（on-chain meta-protocol）概念进行整合和提高，使得开发者能够创建任意的基于共识的、可扩展的、标准化的、特性完备的、易于开发的和协同的应用。以太坊通过建立终极的抽象的基础层-内置有图灵完备编程语言的区块链-使得任何人都能够创建合约和去中心化应用并在其中设立他们自由定义的所有权规则、交易方式和状态转换函数。域名币的主体框架只需要两行代码就可以实现，诸如货币和信誉系统等其它协议只需要不到二十行代码就可以实现。智能合约-包含价值而且只有满足某些条件才能打开的加密箱子-也能在我们的平台上创建，并且因为图灵完备性、价值知晓（value-awareness）、区块链知晓（blockchain-awareness）和多状态所增加的力量而比比特币脚本所能提供的智能合约强大得多。

以太坊账户

在以太坊系统中，状态是由被称为"账户"（每个账户由一个 20 字节的地址）的对象和在两个账户之间转移价值和信息的状态转换构成的。以太坊的账户包含四个部分：

- 随机数，用于确定每笔交易只能被处理一次的计数器
- 账户目前的以太币余额
- 账户的合约代码，如果有的话
- 账户的存储（默认为空）

以太币（Ether）是以太坊内部的主要加密燃料，用于支付交易费用。一般而言，以太坊有两种类型的账户：外部所有的账户（由私钥控制的）和合约账户（由合约代码控制）。外部所有的账户没有代码，人们可以通过创建和签名一笔交易从一个外部账户发送消息。每当合约账户收到一条消息，合约内部的代码就会被激活，允许它对内部存储进行读取和写入，和发送其它消息或者创建合约。

消息和交易

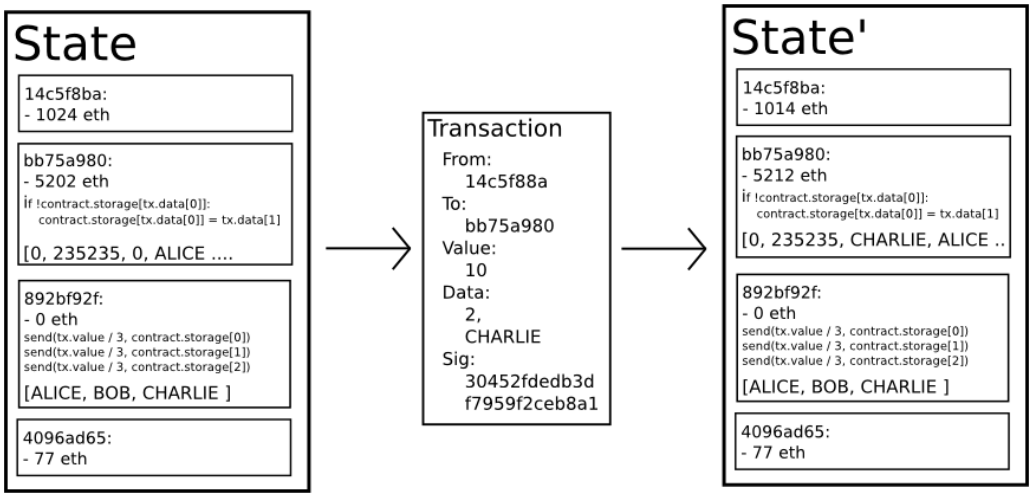
以太坊的消息在某种程度上类似于比特币的交易，但是两者之间存在三点重要的不同。第一，以太坊的消息可以由外部实体或者合约创建，然而比特币的交易只能从外部创建。第二，以太坊消息可以选择包含数据。第三，如果以太坊消息的接受者是合约账户，可以选择进行回应，这意味着以太坊消息也包含函数概念。

以太坊中"交易"是指存储从外部账户发出的消息的签名数据包。交易包含消息的接收者、用于确认发送者的签名、以太币账户余额、要发送的数据和两个被称为 STARTGAS 和

GASPRICE 的数值。为了防止代码的指数型爆炸和无限循环，每笔交易需要对执行代码所引发的计算步骤-包括初始消息和所有执行中引发的消息-做出限制。STARTGAS 就是限制，GASPRICE 是每一计算步骤需要支付矿工的费用。如果执行交易的过程中，“用完了瓦斯”，所有的状态改变恢复原状态，但是已经支付的交易费用不可收回了。如果执行交易中止时还剩余瓦斯，那么这些瓦斯将退还给发送者。创建合约有单独的交易类型和相应的消息类型；合约的地址是基于账号随机数和交易数据的哈希计算出来的。

消息机制的一个重要后果是以太坊的“头等公民”财产-合约与外部账户拥有同样权利，包括发送消息和创建其它合约的权利。这使得合约可以同时充当多个不同的角色，例如，用户可以使去中心化组织（一个合约）的一个成员成为一个中介账户（另一个合约），为一个偏执的使用定制的基于量子证明的兰波特签名（第三个合约）的个人和一个自身使用由五个私钥保证安全的账户（第四个合约）的共同签名实体提供居间服务。以太坊平台的强大之处在于去中心化的组织和代理合约不需要关心合约的每一参与方是什么类型的账户。

以太坊状态转换函数



以太坊的状态转换函数：`APPLY(S,TX) -> S'`，可以定义如下：

1. 检查交易的格式是否正确（即有正确数值）、签名是否有效和随机数是否与发送者账户的随机数匹配。如否，返回错误。
2. 计算交易费用:`fee=STARTGAS * GASPRICE`，并从签名中确定发送者的地址。从发送者的账户中减去交易费用和增加发送者的随机数。如果账户余额不足，返回错误。
3. 设定初值 `GAS = STARTGAS`，并根据交易中的字节数减去一定量的瓦斯值。
4. 从发送者的账户转移价值到接收者账户。如果接收账户还不存在，创建此账户。如果接收账户是一个合约，运行合约的代码，直到代码运行结束或者瓦斯用完。
5. 如果因为发送者账户没有足够的钱或者代码执行耗尽瓦斯导致价值转移失败，恢复原来的状态，但是还需要支付交易费用，交易费用加至矿工账户。
6. 否则，将所有剩余的瓦斯归还给发送者，消耗掉的瓦斯作为交易费用发送给矿工。例如，假设合约的代码如下：

```
if !self.storage[calldataload(0)]:
    self.storage[calldataload(0)] = calldataload(32)
```

需要注意的是，在现实中合约代码是用底层以太坊虚拟机（EVM）代码写成的。上面的合约是用我们的高级语言 Serpent 语言写成的，它可以被编译成 EVM 代码。假设合约存储器开始时是空的，一个值为 10 以太，瓦斯为 2000，瓦斯价格为 0.001 以太并且 64 字节数据，第一个三十二字节的块代表号码 2 和第二个代表词 CHARLIE。的交易发送后，状态转换函数的处理过程如下：

1. 检查交易是否有效、格式是否正确。
2. 检查交易发送者至少有 2000*0.001=2 个以太币。如果有，从发送者账户中减去 2 个以太币。
3. 初始设定 gas=2000,假设交易长为 170 字节，每字节的费用是 5，减去 850，所以还剩 1150。
4. 从发送者账户减去 10 个以太币，为合约账户增加 10 个以太币。
5. 运行代码。在这个合约中，运行代码很简单：它检查合约存储器索引为 2 处是否已使用，注意到它未被使用，然后将其值置为 CHARLIE。假设这消耗了 187 单位的瓦斯，于是剩余的瓦斯为 1150 - 187 = 963。 6．向发送者的账户增加 963*0.001=0.963 个以太币，返回最终状态。 如果没有合约接收交易，那么所有的交易费用就等于 GASPRICE 乘以交易的字节长度，交易的数据就与交易费用无关了。另外，需要注意的是，合约发起的消息可以对它们产生的计算分配瓦斯限额，如果子计算的瓦斯用完了，它只恢复到消息发出时的状态。因此，就像交易一样，合约也可以通过对它产生的子计算设置严格的限制，保护它们的计算资源。

代码执行

以太坊合约的代码是使用低级的基于堆栈的字节码的语言写成的，被称为“以太坊虚拟机代码”或者“EVM 代码”。代码由一系列字节构成，每一个字节代表一种操作。一般而言，代码执行是无限循环，程序计数器每增加一（初始值为零）就执行一次操作，直到代码执行完毕或者遇到错误，STOP 或者 RETURN 指令。操作可以访问三种存储数据的空间：

- **堆栈**，一种后进先出的数据存储，32 字节的数值可以入栈，出栈。
- **内存**，可无限扩展的字节队列。
- **合约的长期存储**，一个秘钥/数值的存储，其中秘钥和数值都是 32 字节大小，与计算结束即重置的堆栈和内存不同，存储内容将长期保持。

代码可以象访问区块头数据一样访问数值，发送者和接受到的消息中的数据，代码还可以返回数据的字节队列作为输出。

EVM 代码的正式执行模型令人惊讶地简单。当以太坊虚拟机运行时，它的完整的计算状态可以由元组(block_state, transaction, message, code, memory, stack, pc, gas)来定义，这里 block_state 是包含所有账户余额和存储的全局状态。每轮执行时，通过调出代码的第 pc（程序计数器）个字节，当前指令被找到，每个指令都有定义自己如何影响元组。例如，ADD 将两个元素出栈并将它们的和入栈，将 gas（瓦斯）减一并将 pc 加一，SSTORE 将顶部的两个元素出栈并将第二个元素插入到由第一个元素定义的合约存储位置，

同样减少最多 200 的 gas 值并将 pc 加一，虽然有许多方法通过即时编译去优化以太坊，但以太坊的基础性的实施可以用几百行代码实现。

区块链和挖矿

虽然有一些不同，但以太坊的区块链在很多方面类似于比特币区块链。它们的区块链架构的不同在于，以太坊区块不仅包含交易记录和最近的状态，还包含区块序号和难度值。以太坊中的区块确认算法如下：

1. 检查区块引用的上一个区块是否存在和有效。
2. 检查区块的时间戳是否比引用的上一个区块大，而且小于 15 分钟。
3. 检查区块序号、难度值、 交易根，叔根和瓦斯限额（许多以太坊特有的底层概念）是否有效。
4. 检查区块的工作量证明是否有效。
5. 将 `S[0]`赋值为上一个区块的 `STATE_ROOT`。
6. 将 `TX` 赋值为区块的交易列表，一共有 `n` 笔交易。对于属于 `0……n-1` 的 `i`，进行状态转换 `S[i+1] = APPLY(S[i],TX[i])`。如果任何一个转换发生错误，或者程序执行到此处所花费的瓦斯（gas）超过了 `GASLIMIT`，返回错误。
7. 用 `S[n]`给 `S_FINAL` 赋值, 向矿工支付区块奖励。
8. 检查 `S-FINAL` 是否与 `STATE_ROOT` 相同。如果相同，区块是有效的。否则，区块是无效的。

这一确认方法乍看起来似乎效率很低，因为它需要存储每个区块的所有状态，但是事实上以太坊的确认效率可以与比特币相提并论。原因是状态存储在树结构中（tree structure），每增加一个区块只需要改变树结构的一小部分。因此，一般而言，两个相邻的区块的树结构的大部分应该是相同的，因此存储一次数据，可以利用指针（即子树哈希）引用两次。一种被称为“帕特里夏树”（“Patricia Tree”）的树结构可以实现这一点，其中包括了对默克尔树概念的修改，不仅允许改变节点，而且还可以插入和删除节点。另外，因为所有的状态信息是最后一个区块的一部分，所以没有必要存储全部的区块历史-这一方法如果能够可以应用到比特币系统中，经计算可以对存储空间有 10-20 倍的节省。

应用

一般来讲，以太坊之上有三种应用。第一类是金融应用，为用户提供更强大的用他们的钱管理和参与合约的方法。包括子货币，金融衍生品，对冲合约，储蓄钱包，遗嘱，甚至一些种类的全面的雇佣合约。第二类是半金融应用，这里有钱的存在但也有很重的非金钱的方面，一个完美的例子是为解决计算问题而设的自我强制悬赏。最后，还有在线投票和去中心化治理这样的完全的非金融应用。

令牌系统

链上令牌系统有很多应用，从代表如美元或黄金等资产的子货币到公司股票，单独的令牌代表智能资产，安全的不可伪造的优惠券，甚至与传统价值完全没有联系的用来进行积分奖励的令牌系统。在以太坊中实施令牌系统容易得让人吃惊。关键的一点是理解，所有的货币或者令牌系统，从根本上来说是一个带有如下操作的数据库：从 A 中减去 X 单位并把

X 单位加到 B 上，前提条件是(1)A 在交易之前有至少 X 单位以及(2)交易被 A 批准。实施一个令牌系统就是把这样一个逻辑实施到一个合约中去。

用 Serpent 语言实施一个令牌系统的基本代码如下：

```
def send(to, value):
    if self.storage[from] >= value:
        self.storage[from] = self.storage[from] value
        self.storage[to] = self.storage[to] + value
```

这从本质上来说是本文将要进一步描述的"银行系统"状态转变功能的一个最小化实施。需要增加一些额外的代码以提供在初始和其它一些边缘情况下分发货币的功能，理想情况下会增加一个函数让其它合约来查询一个地址的余额。就足够了。理论上，基于以太坊的充当子货币的令牌系统可能包括一个基于比特币的链上元币所缺乏的重要功能：直接用这种货币支付交易费的能力。实现这种能力的方法是在合约里维护一个以太币账户以用来为发送者支付交易费，通过收集被用来充当交易费用的内部货币并把它们在一个不断运行的拍卖中拍卖掉，合约不断为该以太币账户注资。这样用户需要用以太币"激活"他们的账户，但一旦账户中有以太币它将会被重复使用因为每次合约都会为其充值。

金融衍生品和价值稳定的货币

金融衍生品是"智能合约"的最普遍的应用，也是最易于用代码实现的之一。实现金融合约的主要挑战是它们中的大部分需要参照一个外部的价格发布器；例如，一个需求非常大的应用是一个用来对冲以太币（或其它密码学货币）相对美元价格波动的智能合约，但该合约需要知道以太币相对美元的价格。最简单地方法是通过由某特定机构（例如纳斯达克）维护的"数据提供"合约进行，该合约的设计使得该机构能够根据需要更新合约，并提供一个接口使得其它合约能够通过发送一个消息给该合约以获取包含价格信息的回复。

当这些关键要素都齐备，对冲合约看起来会是下面的样子：

1. 等待 A 输入 1000 以太币。.
2. 等待 B 输入 1000 以太币。
3. 通过查询数据提供合约，将 1000 以太币的美元价值，例如，x 美元，记录至存储器。
4. 30 天后，允许 A 或 B"重新激活"合约以发送价值 x 美元的以太币（重新查询数据提供合约以获取新价格并计算）给 A 并将剩余的以太币发送给 B。 这样的合约在密码学商务中有非同寻常的潜力。密码学货币经常被诟病的一个问题就是其价格的波动性；虽然大量的用户和商家可能需要密码学资产所带来的安全和便利，可他们不太会乐意面对一天中资产跌去 23%价值的情形。直到现在，最为常见的推荐方案是发行者背书资产；思想是发行者创建一种子货币，对此种子货币他们有权发行和赎回，给予（线下）提供给他们一个单位特定相关资产（例如黄金，美元）的人一个单位子货币。发行者承诺当任何人送还一个单位密码学资产时。发还一个单位的相关资产。这种机制能够使任何非密码学资产被"升级"为密码学资产，如果发行者值得信任的话。 然而实践中发行者并非总是值得信任的，并且一些情况下银行体系太脆弱，或者不够诚实守信从而使这样的服务无法存在。金融衍生品提供了一种替代方案。这里将不再有提供储备以支撑一种资产的单独的发行者，取而代之的是一个由赌一种密码学资产的价格会上升的投机者构成的去中心化市场。与发行者不

同，投机者一方没有讨价还价的权利，因为对冲合约把他们的储备冻结在了契约中。注意这种方法并非是完全去中心化的，因为依然需要一个可信任的提供价格信息的数据源，尽管依然有争议这依然是在降低基础设施需求（与发行者不同，一个价格发布器不需要牌照并且似乎可归为自由言论一类）和降低潜在欺诈风险方面的一个巨大的进步。

身份和信誉系统

最早的替代币，域名币，尝试使用一个类比特币块链来提供一个名称注册系统，在那里用户可以将他们的名称和其它数据一起在一个公共数据库注册。最常用的应用案例把象“bitcoin.org”（或者再域名币中，“bitcoin.bit”）一样的域名与一个 IP 地址对应的域名系统。其它的应用案例包括电子邮件验证系统和潜在的更先进的信誉系统。这里是以太坊中提供与域名币类似的的名称注册系统的基础合约：

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

合约非常简单；就是一个以太坊网络中的可以被添加但不能被修改或移除的数据库。任何人都可以把一个名称注册为一个值并永远不变。一个更复杂的名称注册合约将包含允许其他合约查询的“功能条款”，以及一个让一个名称的“拥有者”（即第一个注册者）修改数据或者转让所有权的机制。甚至可以在其上添加信誉和信任网络功能。

去中心化存储

在过去的几年里出现了一些大众化的在线文件存储初创公司，最突出的是 Dropbox，它寻求允许用户上传他们的硬盘备份，提供备份存储服务并允许用户访问从而按月向用户收取费用。然而，在这一点上这个文件存储市场有时相对低效；对现存服务的粗略观察表明，特别地在“神秘谷”20-200GB 这一既没有免费空间也没有企业级用户折扣的水平上，主流文件存储成本每月的价格意味着支付在一个月里支付整个硬盘的成本。以太坊合约允许去中心化存储生态的开发，这样用户通过将他们自己的硬盘或未用的网络空间租出去以获得少量收益，从而降低了文件存储的成本。

这样的设施的基础性构件就是我们所谓的“去中心化 Dropbox 合约”。这个合约工作原理如下。首先，某人将需要上传的数据分成块，对每一块数据加密以保护隐私，并且以此构建一个默克尔树。然后创建一个含以下规则的合约，每 N 个块，合约将从默克尔树中抽取一个随机索引（使用能够被合约代码访问的上一个块的哈希来提供随机性），然后给第一个实体 X 以太以支撑一个带有类似简化验证支付（SPV）的在树中特定索引处的块的所有权证明。当一个用户想重新下载他的文件，他可以使用微支付通道协议（例如每 32k 字节支付 1 萨博）恢复文件；从费用上讲最高效的方法是支付者不到最后不发布交易，而是用一个略微更合算的带有同样随机数的交易在每 32k 字节之后来代替原交易。

这个协议的一个重要特征是，虽然看起来象是一个人信任许多不准备丢失文件的随机节点，但是他可以通过秘密分享把文件分成许多小块，然后通过监视合同得知每个小块都还被某个节点的保存着。如果一个合约依然在付款，那么就提供了某个人依然在保存文件的证据。

去中心化自治组织

通常意义上"去中心化自治组织（DAO, decentralized autonomous organization）"的概念指的是一个拥有一定数量成员或股东的虚拟实体，依靠比如 67%多数来决定花钱以及修改代码。成员会集体决定组织如何分配资金。分配资金的方法可能是悬赏，工资或者更有吸引力的机制比如用内部货币奖励工作。这仅仅使用密码学块链技术就从根本上复制了传统公司或者非营利组织的法律意义以实现强制执行。至此许多围绕 DAO 的讨论都是围绕一个带有接受分红的股东和可交易的股份的"去中心化自治公司（DAC，decentralized autonomous corporation）"的"资本家"模式；作为替代者，一个被描述为"去中心化自治社区（decentralized autonomous community）"的实体将使所有成员都在决策上拥有同等的权利并且在增减成员时要求 67%多数同意。每个人都只能拥有一个成员资格这一规则需要被群体强制实施。

下面是一个如何用代码实现 DO 的纲要。最简单地设计就是一段如果三分之二成员同意就可以自我修改的代码。虽然理论上代码是不可更改的，然而通过把代码主干放在一个单独的合约内并且把合约调用的地址指向一个可更改的存储依然可以容易地绕开障碍而使代码变得可修改，在一个这样的 DAO 合约的简单实现中有三种交易类型，由交易提供的数据区分：

- `[0,i,K,V]` 注册索引为 i 的对存储地址索引为 K 至 v 的内容的更改建议。
- `[0,i]` 注册对建议 i 的投票。
- `[2,i]` 如有足够投票则确认建议 i。

然后合约对每一项都有具体的条款。它将维护一个所有开放存储的更改记录以及一个谁投票表决的表。还有一个所有成员的表。当任何存储内容的更改获得了三分之二多数同意，一个最终的交易将执行这项更改。一个更加复杂的框架会增加内置的选举功能以实现如发送交易，增减成员，甚至提供委任制民主一类的投票代表（即任何人都可以委托另外一个人来代表自己投票，而且这种委托关系是可以传递的，所以如果 A 委托了 B 然后 B 委托了 C 那么 C 将决定 A 的投票）。这种设计将使 DAO 作为一个去中心化社区有机地成长， 使人们最终能够把挑选合适人选的任务交给专家，与当前系统不同，随着社区成员不断改变他们的站队假以时日专家会容易地出现和消失。 一个替代的模式是去中心化公司，那里任何账户可以拥有 0 到更多的股份，决策需要三分之二多数的股份同意。一个完整的框架将包括资产管理功能-可以提交买卖股份的订单以及接受这种订单的功能（前提是合约里有订单匹配机制）。代表依然以委任制民主的方式存在，产生了"董事会"的概念。

更先进的组织治理机制可能会在将来实现；现在一个去中心化组织（DO）可以从去中心化自治组织（DAO）开始描述。DO 和 DAO 的区别是模糊的，一个大致的分割线是治理是否可以通过一个类似政治的过程或者一个"自动"过程实现，一个不错的直觉测试是"无通用语言"标准：如果两个成员不说同样的语言组织还能正常运行吗？显然，一个简单的传统的持股式公司会失败，而象比特币协议这样的却很可能成功，罗宾·汉森的"futarchy"，一个通过预测市场实现组织化治理的机制是一个真正的说明"自治"式治理可能是什么样子的好例子。注意一个人无需假设所有 DAO 比所有 DO 优越；自治只是一个在一些特定场景下有很大优势的，但在其它地方未必可行的范式，许多半 DAO 可能存在。

进一步的应用

1. **储蓄钱包**。 假设 Alice 想确保她的资金安全，但她担心丢失或者被黑客盗走私钥。她把以太币放到和 Bob 签订的一个合约里，如下所示，这合同是一个银行：

- Alice 单独每天最多可提取 1%的资金。
- Bob 单独每天最多可提取 1%的资金，但 Alice 可以用她的私钥创建一个交易取消 Bob 的提现权限。
- Alice 和 Bob 一起可以任意提取资金。 一般来讲，每天 1%对 Alice 足够了，如果 Alice 想提现更多她可以联系 Bob 寻求帮助。如果 Alice 的私钥被盗，她可以立即找到 Bob 把她的资金转移到一个新合同里。如果她弄丢了她的私钥，Bob 可以慢慢地把钱提出。如果 Bob 表现出了恶意，她可以关掉他的提现权限。

2. **作物保险**。一个人可以很容易地以天气情况而不是任何价格指数作为数据输入来创建一个金融衍生品合约。如果一个爱荷华的农民购买了一个基于爱荷华的降雨情况进行反向赔付的金融衍生品，那么如果遇到干旱，该农民将自动地收到赔付资金而如果有足量的降雨他会很开心因为他的作物收成会很好。

3. **一个去中心化的数据发布器**。 对于基于差异的金融合约，事实上通过过“谢林点”协议将数据发布器去中心化是可能的。谢林点的工作原理如下：N 方为某个指定的数据提供输入值到系统（例如 ETH/USD 价格），所有的值被排序，每个提供 25%到 75%之间的值的节点都会获得奖励，每个人都有激励去提供他人将提供的答案，大量玩家可以真正同意的答案明显默认就是正确答案，这构造了一个可以在理论上提供很多数值，包括 ETH/USD 价格，柏林的温度甚至某个特别困难的计算的结果的去中心化协议。

5.**云计算**。EVM 技术还可被用来创建一个可验证的计算环境，允许用户邀请他人进行计算然后选择性地要求提供在一定的随机选择的检查点上计算被正确完成的证据。这使得创建一个任何用户都可以用他们的台式机，笔记本电脑或者专用服务器参与的云计算市场成为可能，现场检查和安全保证金可以被用来确保系统是值得信任的（即没有节点可以因欺骗获利）。虽然这样一个系统可能并不适用所有任务；例如，需要高级进程间通信的任务就不易在一个大的节点云上完成。然而一些其它的任务就很容易实现并行；SETI@home, folding@home 和基因算法这样的项目就很容易在这样的平台上进行。

6.**点对点赌博**。任意数量的点对点赌博协议都可以搬到以太坊的区块链上，例如 Frank Stajano 和 Richard Clayton 的 Cyberdice。 最简单的赌博协议事实上是这样一个简单的合约，它用来赌下一个区块的哈稀值与猜测值之间的差额, 据此可以创建更复杂的赌博协议，以实现近乎零费用和无欺骗的赌博服务。

7.**预测市场**。 不管是有神谕还是有谢林币，预测市场都会很容易实现，带有谢林币的预测市场可能会被证明是第一个主流的作为去中心化组织管理协议的“futarchy”应用。

8.链上去中心化市场，以身份和信誉系统为基础。

杂项和关注

改进版幽灵协议的实施

“幽灵”协议（"Greedy Heaviest Observed Subtree" (GHOST) protocol）是由 Yonatan Sompolinsky 和 Aviv Zohar 在 2013 年 12 月引入的创新。幽灵协议提出的动机是当前快

速确认的块链因为区块的高作废率而受到低安全性困扰；因为区块需要花一定时间（设为 t）扩散至全网，如果矿工 A 挖出了一个区块然后矿工 B 碰巧在 A 的区块扩散至 B 之前挖出了另外一个区块，矿工 B 的区块就会作废并且没有对网络安全作出贡献。此外，这里还有中心化问题：如果 A 是一个拥有全网 30%算力的矿池而 B 拥有 10%的算力，A 将面临 70%的时间都在产生作废区块的风险而 B 在 90%的时间里都在产生作废区块。因此，如果作废率高，A 将简单地因为更高的算力份额而更有效率，综合这两个因素，区块产生速度快的块链很可能导致一个矿池拥有实际上能够控制挖矿过程的算力份额。

正如 Sompolinsky 和 Zohar 所描述的，通过在计算哪条链"最长"的时候把废区块也包含进来，幽灵协议解决了降低网络安全性的第一个问题；这就是说，不仅一个区块的父区块和更早的祖先块，祖先块的作废的后代区块（以太坊术语中称之为"叔区块"）也被加进来以计算哪一个区块拥有支持其的最大工作量证明。我们超越了 Sompolinsky 和 Zohar 所描述的协议以解决第二个问题 – 中心化倾向，以太坊付给以"叔区块"身份为新块确认作出贡献的废区块 87.5%的奖励，把它们纳入计算的"侄子区块"将获得奖励的 12.5%，不过，交易费用不奖励给叔区块。 以太坊实施了一个只下探到第五层的简化版本的幽灵协议。其特点是，废区块只能以叔区块的身份被其父母的第二代至第五代后辈区块，而不是更远关系的后辈区块（例如父母区块的第六代后辈区块，或祖父区块的第三代后辈区块）纳入计算。这样做有几个原因。首先，无条件的幽灵协议将给计算给定区块的哪一个叔区块合法带来过多的复杂性。其次，带有以太坊所使用的补偿的无条件的幽灵协议剥夺了矿工在主链而不是一个公开攻击者的链上挖矿的激励。最后，计算表明带有激励的五层幽灵协议即使在出块时间为 15s 的情况下也实现了了 95%以上的效率，而拥有 25%算力的矿工从中心化得到的益处小于 3%。

费用

因为每个发布的到区块链的交易都占用了下载和验证的成本，需要有一个包括交易费的规范机制来防范滥发交易。比特币使用的默认方法是纯自愿的交易费用，依靠矿工担当守门人并设定动态的最低费用。因为这种方法是"基于市场的"，使得矿工和交易发送者能够按供需来决定价格，所以这种方法在比特币社区被很顺利地接受了。然而，这个逻辑的问题在于，交易处理并非一个市场；虽然根据直觉把交易处理解释成矿工给发送者提供的服务是很有吸引力的，但事实上一个矿工收录的交易是需要网络中每个节点处理的，所以交易处理中最大部分的成本是由第三方而不是决定是否收录交易的矿工承担的。于是，非常有可能发生公地悲剧。

然而，当给出一个特殊的不够精确的简化假设时，这个基于市场的机制的漏洞很神奇地消除了自己的影响。论证如下。假设：

1. 一个交易带来 k 步操作, 提供奖励 kR 给任何收录该交易的矿工，这里 R 由交易发布者设定， k 和 R 对于矿工都是事先（大致上）可见的。
2. 每个节点处理每步操作的成本都是 C (即所有节点的效率一致)。
3. 有 N 个挖矿节点，每个算力一致(即全网算力的 1/N)。
4. 没有不挖矿的全节点。

当预期奖励大于成本时，矿工愿意挖矿。这样，因为矿工有 1/N 的机会处理下一个区块，所以预期的收益是 kR/N , 矿工的处理成本简单为 kC. 这样当 kR/N > kC， 即 R > NC 时。矿工愿意收录交易。注意 R 是由交易发送者提供的每步费用，是矿工从处理交易中

获益的下限。 NC 是全网处理一个操作的成本。所以，矿工仅有动机去收录那些收益大于成本的交易。 然而，这些假设与实际情况有几点重要的偏离：

1. 因为额外的验证时间延迟了块的广播因而增加了块成为废块的机会，处理交易的矿工比其它的验证节点付出了更高的成本。
2. 不挖矿的全节点是存在的。
3. 实践中算力分布可能最后是极端不平均的。
4. 以破坏网络为己任的投机者，政敌和疯子确实存在，并且他们能够聪明地设置合同使得他们的成本比其它验证节点低得多。 上面第 1 点驱使矿工收录更少的交易，第 2 点增加了 NC; 因此这两点的影响至少部分互相抵消了. 第 3 点和第 4 点是主要问题；作为解决方案我们简单地建立了一个浮动的上限：没有区块能够包含比 BLK_LIMIT_FACTOR 倍长期指数移动平均值更多的操作数。具体地：

```
blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR - 1) + floor(parent.opcount * BLK_LIMIT_FACTOR)) /EMA_FACTOR)
```

BLK_LIMIT_FACTOR 和 EMA_FACTOR 是暂且被设为 65536 和 1.5 的常数，但可能会在更深入的分析后调整。 回复

计算和图灵完备

需要强调的是以太坊虚拟机是图灵完备的； 这意味着 EVM 代码可以实现任何可以想象的计算，包括无限循环。EVM 代码有两种方式实现循环。首先， JUMP 指令可以让程序跳回至代码前面某处，还有允许如 `while x < 27: x = x * 2` 一样的条件语句的 JUMPI 指令实现条件跳转。其次，合约可以调用其它合约，有通过递归实现循环的潜力。这很自然地导致了一个问题：恶意用户能够通过迫使矿工和全节点进入无限循环而不得不关机吗？ 这问题出现是因为计算机科学中一个叫停机问题的问题：一般意义上没有办法知道，一个给定的程序是否能在有限的时间内结束运行。

正如在状态转换章节所述，我们的方案通过为每一个交易设定运行执行的最大计算步数来解决问题，如果超过则计算被恢复原状但依然要支付费用。消息以同样的方式工作。为显示这一方案背后的动机，请考虑下面的例子：

- 一个攻击者创建了一个运行无限循环的合约，然后发送了一个激活循环的交易给矿工，矿工将处理交易，运行无限循环直到瓦斯耗尽。即使瓦斯耗尽交易半途停止，交易依然正确（回到原处）并且矿工依然从攻击者哪里挣到了每一步计算的费用。
- 一个攻击者创建一个非常长的无限循环意图迫使矿工长时间内一直计算致使在计算结束前若干区块已经产生于是矿工无法收录交易以赚取费 用。然而，攻击者需要发布一个 STARTGAS 值以限制可执行步数，因而矿工将提前知道计算将耗费过多的步数。
- 一个攻击者看到一个包含诸如 send(A,self.storage); self.storage = 0 格式的合约然后发送带有只够执行第一步的费用的而不够执行第二步的交易（即提现但不减少账户余额）。合约作者无需担心防卫类似攻击，因为如果执行中途停止则所有变更都被回复。
- 一个金融合约靠提取九个专用数据发布器的中值来工作以最小化风险，一个攻击者接管了其中一个数据提供器，然后把这个按 DAO 章节所述的可变地址调用机制设

计成可更改的数据提供器转为运行一个无限循环，以求尝试逼迫任何从此金融合约索要资金的尝试都会因瓦斯耗尽而中止。然而，该金融合约可以在消息里设置瓦斯限制以防范此类问题。 图灵完备的替代是图灵不完备，这里 JUMP 和 JUMPI 指令不存在并且在某个给定时间每个合约只允许有一个拷贝存在于调用堆栈内。在这样的系统里，上述的费用系统和围绕我们的方案的效率的不确定性可能都是不需要的，因为执行一个合约的成本将被它的大小决定。此外，图灵不完备甚至不是一个大的限制，在我们内部设想的所有合约例子中，至今只有一个需要循环，而且即使这循环也可以被 26 个单行代码段的重复所代替。考虑到图灵完备带来的严重的麻烦和有限的益处，为什么不简单地使用一种图灵不完备语言呢？事实上图灵不完备远非一个简洁的解决方案。为什么？请考虑下面的合约：

```
C0: call(C1); call(C1);
C1: call(C2); call(C2);
C2: call(C3); call(C3);
...
C49: call(C50); call(C50);
C50: (作一个图灵机的步计算和记录结果在合约的长期存储)
```

现在，发送一个这样的交易给 A，这样，在 51 个交易中，我们有了一个需要花费 2^50 步计算的合约，矿工可能尝试通过为每一个合约维护一个最高可执行步数并且对于递归调用其它合约的合约计算可能执行步数从而预先检测这样的逻辑炸弹，但是这会使矿工禁止创建其它合约的合约（因为上面 26 个合约的创建和执行可以很容易地放入一个单独合约内）。另外一个问题点是一个消息的地址字段是一个变量，所以通常来讲可能甚至无法预先知道一个合约将要调用的另外一个合约是哪一个。于是，最终我们有了一个惊人的结论：图灵完备的管理惊人地容易，而在缺乏同样的控制时图灵不完备的管理惊人地困难-那为什么不让协议图灵完备呢？

货币和发行

以太坊网络包含自身的内置货币以太币，以太币扮演双重角色，为各种数字资产交易提供主要的流动性，更重要的是提供了了支付交易费用的一种机制。为便利及避免将来的争议期间（参见当前的 mBTC/uBTC/聪的争论），不同面值的名称将被提前设置：

- 1: 伟
- 10^12: 萨博
- 10^15: 芬尼
- 10^18: 以太

这应该被当作是“元”和“分”或者“比特币”和“聪”的概念的扩展版，在不远的将来，我们期望“以太”被用作普通交易，“芬尼”用来进行微交易，“萨博”和“伟”用来进行关于费用和协议实施的讨论。

发行模式如下：

- 通过发售活动，以太币将以每 BTC 1337-2000 以太的价格发售，一个旨在为以太坊组织筹资并且为开发者支付报酬的机制已经在其它一些密码学货币平台上成功使

用。早期购买者会享受较大的折扣，发售所得的 BTC 将完全用来支付开发者和研究者的工资和悬赏，以及投入密码学货币生态系统的项目。

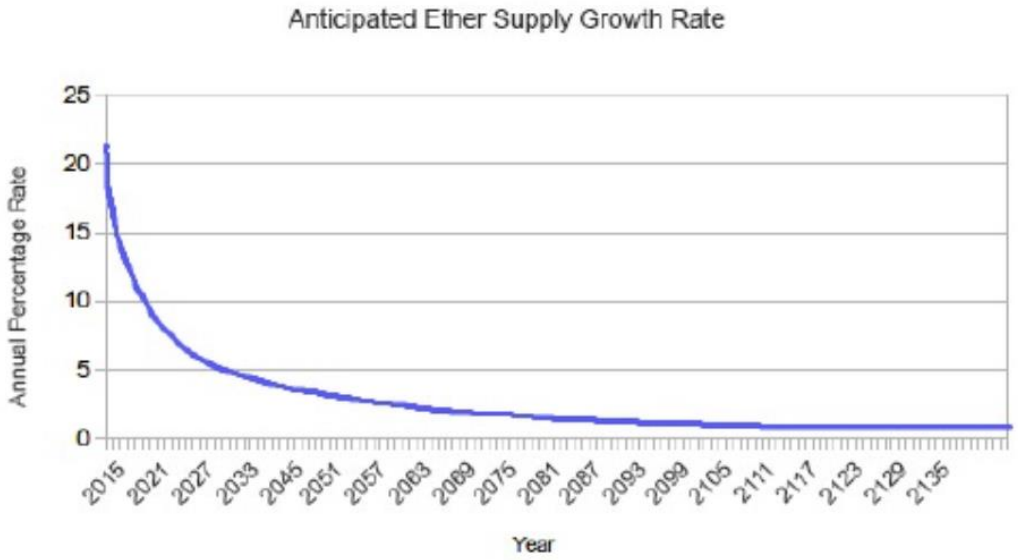
- 0.099x （x 为发售总量）将被分配给 BTC 融资或其它的确定性融资成功之前参与开发的早期贡献者，另外一个 0.099x 将分配给长期研究项目。
- 自上线时起每年都将有 0.26x（x 为发售总量）被矿工挖出。

发行分解

永久线性增长模型降低了在比特币中出现的财富过于集中的风险，并且给予了活在当下和将来的人公平的机会去获取货币，同时保持了对获取和持有以太币的激励，因为长期来看“货币供应增长率”是趋于零的。我们还推断，随着时间流逝总会发生因为粗心和死亡等原因带来的币的遗失，假设币的遗失是每年货币供应量的一个固定比例，则最终总的流通中的货币供应量会稳定在一个等于年货币发行量除以遗失率的值上（例如，当遗失率为 1% 时，当供应量达到 30x 时，每年有 0.3x 被挖出同时有 0.3x 丢失，达到一个均衡）。

分组	上线时	一年后	五年后
货币单位	1.198X	1.458X	2.498X
购买者	83.5%	68.6%	40.0%
早期贡献者分配	8.26%	6.79%	3.96%
长期捐赠	8.26%	6.79%	3.96%
矿工	0%	17.8%	52.0%

除了线性的发行方式外，和比特币一样以太币的的供应量增长率长期来看也趋于零。



挖矿的中心化

比特币挖矿算法基本上是让矿工千万次地轻微改动区块头，直到最终某个节点的改动版本的哈希小于目标值（目前是大约 2190）。然而，这种挖矿算法容易被两种形式的中心化攻击。第一种，挖矿生态系统被专门设计的因而在比特币挖矿这一特殊任务上效率提高上千倍的 ASICs（专用集成电路）和电脑芯片控制。这意味着比特币挖矿不再是高度去中心化的和追求平等主义的，而是需要巨额资本的有效参与。第二种，大部分比特币矿工事实上不再在本地完成区块验证；而是依赖中心化的矿池提供区块头。这个问题可以说很严重：在本文写作时，最大的两个矿池间接地控制了大约全网 50%的算力，虽然当一个矿池或联合体尝试 51%攻击时矿工可以转换到其它矿池这一事实减轻了问题的严重性。

以太坊现在的目的是使用一个基于为每 1000 个随机数随机产生唯一哈希的函数的挖矿算法，用足够宽的计算域，去除专用硬件的优势。这样的策略当然不会使中心化的收益减少为零，但是也不需要。注意每单个用户使用他们的私人笔记本电脑或台式机就可以几乎免费地完成一定量的挖矿活动，但当到了 100%的 CPU 使用率之后更多地挖矿就会需要他们支付电力和硬件成本。ASIC 挖矿公司需要从第一个哈希开始就为电力和硬件支付成本。所以，如果中心化收益能够保持在(E + H) /E 以下，那么即使 ASICs 被制造出来普通矿工依然有生存空间。另外，我们计划将挖矿算法设计成挖矿需要访问整个区块链，迫使矿工存储完成的区块链或者至少能够验证每笔交易。这去除了对中心化矿池的需要；虽然矿池依然可以扮演平滑收益分配的随机性的角色，但这功能可以被没有中心化控制的 P2P 矿池完成地同样好。这样即使大部分普通用户依然倾向选择轻客户端，通过增加网络中的全节点数量也有助于抵御中心化。

扩展性

扩展性问题是以太坊常被关注的地方，与比特币一样，以太坊也遭受着每个交易都需要网络中的每个节点处理这一困境的折磨。比特币的当前区块链大小约为 20GB，以每小时 1MB 的速度增长。如果比特币网络处理 Visa 级的 2000tps 的交易，它将以每三秒 1MB 的速度增长（1GB 每小时，8TB 每年）。以太坊可能也会经历相似的甚至更糟的增长模式，因为在以太坊区块链之上还有很多应用，而不是像比特币只是简单的货币，但以太坊全节点只需存储状态而不是完整的区块链历史这一事实让情况得到了改善。

大区块链的问题是中心化风险。如果块链大小增加至比如 100TB，可能的场景将是只有非常小数目的大商家会运行全节点，而常规用户使用轻的 SPV 节点。这会增加对全节点合伙欺诈牟利（例如更改区块奖励，给他们自己 BTC）的风险的担忧。轻节点将没有办法立刻检测到这种欺诈。当然，至少可能存在一个诚实的全节点，并且几个小时之后有关诈骗的信息会通过 Reddit 这样的渠道泄露，但这时已经太晚：任凭普通用户做出怎样的努力去废除已经产生的区块，他们都会遇到与发动一次成功的 51%攻击同等规模的巨大的不可行的协调问题。在比特币这里，现在这是一个问题，但 Peter Todd 建议的一个改动可以缓解这个问题。

近期，以太坊会使用两个附加的策略以应对此问题。首先，因为基于区块链的挖矿算法，至少每个矿工会被迫成为一个全节点，这保证了一定数量的全节点。其次，更重要的是，处理完每笔交易后，我们会把一个中间状态树的根包含进区块链。即使区块验证是中心化的，只要有一个诚实的验证节点存在，中心化的问题就可以通过一个验证协议避免。如果一个矿工发布了一个不正确的区块，这区块要么是格式错，要么状态 S[n]是错的。因为 S[0]是正确的，必然有第一个错误状态 S[i]但 S[i-1]是正确的，验证节点将提供索引 i，一起

提供的还有处理 APPLY(S[i-1],TX[i]) -> S[i]所需的帕特里夏树节点的子集。这些节点将受命进行这部分计算，看产生的 S[i]与先前提供的值是否一致。

另外，更复杂的是恶意矿工发布不完整区块进行攻击，造成没有足够的信息去确定区块是否正确。解决方案是质疑-回应协议：验证节点对目标交易索引发起质疑，接受到质疑信息的轻节点会对相应的区块取消信任，直到另外一个矿工或者验证者提供一个帕特里夏节点子集作为正确的证据。

综述：去中心化应用

上述合约机制使得任何一个人能够在一个虚拟机上建立通过全网共识来运行命令行应用（从根本上来说是），它能够更改一个全网可访问的状态作为它的“硬盘”。然而，对于多数人来说，用作交易发送机制的命令行接口缺乏足够的用户友好使得去中心化成为有吸引力的替代方案。最后，一个完整的“去中心化应用”应该包括底层的商业逻辑组件【无论是否在以太坊完整实施，使用以太坊和其它系统组合（如一个 P2P 消息层，其中一个正在计划放入以太坊客户端）或者仅有其它系统的方式】和上层的图形用户接口组件。以太坊客户端被设计成一个网络浏览器，但包括对“eth” Javascript API 对象的支持，可被客户端里看到的特定的网页用来与以太坊区块链交互。从“传统”网页的角度看来，这些网页是完全静态的内容，因为区块链和其它去中心化协议将完全代替服务器来处理用户发起的请求。最后，去中心化协议有希望自己利用某种方式使用以太坊来存储网页。

结论

以太坊协议最初是作为一个通过高度通用的语言提供如链上契约，提现限制和金融合约，赌博市场等高级功能的升级版密码学货币来构思的。以太坊协议将不直接“支持”任何应用，但图灵完备编程语言的存在意味着理论上任意的合约都可以为任何交易类型和应用创建出来。然而关于以太坊更有趣的是，以太坊协议比单纯的货币走得更远，围绕去中心化存储，去中心化计算和去中心化预测市场以及数十个类似概念建立的协议和去中心化应用，有潜力从根本上提升计算行业的效率，并通过首次添加经济层为其它的 P2P 协议提供有力支撑，最终，同样会有大批与金钱毫无关系的应用出现。

以太坊协议实现的任意状态转换概念提供了一个具有独特潜力的平台；与封闭式的，为诸如数据存储，赌博或金融等单一目的设计的协议不同，以太坊从设计上是开放式的，并且我们相信它极其适合作为基础层服务于在将来的年份里出现的极其大量的金融和非金融协议。

注解与进阶阅读

注解

1.一个有经验的读者会注意到事实上比特币地址是椭圆曲线公钥的哈希，而非公钥本身，然而事实上从密码学术语角度把公钥哈希称为公钥完全合理。这是因为比特币密码学可以被认为是一个定制的数字签名算法，公钥由椭圆曲线公钥的哈希组成，签名由椭圆曲线签名连接的椭圆曲线公钥组成，而验证算法包括用作为公钥提供的椭圆曲线公钥哈希来检查椭圆曲线公钥，以及之后的用椭圆曲线公钥来验证椭圆曲线签名。

2.技术上来说，前 11 个区块的中值。

3.在内部，2 和“CHARLIE”都是数字，后一个有巨大的 base256 编码格式，数字可以从 0 到 2^256-1。

进阶阅读

1. Intrinsic value: https://tinyurl.com/BitcoinMag-IntrinsicValue
2. Smart property: https://en.bitcoin.it/wiki/Smart_Property
3. Smart contracts: https://en.bitcoin.it/wiki/Contracts
4. B-money: http://www.weidai.com/bmoney.txt
5. Reusable proofs of work: http://www.finney.org/~hal/rpow/
6. Secure property titles with owner authority: http://szabo.best.vwh.net/securetitle.html
7. Bitcoin whitepaper: http://bitcoin.org/bitcoin.pdf
8. Namecoin: https://namecoin.org/
9. Zooko’s triangle: http://en.wikipedia.org/wiki/Zooko’s_triangle
10. Colored coins whitepaper: https://tinyurl.com/coloredcoin-whitepaper
11. Mastercoin whitepaper: https://github.com/mastercoin-MSC/spec
12. Decentralized autonomous corporations, Bitcoin Magazine: https://tinyurl.com/Bootstrapping-DACs
13. Simplified payment verification:https://en.bitcoin.it/wiki/Scalability#Simplifiedpaymentverification
14. Merkle trees: http://en.wikipedia.org/wiki/Merkle_tree
15. Patricia trees: http://en.wikipedia.org/wiki/Patricia_tree
16. GHOST: http://www.cs.huji.ac.il/~avivz/pubs/13/btc_scalability_full.pdf
17. StorJ and Autonomous Agents, Jeff Garzik: https://tinyurl.com/storj-agents
18. Mike Hearn on Smart Property at Turing Festival: http://www.youtube.com/watch?v=Pu4PAMFPo5Y
19. Ethereum RLP: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-RLP
20. Ethereum Merkle Patricia trees: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-Patricia-Tree
21. Peter Todd on Merkle sum trees:http://sourceforge.net/p/bitcoin/mailman/message/31709140/