# 导读

以太坊是什么？

以太坊是一个全新开放的区块链平台，它允许任何人在平台中建立和使用通过区块链技术运行的去中心化应用。就像比特币一样，以太坊不受任何人控制，也不归任何人所有——它是一个开放源代码项目，由全球范围内的很多人共同创建。和比特币协议有所不同的是，以太坊的设计十分灵活，极具适应性。在以太坊平台上创立新的应用十分简便，随着 Homestead 的发布，任何人都可以安全地使用该平台上的应用。

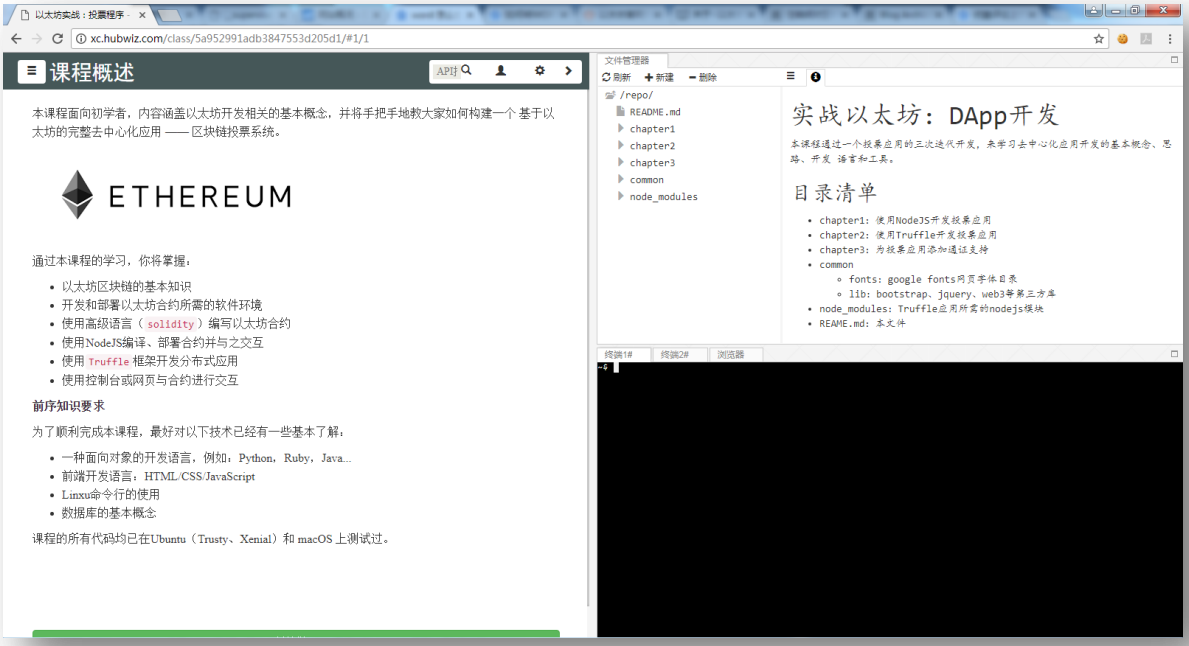
本电子书原文最早由蓝莲花（汪晓明）于 2016 年发布于其博客（http://wangxiaoming.com），由汇智网（http://www.hubwiz.com）编目整理，是目前网上流传的最完整的官网文档中文版。相信众多从事以太坊开发的极客们，都曾受益于汪晓明先生的辛苦付出。

但由于以太坊本身（以及周边生态）的发展非常快，一些实践性内容已经落后于现状。因此编者建议本电子书的读者，在阅读时应注意吸收核心的理念思想，而不要过分关注书中的实践操作环节。

为了弥补这一遗憾，汇智网推出了在线交互式以太坊 DApp 实战开发课程，以去中心化投票应用（Voting DApp）为课程项目，通过三次迭代开发过程的详细讲解与在线实践，并且将区块链的理念与去中心化思想贯穿于课程实践过程中，为希望快速入门区块链开发的开发者提供了一个高效的学习与价值提升途径。读者可以通过以下链接访问《以太坊 DApp 开发实战入门》在线教程：

http://xc.hubwiz.com/course/5a952991adb3847553d205d1

教程预置了开发环境。进入教程后，可以在每一个知识点立刻进行同步实践，而不必在开发环境的搭建上浪费时间：




汇智网 Hubwiz.com

2018.2

# 第一章 以太坊是什么？

## 1.1 以太坊是什么？

以太坊是一个全新开放的区块链平台，它允许任何人在平台中建立和使用通过区块链技术运行的去中心化应用。就像比特币一样，以太坊不受任何人控制，也不归任何人所有——它是一个开放源代码项目，由全球范围内的很多人共同

创建。和比特币协议有所不同的是，以太坊的设计十分灵活，极具适应性。在以太坊平台上创立新的应用十分简便，随着 Homestead 的发布，任何人都可以安全地使用该平台上的应用。

## 1.2 下一代区块链

区块链技术是比特币的底层技术，这一技术第一次被描述是在中本聪 2008 年发表的白皮书“比特币：点对点电子现金系统”中。区块链技术更多的一般性用途在原书中已经有所讨论，但直到几年后，区块链技术才作为通用术语出现。一个区块链是一个分布式计算架构，里面的每个网络节点执行并记录相同的交易，交易被分组为区块。一次只能增加一个区块，每个区块有一个数学证明来保证新的区块与之前的区块保持先后顺序。这样一来，区块链的“分布式数据库”就能和整个网络保持一致。个体用户与总账的互动（交易）受到安全的密码保护。由数学执行并编码到协议中的经济激励因素刺激着维持和验证网络的节点。

在比特币中，分布式数据库被设想为一个账户余额表，一个总账，交易就是通过比特币的转移以实现个体之间无需信任基础的金融活动。但是随着比特币吸引了越来越多开发者和技术专家的注意，新的项目开始将比特币网络用于有价代币转移之外的其他用途。其中很多都采用了“代币”的形式——以原始比特币协议为基础，增加了新的特征或功能，采用各自加密货币的独立区块链。在 2013 年末，以太坊的发明者 Vitalik Buterin 建议能够通过程序重组来运行任意复杂运算的单个区块链应该包含其他的程序。

2014 年，以太坊的创始人 Vitalik Buterin, Gavin Wood 和 Jeffrey Wilcke 开始研究新一代区块链，试图实现一个总体上完全无需信任基础的智能合约平台。

## 1.3 以太坊虚拟机

以太坊是可编程的区块链。它并不是给用户一系列预先设定好的操作（例如比特币交易），而是允许用户按照自己的意愿创建复杂的操作。这样一来，它就可以作为多种类型去中心化区块链应用的平台，包括加密货币在内但并不仅限于此。

以太坊狭义上是指一系列定义去中心化应用平台的协议，它的核心是以太坊虚拟机（“EVM”），可以执行任意复杂算法的编码。在计算机科学术语中，以太坊是“图灵完备的”。开发者能够使用现有的 JavaScript 和 Python 等语言为模型的其他友好的编程语言，创建出在以太坊模拟机上运行的应用。

和其他区块链一样，以太坊也有一个点对点网络协议。以太坊区块链数据库由众多连接到网络的节点来维护和更新。每个网络节点都运行着以太坊模拟机并执行相同的指令。因此，人们有时形象地称以太坊为“世界电脑”。

这个贯穿整个以太坊网络的大规模并行运算并不是为了使运算更高效。实际上，这个过程使得在以太坊上的运算比在传统“电脑”上更慢更昂贵。然而，每个以太坊节点都运行着以太坊虚拟机是为了保持整个区块链的一致性。去中心化的一致使以太坊有极高的故障容错性，保证零停机，而且可以使存储在区块链上的数据保持永远不变且抗审查。

以太坊平台本身没有特点，没有价值性。和编程语言相似，它由企业家和开发者决定其用途。不过很明显，某些应用类型较之其他更能从以太坊的功能中获益。以太坊尤其适合那些在点与点之间自动进行直接交互或者跨网络促进小组协调活动的应用。例如 ,协调点对点市场的应用 ,或是复杂财务合约的自动化。比特币使个体能够不借助金融机构、银行或政府等其他中介来进行货币交换。以太坊的影响可能更为深远。理论上，任何复杂的金融活动或交易都能在以太坊上用编码自动且可靠地进行。除金融类应用外，任何对信任、安全和持久性要求较高的应用场景——比如资产注册、投票、管理和物联网——都会大规模地受到以太坊平台影响。

## 1.4 以太坊如何工作？

以太坊合并了很多对比特币用户来说十分熟悉的特征和技术，同时自己也进行了很多修正和创新。比特币区块链纯粹是一个关于交易的列表，而以太坊的基础单元是账户。以太坊区块链跟踪每个账户的状态，所有以太坊区块链上的状态转换都是账户之间价值和信息的转移。账户分为两类：

外部账户（ EOA ），由私人密码控制 合约账户，由它们的合约编码控制，只能由外部账户“激活” 对于大部分用户来说，两者基本的区别在于外部账户是由人类用户掌控——因为他们能够控制私钥，进而控制外部账户。而合约账户则是由内部编码管控。如果他们是被人类用户“控制”的，那也是因为程序设定它们被具有特定地址的外部账户控制，进而被持有私钥控制外部账户的人控制着。“智能合约”这个流行的术语指的是在合约账户中编码——交易被发

送给该账户时所运行的程序。用户可以通过在区块链中部署编码来创建新的合约。

只有当外部账户发出指令时，合约账户才会执行相应的操作。所以合约账户不可能自发地执行诸如任意数码生成或应用程序界面调用等操作—只有受外部账户提示时，它才会做这些事。这是因为以太坊要求节点能够与运算结果保持一致，这就要求保证严格确定执行。

和比特币一样，以太坊用户必须向网络支付少量交易费用。这可以使以太坊区块链免受无关紧要或恶意的运算任务干扰，比如分布式拒绝服务（DDoS）攻击或无限循环 。交易的发送者必须在激活的“程序”每一步付款，包括运算和记忆储存。费用通过以太坊自有的有价代币，以太币的形式支付。

交易费用由节点收集，节点使网络生效。这些“矿工”就是以太坊网络中收集、传播、确认和执行交易的节点。矿工们将交易分组——包括许多以太坊区块链中账户“状态”的更新——分成的组被称为“区块”，矿工们会互相竞争，以使他们的区块可以添加到下一个区块链上。矿工们每挖到一个成功的区块就会得到以太币奖励。这就为人们带来了经济激励，促使人们为以太坊网络贡献硬件和电力。

和比特币网络一样，矿工们有解决复杂数学问题的任务以便成功地“挖”到区块。这被称为“工作量证明”。一个运算问题，如果在算法上解决，比验证解决方法需要更多数量级的资源，那么它就是工作证明的极佳选择。为防止比特币网络中已经发生的，专门硬件( 例如特定用途集成电路 )造成的中心化现象，以太坊选择了难以存储的运算问题。如果问题需要存储器和 CPU，事实上理想

的硬件是普通的电脑。这就使以太坊的工作量证明具有抗特定用途集成电路性，和比特币这种由专门硬件控制挖矿的区块链相比，能够带来更加去中心化的安全分布。

# 第二章 如何使用文档及以太坊路线图

## 2.1 以太坊的使用：基础指南

通过本节可以获取用户参与到以太坊项目中的基本方法。首先，要想成为网络中的节点，需要运行一个以太坊客户端。在选择客户端这一节中列出了多重实现，同时针对不同的安装应选择什么样的客户端给出了建议。连接到网络会告诉你关于网络、连接故障排除和区块链同步的基本信息。设立私有链等高级的网络主题可以在测试网络章节中看到。

## 2.2 Homestead 的发布

Homestead 是以太坊平台的第二个主要版本，也是以太坊发布的第一个正式版本。它包括几处协议变更和网络设计变更，使网络进一步升级成为可能。以太坊的第一个版本 Frontier 实际上是测试版，供开发者学习、试验并开始建立以太坊去中心化的应用和工具。

## 2.3 以太坊开发路线图中的里程碑

以太坊上线之前计划的初始开发路线图主要有以下几个里程碑：

- 预发布：Olympic testnet —— 2015 年 5 月发布
- 发布第一个版本：Frontier —— 2015 年 7 月 30 日发布
- 发布第二个版本： Homestead —— 2016 年 3 月 14 日发布（π日）
- 发布第三个版本： Metropolis —— 即将宣布

- 发布第四个版本：Serenity ——— 即将宣布

尽管仍然有效，但它背后的实质已有所改变。Olympic testnet 阶段（Frontier 发布之前）见证了很多主要的改进，紧接着就发布了 Frontier。Homestead 标志着测试版结束，开始发布正式版本。Homestead 会自动在 1,150,000 号区块引入，大概会发生在 2016 年 3 月 14 日，也就是π日前后。

如果你正在运行一个和实时网络连接的节点，非常有必要升级到 Homestead 兼容的客户端。这些客户端版本列在以太坊客户端下。如果客户端不兼容，你会进入到错误的分叉，不能和网络其他部分同步。

以太坊区块链一旦到达 1,150,000 号区块，以太坊网络就会经历一个硬分叉，带来几项主要变更，这将在下一章节中阐述。

## 2.4 Homestead 硬分叉变更

以太坊从狭义上来说，是一系列协议。Homestead 带来了几个反向不兼容的协议变更，进而要求硬分叉。这些变更在过程中向以太坊改进建议靠拢，主要包括以下几个内容：

**EIP 2:**

——通过交易创建合约的费用由 21000 增加到 53000。用 CREATE 操作码通过合约来创建合约不受影响。

——S 值比 secp256k1n/2 大的交易签名现在被认定无效。

——如果创建合约时没有足够的 gas 用来支付给状态增加合约编码所需的最终 gas 费用，合约创建就会失败（例如，无 gas 可用）而不会留下一个空合约。

——改变算法难度调整

**EIP 7: DELEGATECALL：** 增加一个新的操作码，DELEGATECALL at 0xf4，它和 CALLCODE 的概念相似，不过会把发送者和父作用域的价值发送到子作用域，比如，创建的调用与原始调用具有相同的发送者和价值。这就意味着合约可以通过信息存储通路，同时遵从父合约中的信息发送者（msg.sender）和信息价值（msg.value）。这样对创建合约的合约来说是好事，但是不要重复那些存储 gas 的附加信息。参见对 EIP 7 的评论。

**EIP 8: devp2p** 向前兼容性符合健壮性原则 RLPx 发现协议和 RLPx TCP 传输协议确保以太坊网路上使用的客户端软件可以应对将来的网络协议升级。对于以太坊的旧版本来说，网络协议升级并不被旧客户端所接受，发现接收到的 hello 数据包不是预期数据时，通信会被拒绝。这个升级意味着未来的客户端版本能够接受即将到来的网络升级和握手通信。

这些变化有以下几项好处：

- EIP-2/1 消除了通过交易创建合约的过量激励 ,通过交易创建的成本是 21000，而通过合约创建的成本是 32000。
- EIP-2/1 在自杀式退款的帮助下修复了协议中的漏洞， 现在只用 11664 gas 就能实现简单的以太币价值转移。

- EIP-2/2 修复了交易可塑性方面的担忧（不是安全性缺陷，是用户界面不便利性）
- EIP-2/3 在合约创建过程中，建立了更加直观的"成功或失败"的区分，而不像现在"成功，失败或者空账户" 三分的情况。
- EIP-2/4 将设置时间戳区别的过量激励消除到 1，以便创建难度稍大的区块，进而保障搞定任何可能的分叉。这样就保证了出块时间维持在 10-20 范围，并且按照模拟可以恢复目标的 15 秒出块时间（现在有效时间是 17 秒）。
- EIP-7 使合约更容易储存另一个地址，作为编码和"通过"调用的可变来源，子编码会和父编码在本质上相同的环境下执行( 除非 gas 减少 ,调用栈深度增加）
- EIP-8 确保以太坊网络上使用的所有客户端软件可以应对未来网络协议升级。

## 2.5 参考资料：

Reddit 上关于 Homestead 发布的讨论：

https://www.reddit.com/r/ethereum/comments/48arax/homestead_release_faq/

初始开发路线图：

https://blog.ethereum.org/2015/03/03/ethereum-launch-process/

EIP 2：

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.mediawiki

EIP 7: DELEGATECALL：

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-7.md

对 EIP 7 的评论：https://github.com/ethereum/EIPs/issues/23

EIP 8: devp2p Forward Compatibility compliance with the Robustness Principle：

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-8.md

EIPs（Ethereum Improvement Proposals）：以太坊改进建议

# 第三章 Web 3：去中心化应用平台

## 3.1 去中心化应用平台

很多人相信像以太坊这样一个公开、无需信任的区块链平台十分适合作为 Web 3.0 的共享"后端"，像 Web3.0 这样去中心化、安全的互联网，它的核心服务，比如 DNS 和数字身份是去中心化的，个体可以参与到经济互动中。

正如以太坊开发者希望的那样，以太坊是一张空白的帆布，你可以在上面创建任何你想要的东西。以太坊协议的目的是普遍化，以使其核心特征能够以任意方式结合。理想状态下，以太坊上的数据采集和处理程序会利用以太坊区块链来建立解决方案，这些解决方案依靠去中心化的一致性提供以往无法实现的新产品和服务。

称以太坊为一个生态系统再合适不过了：核心协议由不同的基础设施、编码和社群支持，他们共同构成了以太坊项目。你也可以通过观察使用以太坊的项目来理解以太坊。现在已经有很多基于以太坊的项目已经非常引人注目了，比如 Augur，Digix，Maker 和其他很多项目（参见数据采集和处理程序）。此外，还有开发团队建立了人人皆可使用的开源组件。尽管这些组织都独立于以太坊基金之外，有各自的组织目标，但他们无疑对整个以太坊生态系统是有益的。

拓展观看/阅读：

- Vitalik Buterin – TNABC 2015：

  https://www.youtube.com/watch?v=Fjhe0MVRHO4
- Gavin Wood – DEVCON 1 – 给小白讲解以太坊：

  https://www.youtube.com/watch?v=U_LK0t_qaPo
- 以太坊伦敦聚会（详尽细节在这里）：

  https://www.youtube.com/watch?v=GJGIeSCgskc

## 3.2 智能合约

你愿意和从未谋面的人签合约吗？你会同意把钱借给埃塞俄比亚的农民吗？你愿意投资一个战乱地区由少数人管理的报纸吗？你会不嫌麻烦为了网上一次 5 美元的购买签一个有法律效力的合约吗？

大多数的答案都是否定的，原因是合约需要的基础太多了：有时需要双方之间互相信任的工作关系，有时要依靠合法的工作体系、警察和律师费用。

在以太坊这些都不需要：如果合约所必需的要求都能放在区块链中，那么就会放在区块链中，这是一个无需信任基础也几乎不用任何成本的环境。

不要想将你现有的合约转移到区块链中会麻烦，想一想那些因为经济上不可行或是没有足够法律保护而被你拒绝的成千上万的小合约吧。

## 3.3 DAO

这里简单地举个例子：想象一下你和朋友有个小生意。律师和会计很贵，完全信任让一个单独的合伙人来看管账簿可能让你精神很紧张（这甚至可能是个诈骗的机会）。你可以尝试一下多个合伙人共同看管账簿，但只要协议没有被严格遵守，就可能导致诈骗发生。

使用智能合约，公司的所有权和基金分配的条款可以在一开始就详细规定。智能合约签署的方式是，只有大部分拥有者批准，合约才可以变更。 这样的智能合约可以像开源软件一样获取，你甚至都不必雇佣自己的程序员来代替会计和律师。

这样的智能合约可以立即按比例决定分配。几个年轻人分配柠檬水站收入，可以像主权基金给拥有基金的亿万公民分配收益一样透明。在这两个案例中，这种透明性带来的花费可能每美元连一美分都不到。

# 第四章 以太坊发展历史回顾

## 4.1 以太坊历史

最近历史记录，请查看 Taylor Gerring 博客发帖。

## 4.2 诞生

2013 年末 Vitalik Buterin 第一次描述了以太坊，作为他研究比特币社群的成果，不久后，Vitalik 发表了以太坊白皮书，他在书中详细描述了以太坊协议的技术设计和基本原理，以及智能契约的结构。2014 年 1 月，Vitalik 在美国佛罗里达州迈阿密举行的北美比特币会议上正式宣布了以太坊。

与此同时，Vitalik 开始和 Gavin Wood 博士合作共同创建以太坊。2014 年 4 月， Gavin 发表了以太坊黄皮书，作为以太坊虚拟机的技术说明。按照黄皮书中的具体说明，以太坊客户端已经用 7 种编程语言实现（C++, Go, Python, Java, JavaScript, Haskell, Rust），使软件总体上更加优化。

- 以太坊发布加密货币 2.0 网络 —— 2014 年 1 月初 Coindesk 文章
- 在 bitcointalk 上宣布以太坊—— Vitalik 首次向比特币社群宣布以太坊。论坛帖子收到 5000 回复。

## 4.3 以太坊基金和以太币预售

除开发以太坊软件外，要发布新的加密货币和区块链，需要大量的辅助程序努力来组装启动和运行所需要的资源。为了快速建立一个包括开发者、矿工和其他利益相关方的大型网络，以太坊宣布了一个以太币（以太坊货币单位）预售计划。通过预售筹募基金的法律和金融复杂性导致了几个法律实体的诞生，包括 2014 年 6 月在瑞士楚格建立的以太坊基金（Stiftung Ethereum）。

从 2014 年 6 月开始，以太坊借助 42 天公开的以太币预售活动对第一批以太币进行了分配，净赚 31,591 比特币，当时价值 18,439,086 美元，交换出大约 60,102,216 以太币。销售所得首先用于偿还日益增加的法律债务，回报开发者们数月以来的努力，以及资助以太坊的持续开发。

- 启动以太币销售——在以太坊博客上第一次官方公告
- Ether.Fund 上关于预售的简明统计页面（此后停用）
- 概览：以太坊的首次公开销售 —— slacknation 博客发帖 ——关于以太币预售的所有统计数字
- 关于预售的条款声明

## 4.4 ETH/DEV 和以太坊开发

以太币预售成功之后，以太坊的开发在非营利组织 ETH DEV 的管理下走向正式化，它依据 Ethereum Suisse 的合约管理以太坊开发 ——Vitalik Buterin，Gavin Wood 和 Jeffrey Wilcke 作为组织的 3 个主管。2014 年间开发者对以太坊的兴趣持续稳定增长， ETH DEV 团队发布了一系列概念验证(PoC)供开发者社群进行评估。ETH DEV 团队在以太坊博客频繁的发帖也保持了以太坊对公众注意力的持续吸引和强劲的发展势头。以太坊论坛和以太坊 reddit 分支上渐增的访问量和用户基础证明平台正在引起一个快速增长和热衷于此事业的开发者社群的兴趣，这一趋势一直延续至今。

## 4.5 DEVCON-0

2014 年 4 月, ETH DEV 组织了 DEVCON-0 活动，世界各地的以太坊开发者聚集在柏林，讨论各种以太坊技术议题。DEVcon-0 的一些陈述和会议后来驱使以太坊向更加可靠、更安全和更加可扩展的方向发展。总体来说，这一活动激励了开发者为发布以太坊这一目标继续努力。

- DEVCON-0 演讲 youtu 播放列表
- DEVCON-0 reddit 发帖
- Gav's DEV 关于 DEVCON-0 的更新
- DEVcon-0 博客发帖概要

## 4.6 DEVgrants 项目

2015 年 4 月，DEVgrants 项目发布，这个项目为所有对以太坊平台和基于以太坊的项目所做的贡献提供基金。成百上千的开发者为以太坊项目和开源项目贡献了时间和智慧。这一项目旨在奖励和扶持开发者们所做的努力。

DEVgrants 项目时至今日仍在运行 ,项目基金情况最近一次更新是在 2016 年 1 月。

- DEVgrants 首次发布
- DEVCON-1 宣布新基金
- DEVgrants 公开 gitter room
- Youtube 上 Wendell Davis 在 DEVCON-1 关于 DEVgrants 的演讲

## 4.7 Olympic 测试网，漏洞报告奖励和安全审查

2014 年和 2015 年的开发经历了一系列概念验证发布，带来了第九届 POC 公开测试网，被称为 Olympic。开发者社群受邀检测网络极限，大量的奖励资金被分配给保持着不同记录或以某种方式成功攻破系统的人。现场发布一个月之后，官方宣布了奖项。

2015 年早期，以太坊奖励项目启动，给那些发现以太坊软件栈任何弱点的人提供 BTC 奖金。这无疑有利于以太坊的可靠性，安全性和以太坊社群技术上的自信。这一奖金项目至今仍然活跃， 并没有结束的计划。

以太坊安全审查开始于 2014 年末，持续到 2015 年上半年。以太坊请了很多第三方软件安全公司对所有协议关键的组成部分（以太坊 VM，网络，工作量证明）开展端对端审查。审查发现了很多安全问题，问题提出并再次检测后，带来了一个更安全的平台。

- Olympic 测试网预发布 —— Vitalik 的博客发文具体介绍了 olympic 奖项
- Olympic 奖项发布 —— Vitalik 博客发文具体介绍了获奖者和奖品
- 漏洞报告奖励项目启动
- 以太坊奖励项目网站
- Least Authority 审查博客发文 —— 附审查报告链接
- Deja Vu 审查博客发文

## 4.8 以太坊 Frontier 启动

以太坊 Frontier 网络于 2015 年 7 月 30 日启动，开发者开始编写智能合约和去中心化应用以部署在以太坊实时网络上。此外，矿工们开始加入以太坊网络

以帮助保障以太坊区块链的安全并从挖矿区块中赚取以太币。尽管 Frontier 的发布是以太坊项目的第一个里程碑，开发者们只试图将其作为测试版本，但结果它比任何人预期得都更有用且可靠， 开发者们立即开始建立解决方案，改进以太坊生态系统。

另请参阅：

- Vinay Gupta 最初的发布流程通告
- Frontier 要来了—— Stephan Tual 的 Frontier 发布通知
- Frontier 启动最后一步 —— 发布之后的增补发帖
- Frontier 发布带来的以太坊上线
- Frontier 网站

## 4.9 DEVCON-1

第二次开发者会议 DEVCON-1 于 2015 年 11 月初在伦敦举办。5 天的会议进行了 100 多次陈述，专题讨论会和快速讨论，吸引了 400 多名参会者，包括开发者，企业家，思考者和业务主管。所有的演讲都录了像并可免费观看。

像 UBS， IBM 和微软这样的大公司明确表示了公司对这一技术的兴趣。微软宣称将在它的新的区块链上提供以太坊以作为微软 Azure 云平台上的服务。这一公告标志着以太坊为中心的区块链技术成为主流的时刻，将和 DEVCON-1 一样被铭记。

- DEVCON-1 演讲 Yoube 播放列表

- DEVCON-1 网站陈述展示全列表，有些可链接到幻灯片

## 4.10 历史资源

简单的时间表图解

## 4.11 参考资料

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》

第 8 页 1.1.5 History of Ethereum Taylor Gerring 博客：

https://blog.ethereum.org/2016/02/09/cut-and-try-building-a-dream/

# 第五章 以太坊社区、基金会、贡献者介绍

## 5.1 社区

发起讨论和问问题，请明智选择论坛，并协助我们维护论坛环境整洁。

## 5.2 Reddit

以太坊 reddit 分论坛是最全面的以太坊论坛，这里是大部分社区讨论发生的地方和核心开发者最活跃的地方。如果你想对新闻、媒体、报道、公告、头脑风暴进行一般的讨论，选这个论坛就对了。一般来讲，这里有与更广泛社区相关的一切以太坊事件。讨论完全不收费。但这个论坛不适合寻求实际帮助或者得到迅速明确的答复（如果有以上两方面需求，可以分别用 Gitter Rooms 和 Stack Exchange）

发帖前请阅读以太坊 reddit 分支规定

更多专业 reddit 分支：

- /r/EthTrader —— 以太币交易，价格和市场
- /r/EtherMining —— 关于以太币挖矿的讨论
- /r/Ethmarket —— 个人用实物和服务交换以太币的市场
- /r/Ethinvestor —— 以太坊投资新闻和前景，以及以太坊市场发展的长期趋势
- /r/Ethereumism —— 再来一点关于主义、学派和神秘学的看法—— 以太坊超凡的一面

## 5.3 Stack Exchange

以太坊 Stack Exchange 是 StackExchange 网络问答社区的一部分。

StackExchange 是免费的问答网站，网站上所有的问题和答案都会为后人保存下来。这个网站最适合询问技术问题。可以通过回答问题来帮助以太坊同仁们，增加威望值。

## 5.4 Gitter Rooms

日常聊天可以选择 Gitter。这是开发者们的虚拟公用工作空间，如果有需要，在这里你可以迅速得到帮助和一些支持。

Gitter 使用 Github 账户，提供 Github 集成（pull 请求通知等），个人频道，Markdown 格式化等。

大部分 Gitter 频道都围绕特定的资源库或是研究管理这类一般议题组织的。请选择合适的房间，保持讨论话题的相关性。

查阅与以太坊组织相关的 gitter 房间全列表。以下是活跃的公开频道列表：

- go-ethereum —— 关于 geth（以及与 go 实现相关的工具）
- cpp-ethereum —— 关于（以及与 C++实现相关的工具）
- web3.js —— 关于 web3.js，以太坊 Java 描述语言用户操作界面库
- Solidity —— Solidity 合约相关编程语言
- serpent —— 用于合约开发的 Serpent 语言
- mist —— GUI 资料获取和处理方式浏览器，官方钱包应用

- light-client —— 关于 light 客户端和 LES 协议
- research —— 以太坊研究
- governance —— 关于开发者管理
- whisper —— 匿名数据电报发表
- swarm —— 去中心化的内容存储和分配网络
- EIPs —— 以太坊改进协议(EIPs)讨论
- ethereumjs-lib —— 以太坊核心功能的 Java 描述语言库
- devp2p —— ÐΞV's 点对点网络协议及框架

## 5.5 以太坊改进协议（EIPs）

以太坊改进协议计划旨在成为协调协议改进的框架和非正式商业流程。人们会首先向以太坊改进协议资源库提出想法作为一个问题或 pull 请求。经过基本的过滤，提议会收到一个数字并以草稿的形式发布。必须经过社区一致同意，以太坊改进协议才能变成活跃状态。提出的改变应该考虑到最终的同意取决于以太坊用户的共识。对于以太坊改进协议的讨论，可进入 gitter 关于以太坊改进协议的频道。

- 以太坊改进协议指南和样本
- 以太坊改进协议模板
- 以太坊改进协议和 README
- 以太坊改进协议讨论的 gitter 频道

## 5.6 Meetups

- Meetup 上主办的通讯录
- 以太坊论坛上的 Meetup 频道

## 5.7 作废声明

Skype

一些社区讨论会仍然在使用 skype 房间，但是我们想取消这一途径，鼓励大家用 gitter 或 slack。

## 5.8 以太坊论坛

Stephan Tual 大名鼎鼎的以太坊论坛将不再维护，可能很快就会停用。我们鼓励大家使用以上列出的推荐选择。

## 5.9 以太坊基金会

以太坊基金会是在瑞士注册的非营利性机构， 旨在管理以太币销售中筹措的基金，以更好地为以太坊和去中心化技术生态系统服务。

Stiftung Ethereum 于 2014 年 6 月在瑞士创建，它的使命是促进新技术和应用的开发 ,尤其是在新的开放的、去中心化的软件架构领域。它的目标是开发、培育、促进和维护去中心化、开放的技术。它主要但并非唯一的重心是促进以太坊协议和相关技术的开发， 以及扶持使用以太坊技术及协议的应用。此外 Stiftung Ethereum 还会通过各种方式支持去中心化的因特网。

查看网页了解更多基金会管理团队

## 5.10 以太坊基金会面向社区的门户

- Homestead 官方网站 —— 主要入口
- Reddit —— 参见社区
- 博客
- Twitter
- Youtube
- Facebook —— 不常用
- Email —— 必要的时候使用

以太坊基金会的正式照会通常在以太坊博客上以综合性发帖的形式呈现。有些发贴是技术性的，有些是组织性的，还有一些是私人的。所有的博客发贴都在 Twitter 和 Reddit 上宣布。我们的录像主要在基金会 Youtube 频道里，包括开发者会议 DEVCON0 和 DEVCON1 上的所有演讲。 欲了解社区讨论论坛，参见社区。

## 5.11 贡献者

此文本由以太坊社群共同创建，是 Homestead 文本倡议项目的一部分，项目协调人是：

- Viktor Trón ("zelig")
- Hudson Jameson ("Souptacular")

我们想感谢每一位参与者的努力：

- Ricardo de Azevedo Brandao
- Santanu Barai
- Brooks Boyd
- RJ Catalano
- Joseph Chow
- Keri Clowes
- François Deppierraz
- Bertie Dinneen
- Erik Edrosa
- Andrey Fedorov
- Rocky Fikki
- Alex Fisher
- Enrique Fynn
- Arno Gaboury
- Taylor Gerring
- Dave Hoover
- Joël Hubert
- Makoto Inoue
- Keith Irwin
- Matthias Käppler
- Bas van Kervel
- Michael Kilday

- Chandra Kumar
- Guangmian Kung
- Hugh Lang
- Yann Levreau
- Roman Mandeleil
- Kévin Maschtaler
- Andrew Mazzola
- Dominik Miszkiewicz
- John Mooney
- Chris Peel
- Craig Polley
- Colm Ragu
- Laurent Raufaste
- Christian Reitwiessner
- Josh Stark
- Scott Stevenson
- Bob Summerwill
- Alex van de Sande
- Paul Schmitzer
- Afri Schoedon
- Sudeep Singh
- Giacomo Tazzari

- Ben Tannenbaum
- Dean Alain Vernon
- Paul Worrall
- Luca Zeug
- Weiyang Zhu
- Will Zeng

以及这些匿名参与者：

- 12v
- c0d3inj3cT
- ijcoe6ru
- LucaTony
- madhancr
- mWo
- Omkara
- tflux99
- xyzether

参考资料：

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》

第 11 页 1.1.6 Community

# 第六章 以太坊客户端的选择与安装

## 6.1 选择客户端

为什么有多个以太坊客户端?

以太坊客户端与 Java 虚拟机和.NET 运行环境类似，能够让你在电脑上运行"以太坊程序"。以太坊客户端按照书面说明（黄皮书）执行，特意设计为可以彼此协作，有点儿像"商品"。

项目早期，在众多不同的操作系统中就有多个可以彼此协作的客户端实现。客户端的多样性对于整个生态系统来说是巨大的成功。它使我们能够证明协议是明确清晰的，为创新打开大门，也让我们都保持诚实。但是对终端用户来说，没有通用的"以太坊安装程序"可供他们使用，可能引起他们的困惑。

进入到 Homestead 阶段以后，Go 客户端占据了主导地位，但情况并不一直是这样 将来也并不必然如此。除了 EthereumH 其他客户端都有 Homestead 兼容的版本。下面的表格包含了最新的版本链接。

| 客户端 | 语言 | 开发者 | 最新版本 |
| --- | --- | --- | --- |
| go-ethereum | Go | 以太坊基金会 | go-ethereum-v1.4.9 |
| Parity | Rust | Ethcore | Parity-v1.2.1 |
| cpp-ethereum | C++ | 以太坊基金会 | cpp-ethereum-v1.2.9 |

| 客户端 | 语言 | 开发者 | 最新版本 |
| --- | --- | --- | --- |
| pyethapp | Python | 以太坊基金会 | pyethapp-v1.2.3 |
| ethereumjs-lib | Javascript | 以太坊基金会 | ethereumjs-lib-v3.0.0 |
| Ethereum(J) | Java | | ethereumJ-v1.3.0-RC3-daoRescue2 |
| ruby-ethereum | Ruby | Jan Xie | ruby-ethereum-v0.9.3 |
| ethereumH | Haskell | BlockApps | 尚无 Homestead 版本 |

## 6.2 安装客户端

很多"官方"客户端的开发都由以太坊基金会管理的资源资助。还有一些其他的客户端由社群或其他商业实体建立。

本章关于特定客户端的小节中可以阅读到更多有关特定客户端的内容。

## 6.3 台式机/笔记本电脑上应该安装什么？

如果你有笔记本电脑或者台式机，大概只需要安装以太坊钱包就可以了。

- 从 Github 下载最新的以太坊钱包压缩文件
- 在任意你希望的位置解锁
- 点击可执行文件（Ethereum-Wallet, Ethereum-Wallet 或 Ethereum-Wallet.app）

- 区块链数据将会被下载

以太坊钱包是 Mist 浏览器"单独的 DApp"部署方式，它将成为 Homestead 之后 Metropolis 开发的核心。Mist 附有绑定的 go-ethereum 和 cpp-ethereum 二进制。如果 Mist 开启的时候，你没有在运行命令行以太坊客户端，它就会开始运行其中一个绑定的客户端。

如果你想在命令行和以太坊互动，并且利用 JavaScript 控制台，那么你会想直接安装一个客户端软件以及 Mist。

开始的时候最适合选择 go-ethereum 和 cpp-ethereum，因为它们的开发始于项目之初，经过了安全审查，适用于所有平台，并且其维护有以太坊基金会指定资源扶持。

- 安装 cpp-ethereum，需按照安装二进制指令
- go-ethereum 只需解压已发布的二进制。

奇偶检验正很快流行起来。 当然这也取决于个人偏好。可以都试一下 :–) 要是你想挖矿，只有 Mist 是不够的。查阅挖矿章节。

## 6.4 手机/平板电脑上应该安装什么？

移动设备上的软件支持还在起步阶段。Go 团队正在发布试用的 iOS 和安卓程序库，一些开发者正在用程序库开始研究手机应用辅助程序，但是目前还没有任何可用的以太坊手机客户端（目前出现了 JAXX，支持安卓、iOS 等多个平台，译者注）。在移动设备上使用以太坊最主要的障碍是 Light 客户端支持尚

不完备。已完成的工作成果在私有分支上关闭，只在 Go 客户端上可用。Doublethinkco 将在接下来的几个月开始为 C++客户端开发 Light 客户端，接下来会有资金支持。

查看 Syng.im，它最先使用了基于 Ethereum（J）的 ethereumj-personal，但是最近跳转到了和 Light 客户端的 Geth 交叉构建。

## 6.5 单板计算机（SBC）上应该安装什么？

按照技术水平的不同，以及你想要达到的目的，可以有不同的选择。

下载一个充分准备好的安装镜像（链接到有具体下载和安装说明的页面） —— 如果你刚开始使用以太坊 AND SBC 板，诸如树莓派，那么这就是为你准备的！只需要下载你正在使用的开发板的特定安装镜像，刻录到 SD 卡上，启动设备。运行以太坊！

下载一个预编译的应用（链接到有具体下载和安装说明的页面） —— 如果你已经有 SBC 运行，并且有特定、偏好的 OS 或是想保留的设置，这是你的最佳选择！你可以只根据平台，下载合适的可执行文件，只需最少的资源库链接和 PATH 设置，就能运行以太坊！

从使用可定制描述语言的资源中创建（链接到有更多细节的页面以及单独的 SBC 链接 https://github.com/ethembedded） —— 想要运行定制的安装程序？ 我们有可以从设备的源上编译的描述语言。我们的描述语言包含自动安装依赖的软件以及客户端本身。这就使你能够安装以太坊客户端的特定版本

（比如"develop"， "master"等），编译你自己的客户端分叉版本，尝试创建程序中的各种复杂的问题找到最佳解决方案。

## 6.6 参考资料

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》第 16 页 1.2.1 Choosing a client

# 第七章 以太坊 C++客户端

以太坊 C++客户端：cpp-ethereum

## 7.1 快速入门

- 以太坊 C++客户端的 Github 项目是 webthree-umbrella。
- 我们将恢复到 cpp-ethereum-github 作为项目重启的一部分。
- 如果你只想安装二进制，直接前往安装二进制。
- 如果你想从源创建，请前往从源创建。
- 你可以在 cpp-ethereum-gitter 与社区和开发者聊天。
- 开发者在 cpp-ethereum-development-gitter 上有深度的交流。
- 请用 Github 事件跟踪器记录所有事件。
- cpp-ethereum 十分便于移植，被运用在广阔的平台上。

## 7.2 详细说明

项目正在新的领导下经历重新启动。在写的时候，我们有很多活动的部分。请对我们有一点耐心。

我们在 Homestead 简化了项目命名，尽管一些过去命名的影子还在，2016 年 5 月从 Christian 有个深入的 C++开发更新。

接下来关键的一步是即将进行的 git 库重组，这将会把我们的编码恢复到 cpp-ethereum 库。

我们也正在致力于将编码库重新许可为 Apache 2.0，这将成为一个放宽核心的长期计划的高潮。2015 年开始了一个将 cpp-ethereum 核心重新许可为 MIT 的活动 ,但一直未完成。这次是对活动的重新激活 ,尤其是看到了与 Linux 基金会 Hyperledger 项目合作的可能性。

现状（方形是应用，圆形是库）

目标重构：

## 7.3 编码的历史

C++以太坊项目在 2013 年 12 月由以太坊基金会的前 CTO GavinWood 发起。它是第二大受欢迎的客户端，远落后于同样由以太坊基金会建立的、居于主导地位的 geth 客户端。

许多原来的 C++开发者在 2015 年末和 2016 年初转移到 Slock.it 和 Ethcore 项目，紧接着 C++开发的资金支持被削减了 75%。这些资金削减是为控制基金会成本所做的一部分努力，刚好它们的发生又先于 ETH 价值最近达到的尖峰，这将基金会置于一个更健康的财务状况。

查看参与了编码工作的贡献者的完整列表

## 7.4 可移植性

以太坊 C++客户端编码十分便于移植，被成功运用在一系列不同的操作系统和设备上。 我们继续拓展范围，对 pull 请求保持开放，给额外的操作系统、编译器和设备增加了支持。

## 7.5 经验证适用的操作系统

– Alpine Linux – Arch Linux – Debian 8 (Jessie) – Fedora 20 – Fedora 21 – Fedora 22 – openSUSE Leap 42.1 – Raspbian – Sailfish OS 2.0 – Ubuntu 14.04 (Trusty) – Ubuntu 14.10 (Utopic) – Ubuntu 15.04 (Vivid) – Ubuntu 15.10 (Wily) – Ubuntu 16.04 (Xenial) – Ubuntu Touch – Ubuntu 15.04 MATE

– FreeBSD

– OS X Yosemite (10.10) – OS X El Capitan (10.11) – OS X 10.10 (Yosemite Server 4.0) – OS X 10.11 (Yosemite Server 5.0) – OS X 10.11 (Yosemite Server 5.1)

– Windows 7 – Windows 8 – Windows 8.1 – Windows 10 – Windows Server 2012 R2

## 7.6 操作系统——工作正在进行

– Maemo – MeeGo – Tizen

– iOS – tvOS – WatchOS • Android

## 7.7 经验证适用的设备

- 各种台式机和笔记本电脑设备（Windows, OS X, Desktop Linux）

– 64 位（重新组装的二进制） – 32 比特（非官方支持，但可以用）

– Linux Jolla Phone Meizu MX4 Ubuntu Edition * Nexus 5 (SailfishOS 2.0)

– Linux BeagleBone Black Odroid XU3 Project C.H.I.P. Raspberry Pi Model A Raspberry Pi Model B+ Raspberry Pi Zero Raspberry Pi 2 Raspberry Pi 3 * Wandboard Quad

## 7.8 设备——工作正在进行

– Linux * Samsung Gear S2

– BSD * Apple Watch

– Linux Nokia N9 (MeeGo) Nokia N900 (Meemo) Samsung Z1 Samsung Z3

– Android Samsung Galaxy S3 Samsung Galaxy S4

– BSD iPhone 3GS iPhone 5

– Linux Samsung RD-210 Samsung RD-PQ * Samsung TM1

– Android Samsung Galaxy Tab S 10.5 Nexus 7

– BSD * iPad Air 2

– Linux DragonBoard 410c Intel Curie Intel Edison Intel NUC * Minnowboard Max

## 7.9 二进制安装

cpp-ethereum 开发团队和更广阔的以太坊社群为各种平台发布了很多不同形式的二进制版本。本章旨在提供那些版本的完整列表。

如果你知道其他第三方所做的程序包努力， 请在 cpp-ethereum gitter 频道告诉我们，我们会添加到这个列表中。

## Ubuntu PPA (Personal Package Archive)

我们为下面的 Ubuntu 版本设置了 PPA 实例。

• Ubuntu Trusty Tahr (14.04) • Ubuntu Utopic Unicorn (14.10) • Ubuntu Vivid Vervet (15.04) • Ubuntu Wily Werewolf (15.10) • Ubuntu Xenial Xerus (16.04)

我们只支持 64 位架构。通过从源建立和禁用 VMJIT 及其他特征，也能够让客户端适用于 32 位 Ubuntu。我们可能会接受 pull 请求来增加这样的支持，但不会投入任何时间来专门开发支持 Ubuntu 32 位架构。

安装 "eth"命令行工具警告： The ethereum-qt PPA will upgrade your system-wide Qt5 installation, from 5.2 on Trusty and 5.3 on Utopic, to 5.5.

最新的稳定版本：

```
sudo add-apt-repository ppa:ethereum/ethereum-qt
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install cpp-ethereum
```

如果你想用前沿的开发者版本：

```
sudo add-apt-repository ppa:ethereum/ethereum-qt
sudo add-apt-repository ppa:ethereum/ethereum
sudo add-apt-repository ppa:ethereum/ethereum-dev
sudo apt-get update
sudo apt-get install cpp-ethereum
```

## 安装 Mix IDE Mix IDE

装载在 Ubuntu 上作为开发者 PPA（见上）。直接按照以上步骤操作，然后再操作：

```
sudo apt-get install mix-ide
mix-ide
```

## Windows 安装程序

我们对每个版本 都生成了 Windows 安装程序。

可以在 Windows 7, Windows 8/8.1, Windows 10 and Windows Server 2012 R2 上运行，尽管我们的自动编译是基于 Windows 8.1 主机。

如果发生运行时错误，报告丢失 msvcr120.dll 或 msvcp120.dll 文件，请从 Microsoft 安装 Visual C++ Redistributable Packages for Visual Studio 2013。

我们只支持 64 位架构。

通过从源建立和禁用 VMJIT 及其他特征，也能够让客户端适用于 32 位 Windows。我们可能会接受 pull 请求来增加这样的支持，但不会投入任何时间来专门开发支持 Windows 32 位架构。

大部分使用 Windows 的个体现在都有 64 位硬件。

## Windows Chocolatey NuGet 程序包

尽管以前做过，但是写这篇文件的时候我们没有再生成 Chocolatey 程序包。

给不熟悉这个技术的人解释，它的本质是 Windows 的 apt-get—— 一个全球性的无声的工具安装程序。

我们想在不久的将来再次支持 Chocolatey，和我们在 OS X 上支持 Homebrew 以及给 Ubuntu 安装 PPA 的原因一样。对于有技术能力的用户，这样操作命令行会很方便：

```
choco install cpp-ethereum

choco update cpp-ethereum
```

## OS X DMG（磁盘映像）

我们为每个版本都生成了 OS X 磁盘映像。我们只支持最新的两个 OS X 版本：

• OS X Yosemite (10.10) • OS X El Capitan (10.11)

我们只支持 64-比特架构。

如果你的系统这两个 OS X 版本都不支持，那么很抱歉你不走运啦！

## OS X Homebrew 程序包

我们在自动架构系统里生成了 Homebrew 程序包。 我们只支持最新的两个 OS X 版本： • OS X Yosemite (10.10) • OS X El Capitan (10.11)

我们只支持 64-比特架构。 如果你的系统这两个 OS X 版本都不支持，那么很抱歉你不走运啦！

所有的 OS X 架构都需要你事先安装 Homebrew 程序包管理器。如果你想从头开始，这是卸载 Homebrew 的方法。

要从 Homebrew 安装以太坊 C++组件，请执行以下指令：

```
brew update
brew upgrade
brew tap ethereum/ethereum
brew install cpp-ethereum
brew linkapps cpp-ethereum
```

或者……如果你也想创建 AlethZero 和 Mix IDE，请执行：

```
brew install cpp-ethereum --with-gui
```

要开启应用，在终端窗口输入其中一个指令：

```
open /Applications/AlethZero.app
open /Applications/Mix.app
eth
```

这是 Homebrew 公式，详细描述了所有支持的命令行选项。

## Raspberry Pi, Odroid, BeagleBone Black, Wandboard

EthEmbedded 的 John Gerryts 在主要的里程碑为各种 SBC 创建了二进制镜像，此外为这些设备检测和维护架构描述语言。EthEmbedded 是 2015 年 5 月的 devgrant 接收者。他给 eth 和 geth 都创建了二进制。

这是来自 EthEmbedded 的 Homestead 二进制。

## 手机的 Linux ARM 交叉结构，可穿戴设备, SBCs

doublethinkco 的 Bob Summerwill 交叉架构了 ARM 二进制 ,对很多种类的硬件都有效 ,从手机到可穿戴的 Linux 发行版( Sailfish OS, Tizen OS, Ubuntu Touch )到 EthEmbedded 针对的同样的 SBC。Doublethinkco 是 2016 年 2 月的 BlockGrantX 接收者。

查看 cpp-ethereum-cross README 文件，了解平台和已知状态的全矩阵。

这是来自 doublethinkco 的交叉建构二进制：已发布——Homestead 的交叉建构 eth 二进制

## ArchLinux 用户库 （AUR）

Arch Linux 程序包是 Afri Schoedon 维护的社群。

在 aur.archlinux.org 上查看以下程序包。

- ethereum （稳定，最新版本）
- ethereum-git （不稳定，最新版本）

要创建和安装这个程序包，按照 AUR 安装程序包说明：

- 获取含有 PKGBUILD 的原始码
- 提取原始码
- 作为简单用户运行所储存文件目录里的 makepkg –sri
- 作为超级用户用 pacman –U 安装生成的程序包

你也可以用 AUR 助手比如 yaourt 或 pacaur 直接在你的系统里安装程序包。

## 7.10 参考资料

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》

第 18 页 1.2.3 cpp-ethereum

# 第八章 以太坊 Go、Java 等客户端介绍

## 8.1 go-ethereum

go-ethereum 客户端通常被称为 geth，它是个命令行界面，执行在 Go 上实现的完整以太坊节点。通过安装和运行 geth，可以参与到以太坊前台实时网络并进行以下操作：

- 挖掘真的以太币
- 在不同地址间转移资金
- 创建合约，发送交易
- 探索区块历史
- 及很多其他

链接：

- 网站: http://ethereum.github.io/go-ethereum/
- Github: https://github.com/ethereum/go-ethereum
- 维基百科: https://github.com/ethereum/go-ethereum/wiki/geth
- Gitter: https://gitter.im/ethereum/go-ethereum

## 8.2 pyethapp

Pyethapp 是以 python 为基础的客户端 ,实现以太坊加密经济状态机。python 实现旨在提供一个更容易删节和扩展的代码库。 Pyethapp 利用两个以太坊核心组成部分来实现客户端：

- pyethereum —— 核心库，以区块链、以太坊模拟机和挖矿为特征
- pydevp2p —— 点对点网络库，以节点发现和运输多码复用和加密连接为特征

链接：

- Github: https://github.com/ethereum/pyethapp
- 维基百科: https://github.com/ethereum/pyethapp/wiki/Getting-Started
- Gitter 聊天: https://gitter.im/ethereum/pyethapp

## 8.3 ethereumjs-lib

正如黄皮书中所说 ,ethereumjs-lib 是核心以太坊功能的 javascript 库。这是个简单的元模块 ,提供以下模块。大部分 JS 模块都在 ethereumjs 上有跟踪。

- 虚拟机 – 以太坊虚拟机和状态处理功能
- 区块链 – 区块链管理
- 区块 – 区块模式定义和验证
- 交易 – 交易模式定义和验证
- 账户 – 账户模式定义和验证
- rlp – 循环长度前缀序列化
- Trie – 改良的 Merkle Patricia 树

- Ethash – 以太坊工作量证明算法
- utils – 多样辅助功能
- devp2p – 网络协议
- devp2p-dpt – 有争议的对等端表

链接：

- Github: https://github.com/ethereumjs/ethereumjs-lib
- 加入 Gitter 聊天: https://gitter.im/ethereum/ethereumjs-lib

Ethereum(J)

Ethereum(J) 是以太坊协议的纯 Java 实现。它作为可以嵌入任何 Java/Scala 项目的库提供，并为以太坊协议及附属服务提供完全支持。Ethereum(J)最开始由 Roman Mandeleil 开发，现在受 资助。

Ethereum(J)支持 CPU 挖矿。目前它由纯 Java 实现 ,可用于私人和测试网络。你甚至可以在实时以太坊网络上挖矿，但是这样从经济角度来说不划算。

链接：

- 博客: http://ethereumj.io/
- Github: https://github.com/ethereum/ethereumj
- Gitter 聊天: https://gitter.im/ethereum/ethereumj

## 8.4 ethereumH

这个程序包提供了写在 Haskell 上的工具，能使你连接到以太坊区块链。

链接：

Github: https://github.com/blockapps/ethereumH BlockApps: http://www.blockapps.net/

## 8.5 Parity

Parity 声称是世界上最快速最轻便的客户端。它用 Rust 语言写成，可靠性、性能和代码清晰度都有所增强。Parity 由 Ethcore 开发。Ethcore 由以太坊基金会的几个会员创建。

- 网站: https://ethcore.io/parity.html
- Github: https://github.com/ethcore/parity
- Gitter 聊天: https://gitter.im/ethcore/parity

Arch Linux 程序包由 Afri Schoedon 和 quininer 进行社群维护。

https://aur.archlinux.org/packages/parity/（稳定，最新版本）

https://aur.archlinux.org/packages/parity-git/（不稳定，最新开发）

已经有人报告在树莓派 2 上成功运行了 Parity。

## 8.6 ruby-ethereum

ruby-ethereum 是以太坊虚拟机上的一个实现，用 Ruby 语言写成。

链接：

- Github: https://github.com/janx/ruby-ethereum
- Gem: https://rubygems.org/gems/ruby-ethereum

相关：

- ruby-serpent: 捆绑在以太坊 Serpent 编译器上的 Ruby 语言.
- ethereum-ruby: 一个 pure-Ruby JSON-RPC 包装，用于和以太坊节点交流。要使用这个库，你需要有运行的以太坊节点和可行的 IPC 支持( 默认 )。目前支持 go-ethereum 客户端。

## 8.7 参考资料

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》第 41 页 1.2.4 go-ethereum

# 第九章 以太坊账户管理

## 9.1 账户

账户在以太坊中发挥着中心作用。共有两种账户类型：外部账户（EOAs）和合约账户。我们这里重点讲一下外部账户，以下会简称为账户。合约账户简称为合约， 在合约章节具体讨论。把外部账户和合约账户都归入到帐户的一般概念是合理的，因为这些实体都是所谓的状态对象。这些实体都有状态：账户有余额,合约既有余额也有合约储存。所有账户的状态正是以太坊网络的状态，以太坊网络和每个区块一起更新，网络需要达成关于以太坊的共识。对于用户通过交易和以太坊区块链互动来说，账户是必不可少的。

如果我们把以太坊限制为只有外部账户，只允许外部账户之间进行交易，我们就会进入到"代币"系统，"代币"系统不如比特币本身有力，只能用于转移以太币。

账户代表着外部代理人（例如人物角色，挖矿节点 ，或是自动代理人）的身份。账户运用公钥加密图像来签署交易以便以太坊虚拟机可以安全地验证交易发送者身份。

## 9.2 钥匙文件

每个账户都由一对钥匙定义，一个私钥和一个公钥。 账户以地址为索引，地址由公钥衍生而来，取公钥的最后 20 个字节。每对私钥 /地址都编码在一个钥匙文件里。钥匙文件是 JSON 文本文件 ,可以用任何文本编辑器打开和浏览。

钥匙文件的关键部分，账户私钥，通常用你创建帐户时设置的密码进行加密。钥匙文件可以在以太坊节点数据目录的 keystore 子目录下找到。确保经常给钥匙文件备份！查看备份和恢复账号章节了解更多。创建钥匙和创建帐户是一样的。

- 不必告诉任何人你的操作。
- 不必和区块链同步。
- 不必运行客户端。
- 甚至不必连接到网络。

当然新账户不包含任何以太币。但它将会是你的，你大可放心，没有你的钥匙和密码，没有人能进入。

转换整个目录或任何以太坊节点之间的个人钥匙文件都是安全的。

警告：请注意万一你从一个不同的节点向另一个节点添加钥匙文件， 账户的顺序可能发生改变。确保不要回复或改变手稿中的索引或代码片段。

## 9.3 创建账号

警告：记住密码并"备份钥匙文件"。为了从账号发送交易，包括发送以太币，你必须同时有钥匙文件和密码。确保钥匙文件有个备份并牢记密码，尽可能安全地存储它们。这里没有逃亡路径，如果钥匙文件丢失或忘记密码，就会丢失所有的以太币。没有密码不可能进入账号，也没有忘记密码选项。所以一定不要忘记密码。

# 9.4 使用 geth account new

一旦安装了 geth 客户端，创建账号就只是在终端执行 geth account new 指令的问题了。

注意不必运行 geth 客户端或者和区块链同步来使用 geth account 指令。

```
$ geth account new

Your new account is locked with a password. Please give a password. Do not
forget this password.
Passphrase:
Repeat Passphrase:
Address: {168bc315a2ee09042d83d7c5811b533620531f67}
```

对于非交互式使用 ,你可以提供纯文本密码文件作为—password 标志的变元。文件中的数据包含密码的原始字节，后面可选择单独跟着新的一行。

```
$ geth --password /path/to/password account new
```

警告：用—password 标志只是为了测试或在信任的环境中自动操作。不建议将密码保存在文件中或以任何其他方式暴露。如果你用密码文件来使用—password 标志 ,要确保文件只对你自己可阅读和列表。你可以在 Mac/Linux 系统中通过以下指令实现：

```
touch /path/to/password
chmod 600 /path/to/password
cat > /path/to/password
>I type my pass
```

要列出目前在你的 keystore 文件夹中的钥匙文件的所有账号，使用 geth account 指令的 list 子指令：

```
$ geth account list

account #0: {a94f5374fce5edbc8e2a8697c15331677e6ebf0b}
account #1: {c385233b188811c9f355d4caec14df86d6248235}
```

```
account #2: {7f444580bfef4b9bc7e14eb7fb2a029336b07c9d}
```

钥匙文件的文件名格式为 UTC——。账号列出时是按字母顺序排列，但是由于时间戳格式，实际上它是按创建顺序排列。

# 9.5 使用 geth 控制台

为了用 geth 创建新账号，我们必须先在控制台模式开启 geth（或者可以用 geth attach 将控制台依附在已经运行着的事例上）：

```
> geth console 2>> file_to_log_output
instance: Geth/v1.4.0-unstable/linux/go1.5.1
coinbase: coinbase: [object Object]
at block: 865174 (Mon, 18 Jan 2016 02:58:53 GMT)
datadir: /home/USERNAME/.ethereum
```

控制台使你能够通过发出指令与本地节点互相作用。比如，试一下这个列出账号的指令：

```
> eth.accounts
{
code: -32000,
message: "no keys in store"
}
```

这就表明你没有账号。你也可以从控制台创建一个账号：

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xb2f69ddf70297958e582a0cc98bce43294f1007d"
```

注意：记得用一个安全性强、随机生成的密码。

我们刚刚创建了第一个账号。如果我们再次试着列出账号，就可以看到新创建的账号了。

```
> eth.accounts
["0xb2f69ddf70297958e582a0cc98bce43294f1007d"]
```

# 9.6 使用 Mist 以太坊钱包

对于相反的命令行，现在有一个基于 GUI 的选项可以用来创建账号："官方"Mist 以太坊钱包。 Mist 以太坊钱包，和它的父项目 Mist, 是在以太坊基金会的赞助下开发，因此是"官方"地位。钱包应用有 Linux, Mac OS X 和 Windows 可用的版本。

警告：Mist 钱包是试用软件，使用需风险自担。

用 GUI Mist 以太坊钱包创建账号再容易不过了。事实上，第一个账号在应用安装期间就创建出来了。

- 一、根据你的操作程序下载钱包应用最新版本。由于你实际上会运行一个完整的 geth 节点，打开钱包应用就会开始同步复制你电脑上的整个以太坊区块链。
- 二、 解锁下载的文件夹，运行以太坊钱包可执行文件。
- 三、 等待区块链完全同步，按照屏幕上的说明操作，第一个账号就创建出来了。
- 四、第一次登录 Mist 以太坊钱包，你会看到自己在安装过程中创建的账号。它会被默认命名为主账号（以太库）
- 五、再另外创建账号很容易；只需点击应用主界面上的添加账号，输入所需的密码即可。

注意：Mist 钱包仍在开发中 ,以上列出的具体步骤可能会随着更新有所变更。

# 第十章 创建安全多签名钱包及高级设置

## 10.1 在 Mist 创建多签名钱包

Mist 以太坊钱包有个选项是可以用多签名钱包使钱包里的余额更安全。用多签名钱包的好处是它需要多个账号共同批准才能够从余额中提取大额资金。创建多签名钱包之前，需要创建多个账号。

在 Mist 创建账号文件很容易。在"账号"菜单下点击"添加账号"。选择一个安全性高又容易记住的密码（记住没有密码找回选项），确认，账号创建就完成了。创建至少 2 个账号。如果你愿意，第二个账号可以在另一台有 Mist 运行的电脑上创建（理论上这样可以使多签名更加安全）。你只需要第二个账号的公钥（存款地址）来创建多签名钱包（复制 /粘贴，不要手动输入）。因为需要第一个账号来创建多签名钱包合约，所以第一个账号必须是在你创建多签名钱包的电脑上。

既然已经创建了账号，保持安全并进行备份（如果不备份，电脑系统崩溃，余额就会丢失）。点击菜单顶端的"备份"。选择"keystore"文件夹，反击 /选择"复制"（不要选择"剪切"，否则结果会很糟糕）。回到桌面，在空白区域反击，选择"粘贴"。你可能会想把这个"keystore"文件夹重命名为"以太坊 - keystore - 备份 - 年 - 月 - 日"，这样以后就能很快辨认出来。这时候你就能把文件夹内容添加到压缩文件里（如果是在线备份，最好用另外一个安全性高又容易记住的密码对档案进行密码保护 ），复制到 U 盘，刻录到 CD/DVD ，或者上传到在线存储设备（ Dropbox/Google Drive 等）。

你现在应该添加大约不到 0.02 以太币到第一个账号里（那个用来创建多签名钱包的账号）。这是创建多签名钱包所需的交易费用。另外再需要 1 以太币（或者更多），因为 Mist 现在需要这样做来确保钱包合约交易有足够的"gas"来正常执行……所以对新人来说，总共需要不到 1.02 以太币。

创建多签名钱包的时候，你会进入到附属在它上面所有账号的完整地址。我推荐把每个地址复制 /粘贴到简单的文本编辑器上（notepad/kedit 等），到 Mist 每个账号的详情页以后，从右侧按键栏里选择"复制地址"选项。不要手动输入地址，或者冒着输入错误的风险，你可能会把交易发送到错误的地址，因此丢失余额。

我们现在准备好了创建多签名钱包。在"钱包合约"下，选择"增加钱包合约"。起个名字，选择第一个账号持有人，选择"多签名钱包合约"。你会看到出现这样的文字：

"这是由 X 个持有人共同控制的联合账号。每天最多可以发送 X 个以太币。任何超过每日限额的交易都需要 X 个持有人确认。"

设置附属在这个多签名钱包上的持有人(账号)数量，每日提款限额(这只要求一个账号提出这些钱款，以及允许多少持有人(账号)批准超过每日限额的提款。

现在加入之前复制/粘贴在文本编辑器中的账号地址，确认所有的设置正确后，点击底部的"创建"按钮。然后需要输入密码发送交易。在"钱包合约"部分，会显示出新的钱包，告诉你"创建"。

钱包创建完成后，就能在屏幕上看到合约地址。选择整个地址，复制/粘贴到文本编辑器的新文件里，保存至桌面，命名为"以太坊-钱包-地址.txt"或其他名称。

现在只需用备份合约文件的方式来备份"以太坊-钱包-地址.txt" ,接着就能用这个地址在 ETH 装载新的多签名钱包。

如果你要从备份中恢复，只需要复制"以太坊 – keystore – 备份"文件夹里的文件到这个攻略第一部分里提到的"keystore"文件夹。如果是在从未安装过 Mist 的机器上安装（第一次创建账号的同时就会建立文件夹），可能需要创建"keystore"文件夹。如果要恢复多签名钱包，不要像我们创建之前一样选择"多签名钱包合约"，只选择"导入钱包"就可以了。

故障排查：

- Mist 不能同步。一个有用的解决方案是将个人电脑硬件时钟与 NTP 服务器同步，确保时间无误后重启。
- Mist 同步后启动，但出现了白屏。有可能是因为你在基于 Linux 的操作系统上运行了"xorg" 视频驱动器（Ubuntu, Linux Mint 等），试试安装制造商的视频驱动器。
- 提示"密码错误"。在现在的 Mist 版本上，这有可能是个错误的提示。重启 Mist，问题就能解决（如果你输入的确实是正确密码）。

## 10.2 使用 Eth

与使用 geth 的、可用的钥匙管理相关的每个选项都同样适用于 eth。以下是与"账号"有关的选项：

```
> eth account list // List all keys available in wallet.
> eth account new // Create a new key and add it to the wallet.
> eth account update [<uuid>|<address> , ... ] // Decrypt and re-encrypt given
keys.
> eth account import [<uuid>|<file>|<secret-hex>] // Import keys from given
source and place in wallet.
```

以下是与"钱包"有关的选项：

```
> eth wallet import <file> //Import a presale wallet.
```

注意："账号导入"选项只能用于导入一般的钥匙文件。"钱包导入"选项只能用于导入预售钱包。

也可以从综合控制台进入钥匙管理（用内置控制台或者 geth 附件）：

```
> web3.personal
{
listAccounts: [],
getListAccounts: function(callback),
lockAccount: function(),
newAccount: function(),
unlockAccount: function()
}
```

# 10.3 使用 EthKey （不推荐使用）

Ethkey 是 C++实现的 CLI 工具，可以让你和以太坊钱包互动。你可以用它罗列、检查、创建、删除和修改钥匙，以及检查、创建和签署交易。我们假定你还没有运行过客户端 ,比如 eth 或者 Aleth 系列的任何客户端。如果你运行过，可以略过这一章节。要创建钱包，用 creatwallet 指令运行 ethkey：

```
> ethkey createwallet
```

请输入管理员密码来保护 keystore（设一个安全性高的！）：会问你要一个"管理员"密码。这能保护你的隐私，并且它会默认为你任何钥匙的密码。你需要再次输入同一文本来进行确认。

注意：使用安全性高的、随机生成的密码。

我们可以通过使用列表指令简单地列出钱包内的钥匙：

```
> ethkey list
No keys found.
```

我们还没创建任何钥匙，它也是这样告诉我们的！我们来创建一个吧。 要创建钥匙，我们需要用 new 指令，需要通过一个名字——这也是我们要给钱包里账号的名字。我们称之为"测试"：

```
> ethkey new test
```

输入密码来保护这个账号（或者用管理员密码就不用输入了）。这会促使你输入密码来保护这个钥匙。如果你只点击回车，就会使用默认的"管理员"密码。这意味着，当你想用账号的时候，不必输入钥匙密码（因为它记住了管理员密码）。总体来说，你应该试着为每个钥匙设置一个不同的密码，因为这样能防止一个密码被盗用而导致其他账号也被入侵。然而为了方便你可能会决定让低安全性的账号使用同一个密码。

在这里，我们用一个极富想象力的密码 123（永远不要用这么简单的密码，除非是暂时的测试账号）。输入密码后，它就会让你再次输入确认。再次输入 123。由于你设置了它的密码，它会让你提供一个密码提示，每次进入的时候都会显示密码提示。提示会储存在钱包里，由管理员密码保护。我们来输入糟糕的密码提示 321 倒序。

```
> ethkey new test
Enter a passphrase with which to secure this account (or nothing to use the
master passphrase):
Please confirm the passphrase by entering it again:
Enter a hint to help you remember this passphrase: 321 backwards
Created key 055dde03-47ff-dded-8950-0fe39b1fa101
```

```
Name: test
Password hint: 321 backwards
ICAP: XE472EVKU3CGMJF2YQ0J9RO1Y90BC0LDFZ
Raw hex: 0092e965928626f8880629cec353d3fd7ca5974f
```

所有正常（或者说直接）的 ICAP 地址都以 XE 开头，这样就很容易辨认。请注意这个钥匙在创建的钥匙后有另外一个标识符，被称为 UUID。这是个特有的钥匙标识符，和账号本身毫无关系。知道了它对攻击者发现你在网上的身份毫无帮助。它刚好也是钥匙的文件名 ,你可以在~/.web3/keys (Mac 或 Linux) 或者$HOME/AppData/Web3/keys (Windows)中发现。让我们通过列出钱包里的钥匙来确认它在正常运行：

```
> ethkey list
055dde03-47ff-dded-8950-0fe39b1fa101 0092e965...
XE472EVKU3CGMJF2YQ0J9RO1Y90BC0LDFZ test
```

它每行会报告一个钥匙（这里总共只有一个钥匙）。在这个例子里，钥匙被储存在 055dde 文件，有个以 XE472EVK 开头的 ICAP 地址。这不容易记住，所以有个专有名称会很有帮助，还是叫 test 吧。

# 第十一章 钱包导入与账号管理

## 11.1 导入预售钱包

### 11.1.1 使用 Mist 以太坊钱包

用 GUI Mist 以太坊钱包导入预售钱包非常简便。实际上，在应用安装期间你会被问到是否要导入预售钱包。

警告：Mist 钱包是试用软件。使用风险自担。

安装 Mist 以太坊钱包的说明在 创建账号：使用以太坊钱包 章节给出。

只需要把.json 预售钱包文件夹拖放到指定区域，输入密码，导入预售钱包。

如果你选择不在应用安装期间导入预售钱包，以后你可以随时导入，只需选择应用菜单栏下方的账号菜单，然后选择导入预售账号。

注意：Mist 钱包仍在开发中，以上列出的具体步骤可能会随着更新有所变更。

### 11.1.2 使用 geth

如果你单独安装 get，导入预售钱包可以通过在终端执行以下操作完成：

```
geth wallet import /path/to/my/presale-wallet.json
```

会提示你输入密码。

## 11.2 更新账号

你可以把钥匙文件更新到最新的钥匙文件格式并且/或者升级钥匙文件密码。

使用 geth

你可以在命令行用更新子命令更新现在的账号，可以使用账号地址或者索引作为参数。记住账号索引反映了创建顺序（按字母顺序排列的钥匙文件名包含了创建时间）。

```
geth account update b0047c606f3af7392e073ed13253f8f4710b08b6
```

或者

```
geth account update 2
```

例如：

```
$ geth account update a94f5374fce5edbc8e2a8697c15331677e6ebf0b

Unlocking account a94f5374fce5edbc8e2a8697c15331677e6ebf0b | Attempt 1/3
Passphrase:
0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b
account 'a94f5374fce5edbc8e2a8697c15331677e6ebf0b' unlocked.
Please give a new password. Do not forget this password.
Passphrase:
Repeat Passphrase:
0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b
```

账户以加密的形式储存在最新版本，它会提示你需要一个密码来解锁账户，另一个密码来保存更新的文件。同一个指令还可以用在将弃用格式的账户变成最新版本或者改变账户密码。

对于非交互式使用，密码可以用 —password 标志详细说明：

```
geth --password <passwordfile> account update
a94f5374fce5edbc8e2a8697c15331677e6ebf0bs
```

由于只能给出一个密码，所以只能执行格式更新，修改密码只在交互式的情况下才有可能。

注意：账号更新有个副作用就是会引起账号顺序变化。更新成功后，同一钥匙所有之前的格式/版本都会被移除！

# 11.2 账号备份和恢复

## 11.2.1 手动备份/恢复

要从账号发送交易，需要有账号钥匙文件。钥匙文件可以在以太坊节点数据目录的钥匙商店（keystore）子目录下找到。默认数据目录的位置与平台相关：

- Windows: C:Usersusername%appdata%RoamingEthereumkeystore
- Linux: ~/.ethereum/keystore
- Mac: ~/Library/Ethereum/keystore

要备份钥匙文件（账号），在 keystore 子目录中复制单独的钥匙文件或复制整个 keystore 文件夹。

要恢复钥匙文件（账号），将钥匙文件重新复制到 keystore 子目录，即其原始地址。

## 11.2.2 导入未加密私钥

导入未加密私钥由 geth 支持

```
geth account import /path/to/<keyfile>
```

这个指令从纯文本文件导入未加密私钥并创建新账号和打印地址。钥匙文件被假定包含未加密私钥作为编码到十六进制的标准 EC 原始字节。账号以加密的形式储存，会提示你输入密码。你需要记住密码用于以后解锁账号。

下面给出一个例子，详细说明数据目录。如果 —datadir 标志没有使用，新账户就会被创建在默认数据目录里，例如钥匙文件会被放在数据目录的钥匙文件子目录里。

```
$ geth --datadir /someOtherEthDataDir account import ./key.prv

The new account will be encrypted with a passphrase.
Please enter a passphrase now.
Passphrase:
Repeat Passphrase:
Address: {7f444580bfef4b9bc7e14eb7fb2a029336b07c9d}
```

对于非交互式使用，密码可以用 —password 标志详细说明：

```
geth --password <passwordfile> account import <keyfile>
```

注意：因为你可以直接把加密账户复制到另一个以太坊事例中，在节点之间转移账号的时候就不需要这个导入/导出机制了。

警告：当你往已存在节点的 keystore 里复制钥匙的时候，你习惯的账户顺序可能会改变。因此要保证你不依赖于账户顺序，否则就要进行复核并更新脚本中使用的索引。

# 第十二章 什么是以太币？如何获取？

## 12.1 以太币是什么？

以太币是以太坊中使用的货币名称，用于在以太坊虚拟机内支付计算。这通过为了以太币购买 gas 间接实现，在 gas 中有所解释。

## 12.2 面额

以太坊有一个面额的度量体系，用作以太币单位。每个面额都有自己独特的名字（有的是在计算机科学与加密经济学演进过程中发挥开创性作用的人物的姓）。最小的面额也就是以太币基础单位，叫做 Wei。下面的列表是面额名称以及 Wei 的价值转换。以太币遵循惯例（尽管有些模棱两可），指定了货币单位（1e18 或者百万的三次方 Wei）。注意很多人误以为货币叫以太坊，但以太坊并不是货币单位。

单位换算列表： 参考《Ethereum Homestead Documentation》第 52 页 1.4.1 What is ether?

| 单位 | Wei 值 | Wei |
| --- | --- | --- |
| wei | 1 wei | 1 |
| Kwei(babbage) | 1e3 wei | 1,000 |

| 单位 | Wei 值 | Wei |
| --- | --- | --- |
| Mwei(lovelace) | 1e6 wei | 1,000,000 |
| Gwei(shannon) | 1e9 wei | 1,000,000,000 |
| microether(szabo) | 1e12 wei | 1,000,000,000,000 |
| milliether(finney) | 1e15 wei | 1,000,000,000,000,000 |
| ether | 1e18 wei | 1,000,000,000,000,000,000 |

## 12.3 以太币供应

- https://blog.ethereum.org/2014/04/10/the-issuance-model-in-ethereum/
- https://www.reddit.com/r/ethereum/comments/44zy88/clarification_on_ether_supply_and_cost_of_gas/
- https://www.reddit.com/r/ethereum/comments/45vj4g/question_about_scarcity_of_ethereum_and_its/
- https://www.reddit.com/r/ethtrader/comments/48yqg6/is_there_a_cap_like_with_btc_with_how_many_ether/

## 12.4 获取 ether

要想获得以太币，你需要

- 成为以太坊矿工（参考挖矿）或者
- 用中心化或无需信任的服务，将其他货币兑换为以太币
- 使用用户友好的 Mist 以太坊 GUI 钱包，从 Beta 6 可以用 API 购买以太币。

## 12.5 无需信任的服务

注意以太坊平台的特殊性在于智能合约使无需信任的服务成为可能，免除了货币兑换交易中所需的信任第三方，比如作非居间化投资货币兑换业务。 这样的项目（写此文本时有 alpha/prelaunch）有以下几个特点：

- BTCrelay
  - 更多信息（关于 ETH/BTC 双向固定，不需要修正比特币代码）
  - BTCrelay 审核
- EtherEx 去中心化兑换。

## 12.6 中心化兑换市场列表

参考《Ethereum Homestead Documentation》第 53 页 1.4.3 Getting ether

| 兑换 | 货币 |
| --- | --- |
| Poloniex | BTC |
| Kraken | BTC, USD, EUR, CAD, GBP |

| 兑换 | 货币 |
| --- | --- |
| Gatecoin | BTC, EUR |
| Bitfinex | BTC, USD |
| Bittrex | BTC |
| Bluetrade | BTC, LTC, DOGE |
| HitBTC | BTC |
| Livecoin | BTC |
| Coinsquare | BTC |
| Bittylicious | GBP |
| BTER | CNY |
| Yunbi | CNY |
| Metaexchange | BTC |

# 12.7 中心化固定汇率兑换

| 兑换 | 货币 |
| --- | --- |
| Shapeshift | BTC, LTC, DOGE,其他 |
| Bity | BTC, USD, EUR, CHF |

# 12.8 贸易和价格分析

- 根据 coinmarketcap 上的量列出的 ETH 市场详尽清单
- 主要 ETH 市场的实时数据合计：
  - Tradeblock
  - EthereumWisdom
  - Cryptocompare
  - Coinmarketcap

# 第十三章 钱包、以太币、Gas 介绍

## 13.1 在线钱包，纸钱包和离线存储

这只是链接和说明的集散地。请将它从列表模式转换到生态系统。这边举的例子，可能会解释偏执的实践，列出潜在风险。

- Mist 以太坊钱包
  - 下载版本
  - Mist 以太坊钱包开发者预览-基础博客发文
  - 如何简单地设置以太坊 Mist 钱包！- Tommy Economics 教程
- Kryptokit Jaxx
  - Jaxx 主网站
  - 手机版本
- Etherwall
  - Etherwall 网站
  - Etherwall 源
- MyEtherWallet
  - MyEtherWallet 网站
  - MyEtherWallet 源
  - Chrome 延伸
- 离线存储

- ConsenSys 的冰箱 – 基于综合了光源钱包和 HD 钱包库的离线存储
- Reddit discussion 1
- 如何设置离线存储钱包

- Hardware 钱包

- reddit discussion 2
- reddit discussion 3

- Brain 钱包

- brain 钱包不安全，请不要使用。

  https://www.reddit.com/r/ethereum/comments/45y8m7/brain_wallets_are_now_generally_shunned_by/
- 对 brain 钱包要特别谨慎，请参考近期的争论：

  <https://reddit.com/r/ethereum/comments/43fhb5/brainwallets >

  vs http://blog.ether.camp/post/138376049438/why-brain-wallet-is-the-best

- Misc

- Kraken 钱包清理工具– 预售钱包导入
- 安全储存以太币的推荐方法
- 如何购买和储存以太币
- 向外行人介绍蛮力以及不要用 brain 钱包的原因
- Pyethsaletool
- 账号和钱包

## 13.2 发送以太币

以太坊钱包支持通过图像界面发送以太币。

以太币也可以用 geth 控制台转换。

```
> var sender = eth.accounts[0];
> var receiver = eth.accounts[1];
> var amount = web3.toWei(0.01, "ether")
> eth.sendTransaction({from:sender, to:receiver, value: amount})
```

了解更多以太币转移交易，请参考账户类型，Gas 和交易。

以太坊在加密货币领域是独一无二的，原因在于以太币作为加密燃料具有实用价值，通常被称作"gas"。除交易费用外，gas 是每个网络请求的中心部分，需要发送者为消费的运算资源付费。Gas 成本是基于请求规模和复杂性的动态计算，乘以现在的 gas 价格。它作为加密燃料的价值能够增加以太币和以太坊作为一个整体的稳定性和长期需求。了解更多信息，请参考账户类型，Gas 和交易。

## 13.3 Gas 和以太币

- https://www.reddit.com/r/ethereum/comments/271qdz/can_someone_explain_the_concept_of_gas_in_ethereum/
- https://www.reddit.com/r/ethereum/comments/3fnpr1/can_someone_possibly_explain_the_concept_of/
- https://www.reddit.com/r/ethereum/comments/49gol3/can_ether_be_used_as_a_currency_eli5_ether_gas/

Gas 被认为是网络资源/使用的固定成本。想要发送交易的真实成本保持一致，所以不希望 gas 发行，一般来说货币是不稳定的。因此我们发行以太币，它的价值会变动，但也会根据以太币来执行 gas 价格换算。如果以太币价格上升，gas 价格换算成以太币会下降，以此来保持 gas 的真实花费相同。 gas 有多个相关词汇：gas 成本，gas 限制和 gas 费用。gas 背后遵循的原则是使以太坊网络上每个交易或计算成本保持稳定的价值。

- gas 成本是静态值，是以 gas 为单位的计算成本，目的是保持 gas 的真实价值不变，所以这个成本会一直稳定。
- gas 价格是以另一货币或代币比如说以太币为单位的 gas 成本。为了保持 gas 价值稳定，gas 价格是浮动值 ,如果代币或货币成本波动 ,gas 价格也会变化以保持同样的真实价值。Gas 价格的设定是根据多少用户愿意花费和多少进程节点愿意接受的平衡价格。
- Gas 限制是每个区块能使用的 gas 最大限额，被视为计算工作量，交易量和区块大小的最大值，矿工可以随着时间慢慢改变这个值。
- gas 费用是运行一个特别的交易或程序（被称作合约）所需的 gas。 一个区块的 gas 费用可以用来暗示计算工作量，交易量和区块大小。gas 费用支付给矿工（或 PoS 中的担保承包人） 。

## 13.4 参考资料

文章中的列表对应的链接可以参考《Ethereum Homestead Documentation》第 54 页 1.4.4 Online wallets, paper wallets, and cold storage

# 第十四章 公有链、联盟链和私有链

## 14.1 以太坊网络

去中心化共识的基础是参与节点的点对点网络，节点维护并保证区块链网络的安全。参见挖矿。

## 14.2 以太坊网络数据统计

EthStats.net 是以太坊网络实时数据的仪表板，这个仪表板展示重要信息，诸如现在的区块，散表难度，gas 价格和 gas 花费等。页面上显示的节点只是精选了网络上的实际节点。任何人都可以在 EthStats 仪表板上添加他们的节点。Github 上的 Eth-Netstats README 描述了如何连接。

EtherNodes.com 展示了节点数的当前和历史数据以及以太坊主网络和 Morden 测试网络上的其他信息。

当前实时网络上客户端实现分配 – EtherChain 上的实时数据。

## 14.3 公有链、私有链和联盟链

当今大多数以太坊项目都依靠以太坊作为公有链，公有链可以访问到更多用户，网络节点，货币和市场。然而通常有理由更偏好私有链或联盟链（在一群值得信任的参与者中）。例如，银行领域的很多公司都希望以太坊作为他们私有链的平台。

以下是博客发文《关于公有链和私有链》的摘录，它解释了三种区块链在许可方面的区别：

- 公有链：世界上所有人都可以阅读和发送交易。如果他们合法都有希望看到自己被包括在内。世界上任何人都能参与到共识形成过程——决定在链条上添加什么区块以及现状是怎样的。作为中心化或准中心化信任的替代品，公有链受加密经济的保护，加密经济是经济激励和加密图形验证的结合，用类似工作量证明或权益证明的机制，遵循的总原则是人们影响共识形成的程度和他们能够影响的经济资源数量成正比。这类区块链通常被认为是"完全去中心化"。

- 联盟链：共识形成过程由预先选择的一系列的节点所掌控，例如，设想一个有 15 个金融机构的团体，每个机构都操作一个节点，为了使区块生效，其中的 10 个必须签署那个区块。阅读区块链的权利可能是公开的，或仅限于参与者，也有混合的路径，比如区块的根散表和应用程序编程接口一起公开，使公共成员可以进行一定量的查询，重获一部分区块链状态的加密图形证明。这类区块链被认为是"部分去中心化"。

- 私人区块链：书写许可对一个组织保持中心化。阅读许可可能是公开的或者限制在任意程度。应用很可能包含对单个公司内部的数据库管理，审查等，因此公共的可读性在很多情况下根本不必要，但在另一些情况下人们又想要公共可读性。 私有链/联盟链可能和公有链毫无联系，他们仍然通过投资以太坊软件开发，对以太坊整体生态系统有利。经过一段时间，这会转变成软件改善，知识共享和工作机会。

# 14.4 如何连接

Geth 会持续尝试在网络上连接到其他节点，直到有了端点为止。如果你在路由器上有可用的 UPnP 或者在面向因特网的服务器上运行以太坊，它也会接受其他节点的连接。

Geth 通过发现协议找到对等端。在发现协议中，节点互相闲聊发现网络上的其他节点。最开始，geth 会使用一系列辅助程序节点，这些辅助程序节点的端点记录在源代码中。

# 14.5 检查连接和 ENODE 身份

要检查客户端在交互控制台上连接了多少对等端点，net 模块有两个属性可以提供信息，告诉你对等端点的数量以及你是否在监听的节点。

```
> net.listening
true
> net.peerCount
4
```

了解更多关于连接对等端点的信息，比如 IP 地址、端口号和支持协议，用管理员对象的 peers()功能。admin.peers()会返回到现在已连接的对等端点列表。

```
> admin.peers
[{
ID:
'a4de274d3a159e10c2c9a68c326511236381b84c9ec52e72ad732eb0b2b1a2277938f785
93cdbe734e6002bf23114d434a085d260514ab336d4acdc312db671b',
Name: 'Geth/v0.9.14/linux/go1.4.2',
Caps: 'eth/60',
RemoteAddress: '5.9.150.40:30301',
LocalAddress: '192.168.0.28:39219'
}, {
```

```
ID:
'a979fb575495b8d6db44f750317d0f4622bf4c2aa3365d6af7c284339968eef29b69ad0d
ce72a4d8db5ebb4968de0e3bec910127f134779fbcb0cb6d3331163c',
Name: 'Geth/v0.9.15/linux/go1.4.2',
Caps: 'eth/60',
RemoteAddress: '52.16.188.185:30303',
LocalAddress: '192.168.0.28:50995'
}, {
ID:
'f6ba1f1d9241d48138136ccf5baa6c2c8b008435a1c2bd009ca52fb8edbbc991eba36376
beaee9d45f16d5dcbf2ed0bc23006c505d57ffcf70921bd94aa7a172',
Name: 'pyethapp_dd52/v0.9.13/linux2/py2.7.9',
Caps: 'eth/60, p2p/3',
RemoteAddress: '144.76.62.101:30303',
LocalAddress: '192.168.0.28:40454'
}, {
ID:
'f4642fa65af50cfdea8fa7414a5def7bb7991478b768e296f5e4a54e8b995de102e0ceae
2e826f293c481b5325f89be6d207b003382e18a8ecba66fbaf6416c0',
Name: '++eth/Zeppelin/Rascal/v0.9.14/Release/Darwin/clang/int',
Caps: 'eth/60, shh/2',
RemoteAddress: '129.16.191.64:30303',
LocalAddress: '192.168.0.28:39705'
} ]
```

要检查 geth 使用的端口，发现你自己的 enode URI 执行：

```
> admin.nodeInfo
{
Name: 'Geth/v0.9.14/darwin/go1.4.2',
NodeUrl:
'enode://3414c01c19aa75a34f2dbd2f8d0898dc79d6b219ad77f8155abf1a287ce2ba60
f14998a3a98c0cf14915eabfdacf914a92b27a01769de18fa2d049dbf4c17694@[::]:303
03',
NodeID:
'3414c01c19aa75a34f2dbd2f8d0898dc79d6b219ad77f8155abf1a287ce2ba60f14998a3
a98c0cf14915eabfdacf914a92b27a01769de18fa2d049dbf4c17694',
IP: '::',
DiscPort: 30303,
TCPPort: 30303,
Td: '2044952618444',
ListenAddr: '[::]:30303'
}
```

## 14.6 更快下载区块链

启动以太坊客户端时，会自动下载以太坊区块链。用于下载以太坊区块链的时间会根据客户端、客户端设置、连接速度和可用的端点数量变化。下面是更快获取以太坊区块链的一些选项。

## 14.7 使用 geth

如果你在用 geth 客户端，你可以做些什么来加速下载以太坊区块的时间。如果你用—fast 标志来执行以太坊快速同步，不会保留过去的交易数据。

注意：你不能在执行所有或者部分正常的同步操作之后再使用这个标志，也就是说在用这个指令之前，不能下载以太坊区块链的任何部分。查看这个 Ethereum Stack.Exchange answer 了解更多。

下面是想要更快同步客户端时使用的一些标志。

—fast

这个标志使通过状态下载而不是下载整个区块数据来实现快速同步成为可能。这样也能大幅减少区块链尺寸。注意：—fast 只在从头开始同步区块链，并且是出于安全原因第一次下载区块链时 ,才会运行。查看 Reddit 发文了解更多。

—cache=1024

分配到内部缓存的千兆内存（最少 16MB / 数据库）。默认是 16MB，所以根据你电脑内存多少，增加到 256, 512, 1024 (1GB)或者 2048 (2GB)会带来不同。

—jitvm 这个标志可以激活 JIT VM。

完整的控制台命令示例：

```
geth --fast --cache=1024 --jitvm console
```

了解更多关于快速同步和区块链下载次数的讨论，查看这篇 Reddit 发文。

## 14.8 导出/导入区块链

如果你已经同步了整个以太坊节点，可以从完全同步的节点中导出区块链数据并将其导入新节点。你可以在 geth 中用 geth export filename 指令导出所有节点，并用 geth import filename 将区块链导入节点，来实现这一目的。

## 14.9 静态节点，信任节点和启动节点

Geth 支持一个叫静态节点的特征，如果你有特定的端点，你会一直想与静态节点连接。如果断开连接 ,静态节点会再次连接。你可以配置永久性静态节点，方法是将如下所说的放进/static-nodes.json（这应该是和 chaindata 以及 keystone 在同一个文件夹）

```
[
"enode://f4642fa65af50cfdea8fa7414a5def7bb7991478b768e296f5e4a54e8b995de1
02e0ceae2e826f293c481b5325f89be6d207b003382e18a8ecba66fbaf6416c0@33.4.2.1
:30303",
"enode://pubkey@ip:port"
]
```

你也可以在运行期间通过 Javascript 使用 admin.addPeer()加入静态节点。

```
>
admin.addPeer("enode://f4642fa65af50cfdea8fa7414a5def7bb7991478b768e296f5
e4a54e8b995de102e0ceae
```

# 14.10 连接的常见问题

有时候可能无法连接，最常见的原因有：

- 本地时间不正确。要参与到以太坊网络中，需要精确的时钟。检查 OS 如何同步时钟（例如 sudo ntpdate -s time.nist.gov），即便只快了 12 秒也有可能导致 0 端点。
- 有的防火墙配置可能会阻止 UDP 流通。可以用静态节点功能或者控制台上的 admin.addPeer() 来手动配置连接。

  不使用发现协议来启动 geth，你可以用—nodiscover 参数。你只会在运行测试节点或有固定节点的实验测试网络时才想要这样做。

# 第十五章 搭建测试网络和私有链

## 15.1 测试网络

### 15.1.1 Morden 测试网

Morden 是公开的以太坊替代测试网。它会贯穿于整个软件里程碑 Frontier 和 Homestead。

## 用法

eth (C++客户端) 0.9.93 及以上版本自动支持。比如开启以下任意客户端时，通过—morden 参数。

PyEthApp (Python 客户端) PyEthApp 支持 v1.0.5 以后的 morden 网络。

geth (Go 客户端)

## 细节

除以下几条，所有参数都和主要的以太坊网络相同：

- 网络名称：Morden
- 网络身份：2
- genesis.json（如下）;
- 初始账户随机数（IAN）是 220 （不像之前的网络中是 0）

– 状态树形结构中的所有账户都有随机数>= IAN。

– 账户被插入到状态树形结构中时，都会被赋予一个初始随机数= IAN。

- 初始通用区块散表：

  0cd786a2425d16f152c658316c423e6ce1181e15c3295826d7c9904cba9ce303

- 初始通用状态根：

  f3f4696bbf3b3b07775128eb7a3763279a394e382130f27c21e70233e04946a9 Morden 的 genesis.json

### 获取 Morden 测试网以太币

有两种方法可以获取 Morden 测试网以太币：

- 用 CPU/GPU 挖矿（参见挖矿）。
- 用以太坊 wei 龙头。

## 15.2 设置本地私有测试网

## eth (C++ 客户端)

可以使用–genesis 和–config 连接到或创建一个新的网络。

可以同时使用–config 和–genesis。

那样的话，–config 提供的初始区块描述会被–genesis 选项覆盖。

注意：包含一个网络的 JSON 描述。

- sealEngine (用来在区块挖矿的引擎)

  "Ethash"是以太坊工作量证明引擎(用于实时网络)。

  "NoProof" 在区块挖矿不需要工作量。

- params (诸如 minGasLimit, minimumDifficulty, blockReward, networkID 等一般的网络信息)
- genesis (初始区块描述)
- accounts (设置包含账户/合约的初始状态) 这是一个 Config 的例子(用于 Olympic 网络)：

  注意：包含一个网络的 JSON 描述。

  内容与'config'参数提供的初始领域相同。

# geth (Go 客户端)

你可以在私有测试网上生成或挖掘自己的以太币。这个试验以太坊方法很划算，可以避免不得不挖矿，或找到 Morden 测试网络的以太币。

在私有链中需要详细说明的事件有：

- 定制初始文件
- 定制数据目录
- 定制网络 ID
- (推荐) 废弃节点发现

## 初始文件

初始区块是区块链的起始 — 第一个区块，区块 0，唯一没有指向前面区块的一个区块。协议确保其他节点不会和你的区块链一致，除非他们和你有相同的初始区块，这样你想创建多少私有测试网区块链，就可以创建多少！

```
{
"nonce": "0x0000000000000042", "timestamp": "0x0",
"parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
"extraData": "0x0", "gasLimit": "0x8000000", "difficulty": "0x400",
"mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
"coinbase": "0x3333333333333333333333333333333333333333", "alloc": { }
}
```

存储文件为 CustomGenesis.json。用下面的标志启动 geth 节点的时候，你会引用到这个。

```
--genesis /path/to/CustomGenesis.json
```

# 私有网络的命令行参数

有一些必需的命令行选项（又称为"标志"）来确保你的网络是私有的。我们已经谈到了初始标志，下面还有几个。注意所有下面的指令都会用在 geth 以太坊客户端。

```
--nodiscover
```

使用这个命令可以确保你的节点不会被非手动添加你的人发现。否则，你的节点可能被陌生人的区块链无意添加 ,如果他和你有相同的初始文件和网络 ID。

```
--maxpeers 0
```

如果你不希望其他人连接到你的测试链，可以使用 maxpeers 0。反之，如果你确切知道希望多少人连接到你的节点，你也可以通过调整数字来实现。

```
--rpc
```

这个指令可以激活你节点上的 RPC 界面。它在 geth 中通常被默认激活。

```
--rpcapi "db,eth,net,web3"
```

这个命令可以决定允许什么 API 通过 RPC 进入。在默认情况下，geth 可以在 RPC 激活 web3 界面。

重要信息：请注意在 RPC/IPC 界面提供 API，会使每个可以进入这个界面（例如 dapp's）的人都有权限访问这个 API。注意你激活的是哪个 API。Geth 会默认激活 IPC 界面上所有的 API ,以及 RPC 界面上的 db,eth,net 和 web3 API。

```
--rpcport "8080"
```

将 8000 改变为你网络上开放的任何端口。Geth 的默认设置是 8080.

--rpccorsdomain "http://chriseth.github.io/browser-solidity/" 这个可以指示什么 URL 能连接到你的节点来执行 RPC 定制端任务。务必谨慎，输入一个特定的 URL 而不是 wildcard（*），后者会使所有的 URL 都能连接到你的 RPC 实例。

```
--datadir "/home/TestChain1"
```

这是你的私有链数据所储存在的数据目录（在 nubits 下）。选择一个与你以太坊公有链文件夹分开的位置。

```
--identity "TestnetMainNode"
```

这会为你的节点设置一个身份，使之更容易在端点列表中被辨认出来。这个例子说明了这些身份如何在网络上出现。

# 发布 geth

你创建了定制初始区块 JSON 并建立区块链数据目录后，在控制台输入以下指令，进入 geth：

```
geth --identity "MyNodeName" --genesis /path/to/CustomGenesis.json --rpc
--rpcport "8080" --rpcco
```

注意：请改变标志与定制设置匹配。

每次想要进入定制链的时候，你都需要用定制链指令启动 geth 实例。如果你只在控制台输入"geth"，它不会记住你设置的所有标志。

## 给账户预分配以太币

"0x400"难度能让你再私有测试网链上快速挖以太币。如果你创建了自己的链，开始挖矿，你应该几分钟就会有上百个以太币，远远超过了在网络上测试交易所需的数量。如果你还想给账户预分配以太币，就需要：

1. 创建私有链以后再创建新的以太坊账户。
2. 复制新的账户地址。
3. 在 Custom_Genesis.json 文件中添加以下指令：

   "alloc": { "": { "balance": "200000000000000000000" } }

   注意：用你的账户地址取代 0x1fb891f92eb557f4d688463d0

   d7c560552263b5a

   保存初始文件，重新运行私有链指令。 Geth 完整装载以后，关闭它。

我们想指派一个地址给变量 primary，查看它的余额。

在终端运行 geth account list 指令 ,查看指派给你的新地址账户号码是什么。

```
> geth account list

Account #0: {d1ade25ccd3d550a7eb532ac759cac7be09c2719}
Account #1: {da65665fc30803cb1fb7e6d86691e20b1826dee0}
Account #2: {e470b1a7d2c9c5c6f03bbaa8fa20db6d404a0c32}
Account #3: {f4dd5c3794f1fd0cdc0327a83aa472609c806e99}
```

记录你预分配以太币的账户号码。或者，可以用 geth console（和最先启动 geth 时保持一样的参数）启动控制台。提示出现以后，输入

```
> eth.accounts
```

这会返回到你拥有的账户地址排列。

```
> primary = eth.accounts[0]
```

注意 :用你的账户指数取代 0 ,这个控制台指令会返回到你第一个以太坊地址。

输入以下指令：

```
> balance = web3.fromWei(eth.getBalance(primary), "ether");
```

这应该会返回到 7.5，意味着你账户里有那么多以太币。我们必须在你初始文件的分区里放那么多数量是因为"余额"领域以 wei 为单位取一个数字，wei 是以太坊货币以太币的最小面额（参见以太币）。

- https://www.reddit.com/r/ethereum/comments/3kdnus/question_about_private_chain_mining_dont_upvote/
- http://adeduke.com/2015/08/how-to-create-a-private-ethereum-chain/

# 第十六章 什么是挖矿和 Ethash 算法？

## 16.1 挖矿简介

挖矿这个词源于对加密货币与黄金的类比。黄金或贵金属很稀有，电子代币也是，增加总量的唯一方法就是挖矿。以太坊也是这样，发行的唯一办法就是挖矿。但是不像其他例子，挖矿也是通过在区块链中创建、验证、发行和传播区块来保护网络的方法。

- 挖以太币=保护网络=验证计算

## 16.2 什么是挖矿？

以太坊，和所有区块链技术一样，使用激励驱动的安全模式。共识基于选择具有最高总难度的区块。矿工创造区块，其他人检测有效性。区块只有在包含特定难度的工作量时才有效，还有其他合格性条件。请注意到以太坊 Serenity 里程碑，可能就会被取代（参考权益证明模型）。

以太坊区块链在很多方面与比特币区块链类似，但也有些不同。在区块链架构方面，以太坊和比特币之间最主要的的区别是，不像比特币，以太坊区块不仅包含交易列表也包含最近状态（merkle patricia 特里结构的根散表编码在状态中更精确）除此之外，另外两个值，区块数和难度，也储存在区块中。

使用的工作量证明算法叫 Ethash（Dagger-Hashimoto 算法的改良版本），包括找到算法的随机数输入以使结果低于特定的难度阈值。工作量证明算法的意义在于，要找到这样一个随机数，没有比列举可能性更好的策略，而解决方

法的验证琐碎又廉价。由于输出有均匀分布（是散表功能应用的结果），我们可以保证，平均而言，需要找到这样一个随机数的时间取决于难度阈值。这使得只通过操纵难度来控制找到新区块的时间成为可能。

正如协议中所描述的，难度动态调整的方式是每 15 秒整个网络会产生一个区块。我们说网络用 15 秒区块时间生产一个区块链。这个"心跳"基本上主要强调系统状态同步，保证不可能维持一个分叉（允许 double spend）或被恶意分子重写历史，除非攻击者有半数以上的网络挖矿能力（即所谓的 51%攻击）。

任何参与到网络的节点都可能是矿工，预期的挖矿收益和他们的（相对）挖矿能力或者说成正比，比如被网络总散表率标准化的，每秒尝试的随机数数量。

Ethash 工作量证明是内存难解的，这使它能抵抗 ASIC。内存难解性由工作量证明算法实现，需要选择依靠随机数和区块标题的固定资源的子集合。这个资源（几十亿字节大小的数据）叫做 DAG。每 3000 个区块的 DAG 完全不同，125 小时的窗口叫做 epoch（大约 5.2 天），需要一点时间来生成。由于 DAG 只由区块高度决定，它可以被事先生成，如果没有被事先生成，客户端需要等到进程最后来生产区块。如果客户端没有预生成并提前缓存 DAG，网络可能会在每个 epoch 过渡经历大规模区块延迟。注意不必要生成 DAG 以验证工作量证明，它可以在低 CPU 和小内存的状态下被验证。

在特殊情况下，从零开始创建节点的时候，只有在为现存 epoch 创建 DAG 的时候才会开始挖矿。

## 16.3 挖矿奖励

获奖区块的成功工作量证明矿工会获得：

- "获胜"区块的静态区块奖，包含 5.0（5 个）以太币
- 区块内支出的 gas 成本 — 一定数量的以太币，取决于当前 gas 价格
- 叔伯块的额外奖励，形式是每个叔伯块包含额外的 1/32

在区块中执行所有交易所消费的、由获胜矿工提交的 gas 都由每个交易的发送者支付。已发生的 gas 成本归到矿工账户作为共识协议的一部分。随着时间变化，这会使数据区块奖变得矮小。

叔伯块是稳定的区块，比如说，和包含先前区块（最多回 6 个区块）的父区块。有效的叔伯块会受到奖励以中和网络滞后给挖矿奖励带来的影响，因而提升安全性（这叫做 GHOST 协议）。叔伯块由成功工作量证明矿工形成的区块中所包含的叔伯块接收 7/8 的数据区块奖励（=4.375 以太币）。每个区块最多允许 2 个叔伯块。

- reddit 上的叔伯块 ELI5
- 解释叔伯块的分论坛

挖矿的成功取决于设定的区块难度。区块难度动态调整每个区块，以规定网络散列能力来创造 12 秒区块时间。找到区块的机会因此由与难度相关的散列率产生。

## 16.4 Ethash DAG

Ethash 将 DAG（有向非循环图）用于工作量证明算法，这是为每个 epoch 生成，例如，每 3000 个区块（125 个小时，大约 5.2 天）。DAG 要花很长时间生成。如果客户端只是按需要生成它，那么在找到新 epoch 第一个区块之前，每个 epoch 过渡都要等待很长时间。然而，DAG 只取决于区块数量，所以可以预先计算来避免在每个 epoch 过渡过长的等待时间。Geth 和 ethminer 执行自动的 DAG 生成，每次维持 2 个 DAG 以便 epoch 过渡流畅。挖矿从控制台操控的时候，自动 DAG 生成会被打开和关闭。如果 geth 用—mine 选项启动的时候，也会默认打开。注意客户端分享 DAG 资源，如果你运行任何客户端的多个实例，确保自动的 DAG 生成只在一个实例中打开。

为任意 epoch 生成 DAG：

```
geth makedag <block number> <outputdir>
```

实例 geth makedag 360000 ~/.ethash.。请注意 ethash 为 DAG 使用 ~/.ethash (Mac/Linux) 或~/AppData/Ethash (Windows)，这样它可以在不同的客户端实现以及多个运行实例中分享。

## 16.5 算法

我们的算法，Ethash（之前被称为 Dagger-Hashimoto），是基于一个大的、瞬时的、任意生成的、形成 DAG（Dagger-part）的资料组规定，尝试解决它一个特定的约束，部分通过区块标题散列来决定。

它被设计用于在一个只有慢 CPU 的环境中来散列快速验证时间，但在被提供大量高带宽内存时，为挖矿提供大量的加速。大量内存需求意味着大规模矿工

获得相对少的超线性利益。高带宽需求意味着从堆在很多超速处理单元、分享同样内存的加速在每个单独的单元给出很少的利益（译者注：通过阻止专用芯片共享内存的方式，降低矿机的作用）。

没有节点验证的利益因而阻碍中心化，这在挖矿中很重要。

外部挖矿应用和以太坊工作规定和报送的后台程序之间的交流通过 JSON-RPC API 发生。提供两个 RPC 功能 ;eth_getWork 和 eth_submitWork。

这些被正式记录在 JSON-RPC API 维基百科文章的矿工条目下。

为了挖矿你需要一个完全同步的、能够挖矿的以太坊客户端和至少一个以太坊账户。这个账户用于发送挖矿奖励，通常被称为货币基或以太基。查看这个说明的"创建帐户"章节，学习如何创建帐户。

警告：开始挖矿前，确保区块链和主链完全同步，否则就不能在主链上挖矿。