
 区块链大本营

区块链开发者报告

BLOCKCHAIN DEVELOPER REPORTS

JAN 2019 |  CSDN

5 技术扫描

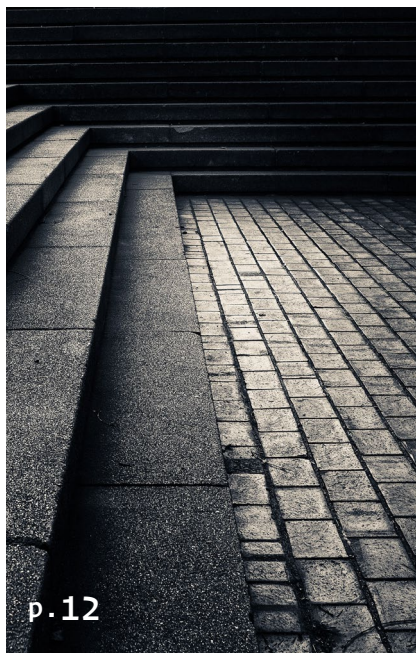
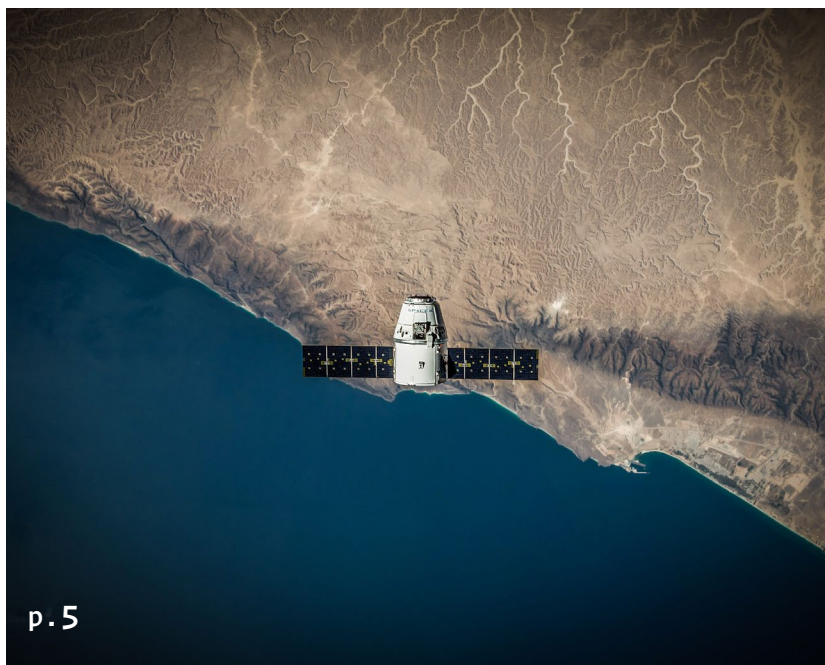
报告发布周期中，值得关注的新工具与框架、新技术、语言、平台。并根据其成熟度，分为暂缓、评估、验证、采用

12 公链选择指南

涵盖广受开发者关注的 36 个公链项目，由专家委员会，依据针对开发者的需求，评估项目的各关键部分

16 公链与应用开发实战

内容涵盖从入门到进阶，上手公链开发，通证简史，公链设计与开发实践和经验揭秘，DApp 开发者公链选择指南，以及如何打磨易用的区块链产品



知识产权及免责声明

本报告由 CSDN 制作，所有数据、表格、图片均受有关商标和著作权法律保护，若需复制、传播请先与我们联系授权。

报告中所载内容为 CSDN 通过访谈、市场调查、信息调研整理及其他方式获得，受研究方法和数据获取渠道所限，本报告只提供开发者参考，不构成任何投资或交易买卖建议。

关注我们



blockchain.csdn.net

✉ heycc@csdn.net

让区块链回归技术和应用的本质

过去一年，你我共同见证了百链上线、通证盛行、监管发布、应用爆发、行业跌宕起伏，目睹了人才和行业的教育普及，以及围绕区块链的各种新技术特征的提出和落地。

区块链已经从一个仅仅支撑交易转账的 Bitcoin 类架构向集多领域技术于一身的新一代信任基础设施方向演进，越来越多的细分行业和社会结构开始出现融合的苗头。

作为一个多学科复合的新技术，区块链在各个方向上的解决方案都开始成形，并作为一个新兴学科开始登堂入室进入各大高校，区块链应用也呈现出多元化的发展方向，在数量和种类上都出现井喷式增长，区块链行业也开始积极配合监管，区块链概念也开始在大众中普及。

新年伊始，我们将继续以“让区块链回归技术和应用的本质”为宗旨，服务开发者，携手迈入 2019。

区块链大本营

2019 年 1 月

顾问组

(排名不分先后)

来 鑫	迅雷链总工程师
黄海泉	京东区块链 JD Chain 首席架构师
张建俊	腾讯区块链技术负责人
荆 博	百度区块链系统部资深研发工程师
尚 书	Zerohm 联合创始人
蓝昊翔	公信宝区块链研发总监
冯英飞	Alabs ADAG 主链项目技术负责人
钟文斌	量子链技术负责人
强科臻	Aurora 极光链 CTO
刘虔铭	NEL 项目总监
罗 鹏	NULS 产品经理
相里朋	工信部电子五所高级工程师
谭智勇	欧链科技联合创始人兼首席科学家
王志坚	NULS 核心团队技术产品部负责人
孙建平	银行应用软件系统资深专家
鲍 帅	好码安全科技联合创始人
Michael Yuan	CyberMiles 联合创始人首席科学家
吴忠信	北京志顶科技技术总监
宋承根	北京欧链科技 CTO
陈敏莲	湖南医学会信息专业委员会专家
张剑南	DoraHacks 发起人
杨 胜	安妮股份区块链系统架构师
海贼王	慢雾安全负责人
刘宇翔	Mediconcen CTO
郭莹城	广电运通首席架构师
淦 涵	链安区块链安全专家
唐飞虎	仙女座科技 CEO
韩思远	Virgo Network 技术负责人

技术扫描

点评 / 来鑫，黄海泉，张建俊，荆博

技术扫描归纳了专家对他们正在关注技术的评价结果，为从 CTO 到一线开发者提供技术选型指南。与区块链相关的各类技术项目被归类为工具与框架、技术、语言、平台四类，如果条目可能出现在多个象限，则选择最贴切的一个。我们还进一步将这些技术分为采用、验证、评估、暂缓四个阶段，阐明对在项目中应用这些技术的建议。在后续报告中，我们将进一步扩充覆盖技术的数量与范围。



应用等级

① 采用

我们主张业界采用这些技术。如果适合我们自己的项目，也会毫不犹豫地使用。

② 验证

值得追求。企业应该在风险可控的项目中尝试此技术。

③ 评估

值得研究一番的技术，需确认它将对企业产生何种影响。

④ 暂缓

三思而行，别用这技术启动任何新项目，它们可能已经过时。

工具 & 框架

采用

1. Ethers.js
2. Truffle
3. Geth
4. Solc
5. OpenZeppelin
6. Mist
7. Remix
8. Docker
9. RocksDB

验证

10. MyEtherWallet
11. MetaMask
12. Embark
13. GanacheCLI

评估

14. Substrate
15. MySQL

暂缓

16. EtherScripter

技术

采用

17. Gossip
18. PoW
19. PBFT



语言

采用

- 20. Java
- 21. Go
- 22. JavaScript

验证

- 23. Solidity
- 24. Rust
- 25. C++
- 26. Vyper

评估

- 27. Clojure
- 28. Red

采用

验证

评估

暂缓

平台

验证

- 29. Thunder Chain
- 30. EVM
- 31. Azure BaaS
- 32. Blockchain Testnet
- 33. Coinbase's API

评估

- 34. Tierion

33

32

30

31

29

34

工具 & 框架

Ethers.js 是针对以太坊钱包功能完整实现的工具包，其 API 文档十分详尽。

Truffle 用于开发合约工程的框架，是一套本地集成开发环境和编译测试调试工具。在 Truffle 提供的开发环境里，整合了前端实例化合约的方法，可以很便捷的调用合约功能。另外还有各种 box 支持开箱即用，整合了前端应用调用合约工程的途径。

Geth 是一个以太坊客户端，用 Go 语言编写，它是目前最常用的以太坊客户端，有着丰富的 API，不过没有可视化操作界面。可以与 Mist 配合来进行以太坊智能合约的开发调试。

Solc 是以太坊官方提供的 Solidity 编译工具。

OpenZeppelin 是已实现的一系列经安全验证的合约工具和 ERC 标准合约库，开发者可以通过继承这些合约和 library 方便地进行合约开发。

Mist 是以太坊 PC 钱包，可以选择连接不同的网络。包含账户管理、交易、调用合约、部署合约以及部分 DApp 功能的调用。针对以上功能所提供的界面化操作极大的方便了普通用户发送交易和调用合约。

使用 Mist 可以轻松连接以太坊测试网络，并支持开发、部署、调试智能合约。对开发者非常友好。

Remix 是一个集成的编写部署调试合约的浏览器 IDE，可以在 Remix 提供的浏览器页面快速编写部署合约，支持正式和测试网络连接、静态检查、本地调试、交易记录、事件和日志查询等功能。另外结合 Remixd 提供的 ws 服务可以连接本地文件进行开发。对兼容以太坊智能合约的链平台来说，Remix 是非常好用的合约编辑、测试、部署工具，用户众多，已经比较成熟。

Docker 在区块链中有多种用途，包括用于 BaaS 服务中作为实现动态节点管理的基础组件，在 Fabric 作为智能合约的运行容器等。

RocksDB 是一种 NoSQL 数据库，具有高性能和使用简单的特点，许多区块链项目都以 RocksDB 作为底层的存储数据库。

MyEtherWallet 是一个很受欢迎的网页版以太坊在线钱包，具有丰富的资产管理，账户备份功能。对于不想在本地安装钱包客户端，存储全量区块信息的用户来说，是一个不错的选择。曾经因为 DNS 劫持的问题，导致一些用户访问了钓鱼网站，损失了大量以太币资产。MyEtherWallet 代码开源，它不会存储用户的钱包信息账号，就算有一天 MyEtherWallet 网站不能使用，你也可以通过钱包的私钥和密码在其他钱包上找回你的钱包账号，可放心使用。

MetaMask 是一款在谷歌浏览器 Chrome 上使用的插件类型的以太坊钱包，该钱包不需

要下载，只需要在谷歌浏览器添加对应的扩展程序即可，非常轻量级，使用起来也非常方便，不需要下载全量区块信息，也能让小白用户管理自己的数字资产。对于以太坊 DApp 的普及和传播起到了非常大的作用。当年的以太猫游戏的火爆，Metamask 功不可没。

Embark 目前集成了 EVM 区块链（以太坊）、去中心化存储（IPFS）和去中心化通信平台（Whisper 和 Orbit），部署支持 Swarm，方便开发者搭建自己的区块链应用。

Ganache 是 Truffle 官方推荐使用的客户端之一（另一个是 Truffle 内置的 Truffle Develop）。

以太坊联合创始人 Gavin Wood 认为，所有人都从头构建网络和共识代码，非常浪费精力。使用 **Substrate** 来构建新项目，开发者所要做的，就是在代码调用少量函数，就能获得各种密码学模板，以及定制、搭建和发布新区块链所需要的方方面面，“Substrate 为区块链开发者提供最大限度的自由，花费最少的精力”。

关系数据库，如 **MySQL** 在一些区块链系统中用作记录系统对象的关联关系，以便提供比 NoSQL 更丰富的查询能力。但关系数据库的灵活性、伸缩性不好，并不太适合区块链这类数据单调递增的场景，在数据量增长上来之后对关系数据库的运维管理复杂，因此不推荐使用，建议考虑其它的替代方案。

曾经广泛使用的 **EtherScripter** 已被淘汰，Embark、Truffle 是其替代者。

技术

GOSSIP 是一种消息通讯算法，广泛用于面向公链的区块链系统中，包括比特币、以太坊等，用于实现点对点通信。

POW 是目前面向公链的众多共识算法中唯一具有安全性证明的算法，在中本聪的论文已论证，并且得到比特币系统的运行所证明。缺点是需要消耗算力。

PBFT 是一种拜占庭容错的共识算法，适合用于面向联盟链的场景，在许多区块链项目上都采用了 PBFT 算法及其变种算法；但原始的 PBFT 算法本身不支持节点的动态增加，在实际应用时需要注意。

语言

Java 作为一项成熟的通用语言，掌握开发人员多，生态全面，被广泛运用于各类企业级系统、中间件的开发，在区块链中也被一些项目用做系统开发语言和合约开发语言。

Go 是高效快速的应用逻辑开发语言，编译速度优越，已经有较为强大的基础库，属于新兴语言，很被大家看好。作为一项通用语言，具有内存安全、性能优异的特点，用于区块链底层系统开发。由于 Go 简洁易用的语言特性、功能丰富的代码库以及优异的性能表现，很多区块链底层是用 Go 语言实现的，Hyperledger Fabric 等项目也主推用 Go 编写智能合约或 DApp，可说 Go 语言在区块的应用中已经非常成熟。

Go 的高性能，预编译与语言学习的易上手性，使得它非常适合被用来编写区块链项目。目前，大量的新的区块链项目都是用 Go 来编写。

JavaScript 开发方便，很多区块链项目支持。Lisk 支持使用 JavaScript 来开发智能合约，这使得智能合约的开发对于 Web 开发者来说，变得更加的简单。

Solidity 是以太坊 EVM 虚拟机上编写智能合约的图灵完备语言，也是当前智能合约最实用和使用最广泛的语言，语法类似 JavaScript，适用于以太坊及其扩展实现，只能由 EVM 加载执行。但受制于 EVM 包括以太坊本身的扩展性不够等设计缺陷，在开发效率和运行速度上有一定的限制。不过 Solidity 还在不断的发展和更新，可以期待未来更优的使用体验，但如果用于正式的重要的业务中，需要谨慎考虑。

Rust 作为一项通用语言，具有内存安全、性能优异的特点，也广泛使用于区块链底层系统的开发。已经有一些区块链项目采用了该语言作为开发语言。但开发人群还相对较少，选择该语言还需要考虑开发生态的成熟度。Rust 在不影响性能的前提下提供的安全特性非常有吸引力，这些安全特性也很适用于区块链开发。如 steem 等区块链项目用 Rust 开发，经过适当的限制后也适用于智能合约开发。Rust 可以被 C 及其它多种语言轻松调用，且由于其高性能特性，非常适合被用来编写区块链相关组件，比如可插拔密码库和共识机制。

作为一项有历史的成熟的经典语言，**C++** 通常用于开发偏底层的系统，且不断迭代升级，引入新特性新思想，性能一直较好，但由于语言特性复杂，容易产生内存安全的问题，对开发者要求高，在有其它内存安全的语言可选择的情况下，C++ 不是进行区块链系统开发的最优选择。EOS 等项目是用 C++ 开发的，但由于其复杂性，使用时需要非常小心。

另外，学习曲线较陡，使得越来越多的新的区块链项目不再使用 C++ 语言来进行编写。但是 C++ 高性能和编译特性，使得它仍然会在一些区块链项目的底层开发中发挥作用。

Vyper 是新的以太坊语言，它为开发者提供了 Solidity 的一种替代选择。

已有项目采用 Clojure 编写合约解释器，但学习曲线较为陡峭，现有开发者中，熟练掌握的人相当少，可能会面对缺人或维护困难的窘境。

Red 拥有出色的表达及 DSL 能力，但项目仍在开发中，基础尚不完备。

平台

迅雷链平台 (Thunder Chain) 是高性能、高可靠性、开发方便、技术支持完善的主链平台。

EVM 是以太坊上的智能合约引擎，用于执行 Solidity 的智能合约，它是一个栈式虚拟机，其安全性是还需要完善，众多针对以太坊的攻击方法都是由 EVM 本身的缺陷带来的，虽然爆出的缺陷已经大部分修复，但毕竟 EVM 出现的时间相对还很短，所以还需更多时间来观察其进展。

和普通节点相比，**Azure BaaS** 节点好处主要是，能让开发者快速建立自己所需的开发环境，能帮助更快地验证自己的概念和模型，工具性更强，便于创建、部署、运行和监控区块链服务。但其实用性还有待验证。

Blockchain Testnet 可用于测试和体验操作。允许 DApp 开发者使用测试网络方便的进行开发和调试，而不必担心在主链上测试导致的费用问题。

Coinbase's API 的提供者 Coinbase 是全球用户最多的交易所之一。

Tierion 是将区块链用于验证任何数据、文件或过程的平台，应用场景包括文件存在性证明、审计跟踪等，应用场景有限，决定使用这个平台之前请先评估是否跟自己的实际业务场景相匹配。❖

公链选择指南

评委 / 相里朋，谭智勇，王志坚，孙建平，尚书，鲍帅，傅娆，吴忠信，宋承根，陈敏莲，
张剑南，杨胜，海贼王，刘宇翔，郭莹城，淦涵，唐飞虎，韩思远

尽管公链项目排行屡见不鲜，但均未针对开发者的需求而设计。例如，我们通过与上百位区块链开发者深入沟通，了解到他们对于公链项目最关注的角度是与实际开发过程及产品关联更紧密的文档完备、性能优秀等。因此，本期报告选择了一批具有独立主链，公有节点可自由创建，并且代码开源的公链项目，邀请专家对开发者最关注的 10 个维度评分，供开发者参考。未来，我们将持续更新并扩充该列表。

评分标准

(回避自有项目后，评分均值取整)

评价维度	满分	标准
文档完备	10	代表文档、教程齐备易用
性能优秀	10	代表速度快，性能强
行业认可	10	代表在项目适用的领域或行业有口皆碑
基础架构	10	代表基础架构完备
用于生产	10	代表推荐在生产环境中使用
易于开发	10	代表广泛支持常用编程语言，提供基本工具，易于开发者采用
特色创新	10	代表技术原创和创新度高，不容易被替代
社区活跃	10	代表社区活跃，开发中遇到的问题，能迅速得到社区或官方响应
生态应用	10	代表生态和应用开发活跃
项目可靠	10	代表项目运行、运营稳定

项目名称	综合得分	文档完备	性能优秀	行业认可	基础架构	用于生产	易于开发	特色创新	社区活跃	生态应用	项目可靠
Ethereum	9.1	10	7	10	8	9	7	10	10	10	10
Ethereum Classic	8.8	9	7	10	8	9	7	8	10	10	10
EOS	8.4	8	9	8	9	8	8	9	8	9	8
Nebulas	8.4	9	8	8	9	8	9	9	8	8	8
Steem	8.2	8	8	9	8	9	8	8	8	8	8
Ark	8.1	9	8	8	8	8	8	8	8	8	8
Stellar	8.1	9	8	8	7	8	9	8	8	8	8
NEM	8.1	8	8	8	8	8	8	9	8	8	8
NULS	8.0	8	8	8	8	8	8	8	8	8	8
GXChain	7.6	8	8	7	8	7	8	7	7	8	8
Cardano	7.6	8	8	8	7	6	8	9	7	7	8
Bitcoin	7.5	9	6	10	7	5	5	10	8	5	10
Sia	7.5	9	6	8	7	8	8	9	7	5	8
Qtum	7.4	8	7	7	8	8	7	7	7	7	8
IOTA	7.4	8	8	8	7	6	8	9	8	5	7
BitShares	7.3	8	8	7	8	6	6	8	7	7	8
Monero	7.3	8	6	10	7	5	5	10	7	5	10
Tezos	7.3	7	7	7	8	8	7	8	7	7	7

项目名称	综合得分	文档完备	性能优秀	行业认可	基础架构	用于生产	易于开发	特色创新	社区活跃	生态应用	项目可靠
Zcash	7.3	8	6	10	7	5	5	10	7	5	10
Ripple	7.0	8	8	9	7	5	5	8	7	5	8
Komodo	6.9	8	6	8	7	6	5	8	8	5	8
Verge	6.8	8	6	8	7	5	5	9	7	5	8
Dash	6.8	9	6	8	7	5	5	8	7	5	8
Bitcoin Cash	6.8	9	6	8	7	5	5	8	7	5	8
Litecoin	6.8	9	6	8	7	5	5	8	7	5	8
Lisk	6.7	6	6	7	8	7	8	7	6	6	8
Bytecoin	6.7	8	6	8	7	5	5	8	7	5	8
Decred	6.7	8	6	8	7	5	5	8	7	5	8
NANO	6.6	8	8	8	7	5	5	7	5	5	8
Waves	6.4	6	7	6	7	6	6	8	6	6	6
NEO	6.3	6	7	6	7	6	6	7	6	6	6
Stratis	6.3	6	7	6	7	6	6	7	6	6	6
Achain	6.3	6	7	6	7	6	6	7	6	6	6
CyberMiles	6.3	6	7	6	7	6	6	7	6	6	6
MOAC	6.3	6	7	5	7	5	7	7	6	6	8
Hcash	6.3	6	7	6	7	6	7	7	6	6	6

从入门到进阶，上手公链开发

文 / 蓝昊翔

如何选择一条公链

了解公链的第一步，是阅读白皮书。白皮书是公链的灵魂，也是驱动公链开发的指导性文档，通过阅读白皮书，可以找到一条区块链开发的完整愿景和路线图。

白皮书从形态上可分为产品白皮书和技术白皮书（或称为技术黄皮书），前者偏向于介绍背景和公链的功能愿景，后者则比较接近可行性论证和具体模块的技术设计，包含算法、公式等。

如何做出选择？公链都有自己的定位，其设计都围绕着白皮书中所描述的愿景和路线展开。如果说要选择一条公链参与其中，仔细阅读白皮书是第一步，对白皮书的技术和非技术愿景产生了认同，这个时候我们有才有理由告诉自己，这是我想做的事情。

对于公链这种社区型的项目来说，不像公司招人一样严格筛选，参与公链生态会更加开放一些，对于所有的贡献者是来者不拒的，当然，要做到给公链提交代码或者参与公链的设计，自身的硬实力是不可或缺的。

我能为公链贡献什么

如上文所说，我们选择了一条公链，那我们能为这个公链做什么呢？讨论这个问题，我想从以下两个大的方面展开。

擅长技术

如果你有技术背景，在公链上选择就很多，如果你擅长区块链底层技术，哪怕是其中一个模块一个学科，那么恭喜你，你可能是众多公链竞相吸引的公链开发者之一；如果你在特定领域的背景不是那么强，或者说对枯燥的底层算法没那么感兴趣，那不妨尝试下做一些区块链上层的应用，因为这些应用对于整个生态来说，像树叶一般：只有枝叶繁茂的大树，才能吸引众人眼球。

参与底层开发

在读懂白皮书或技术黄皮书的前提下，我们对公链的技术就有所了解，其实从每条公链

的核心组成部分去分析，我们可以概括出以下的知识结构，不妨看看，对这些技术概念，你掌握的如何？

核心技术模块

- 数据结构：区块、交易、账户模型、未花费交易输出 (UTXO)
- 密码学
 - ◆ 编码方式：Base64、Base58
 - ◆ 哈希算法：SHA2、SHA3
- 对称加密：AES
- 椭圆曲线密码学
 - ◆ 私钥、公钥、地址
 - ◆ 密钥交换算法
 - ◆ 签名验签
- P2P 网络：Kademlia
- 共识机制：一致性算法、PoW、PoS、DPoS、BFT
- 智能合约
 - ◆ 什么是智能合约
 - ◆ JVM、EVM、WebAssembly 有什么关系

如果对上述的内容你了然于心，那么说明你已经是一位合格的开发者（至少在我看来是的），如果对部分内容还熟悉，那欢迎选择性阅读以下内容：

私钥、公钥、地址

- 私钥：一个大整型（BigInteger）随机数 a
- 公钥：由椭圆曲线上一个固定点 G 通过几何算法计算出的 $a \cdot G$ ，其中 a 为私钥，由于椭圆曲线上的几何运算逆运算非常复杂，因此私钥可以推导公钥，公钥无法逆向推导私钥
- 地址：一般为公钥的摘要格式，当然在多签的网络里，可以用一个地址对应多个公钥

上述的公私钥指的是大多数区块链系统里面的公私钥，目前的主流区块链公私钥都是基于 ECC（椭圆曲线密码学）的。如果你希望了解更多，这是一篇不错的文章
<https://www.jianshu.com/p/af6328cc693e>

交易、区块、矿工

- 交易：一个用私钥签名的消息体，比如一个转账交易。每个交易都是可验证的，每个区块链的节点会对验证的交易进行存储，并且通过 P2P 网络广播出去

- 区块：一个区块中包含时间戳，包含一段时间内的多笔交易，以及一些验证字段（如当前区块的 hash，上个区块的 id 等），还会包含记录这个区块的矿工的签名；一段时间内发生的交易是散乱无序的，每个节点接受到交易的顺序是不一样的，那么，需要一种机制让所有的节点来通过一种约定的方式来确定交易发生的顺序，我们称这种行为为出块，我们称这样的机制为共识机制
- 矿工：即打包一段时间内的交易的人，矿工负责对所有交易进行线性梳理，通过出块的方式告知网络，先发生了什么，后发生了什么，只有通过这样的线性的方式，才能让整个网络的账本（或者说状态）是一致的。

Base64、Base58

其实很好理解，Base64 即基于 64 个字符的编码，Base58 即基于 58 个字符的编码。相比 Base64，Base58 不使用数字“0”，字母大写“O”，字母大写“I”，和字母小写“l”，以及“+”和“/”符号。

这样做的主要原因是为了肉眼容易识别，在输入的时候不容易打错，因此目前大多数的区块链系统是优先选择 Base58 的，当然像以太坊这样的项目甚至直接使用 16 进制编码了。

账户模型、UTXO

账户模型和 UTXO 实际上是两种记录账户余额的数据结构。UTXO 的简单说法就是总量不变的前提下，如 A 余额 10，B 余额 20，记为 (10,20)，而当 B 向 A 转账 5 后，记为 (10,20)⇒(15,15)；而同样的事情，在账户模型下，我们可能会记为 {A:15,B:15}。

相比之下：UTXO 的可验证性更强，而账户模型具备更好的可扩展性，比如账户模型下，更容易实现多重签名（一个账户由多把私钥来管理）。

关于密码学、P2P 网络、共识机制和智能合约，都属于一言难尽的内容，在此不多赘述。有兴趣的小伙伴可以看文后的推荐阅读资料。

常用的开发语言

对于不同的底层公链，开发语言可能有所不同，网上有比较多客观的分析，从我们对目前主流公链的比较主观的判断来看，不同语言的使用频率排序是这样的（仅从公链底层模块分析）：

C++ > GoLang > Rust > Java > Javascript

话说回来，毕竟语言只是工具，从核心技术和算法角度来说，语言的难度不算什么。

社区和公司的不同

公链项目的典型特征是以社区的方式开展的，这和公司的组织形式天差地别，没有固定的地理范围，没有语言和国家限制，这和区块链的网络结构很像，所有人都是点对点协作的。唯一能帮助所有技术贡献者聚焦的，除了白皮书，还有技术文档和技术社区。技术文档定义了一种规范，帮助所有人以特定的规范进行开发或调用，而社区是讨论和形成规范的地方。

参与生态建设

如果你对底层的技术不那么擅长，也可以选择做一些不那么底层的事情；更重要的是其实对于公链来说，底层开发不需要太多的人，而对于主链之上的工具和 DApp，则是一片巨大的蓝海。

基础设施开发

公链就算有再好的架构和 API 设计，也离不开几个重要的基础设施，来更好地连接开发者和用户，比如：

- SDK、IDE: SDK 帮助开发者方便地调用主链 API，IDE 则让开发者更好地编写智能合约。
- 客户端：无论是管理私钥，还是体验链上 DApp，客户端是用户的最好选择，一个方便、简单且安全的客户端往往是一条公链最大的流量入口，即用户聚集的地方。
- 区块浏览器、DApp 统计等信息类站点：区块浏览器就是区块链世界的 Google，无论是开发者还是用户，都离不开公链的汇总信息和统计类 API，这样的信息类站点，也是一个巨大的流量入口。

DApp 开发

构建在公链上的应用，不依赖于中心化的服务器，所有的和逻辑都运行在去中心化网络里，DApp 因此而得名；通过学习智能合约开发，应用一些主链 API 和 SDK，结合一些必要的前端技能，我们便可以开发自己的 DApp，并且快速发布到整个区块链网络，供用户使用，我们需要做的是找到用户的兴趣和痛点，并花时间去实现它。

不擅长技术

前面说了很多技术内容，也许会让不懂技术的小伙伴心灰意冷，但实际上区块链并不是一个只属于技术极客的世界，即使你不擅长代码实现，也是可以做很多事情的，比如：

优化提案：对公链开发来说，很多的想法可能不是核心团队提出来的，而是来自于社区的意见收集，而几乎所有的公链都有这样一个讨论和整理意见的地方，叫 Improvement Proposal，如比特币的 BIPs，和以太坊的 EIPs，参与的方式是先发起 Issue 讨论，社区协调员指定提案编号后整理成文档发布成一个固定格式的 Proposal 文档。任何人既可以参与

自己感兴趣的话题讨论，也可以发起讨论，即使你不懂技术，一个很好的意见，也可能改变公链未来的方向。

技术布道：如果你通晓公链的技术原理和通证设计，在保持技术中立和科学严谨的前提下，便可以考虑将你的想法传播给更多的人，从而扩大社区的共识范围，这其实是一件伟大且影响深远的事情，在这个过程中，你会体会到想法的碰撞和汇聚，直到一个想法最终影响到很多后来的人。

文档翻译：对于有语言基础的人来说，文档翻译可能是最简单不过的事情，但对社区来说却是非常重要，因为如前文所说，无论是开发者还是用户，进入社区到深入了解公链生态的第一步，可能就是白皮书或者技术文档，如果语言不是障碍，公链将更容易吸收来自世界各地的力量。所以，这么简单且重要的事情，何乐而不为呢？

合约、经济等理论

通证、通证经济、通证模型

在区块链的世界里，通证是密码学管理的、数字化的、可流通的权益证明，通证经济指的是把通证应用于激励和流通的经济活动，而通证模型是通证产生、流通和使用的规则。

智能合约

智能合约是运行在可复制、共享的账本上的计算机程序，可以处理信息，接收、储存和发送价值。

智能合约是通证模型的实现，一份完整的合约，不只是编程技术本身，还包含一个合理的通证模型，即通证是如何产生、流通和使用的。

智能合约的执行环境是一个沙盒，通过主链的 API 与外界交互，合约的执行环境，也就是我们常说的虚拟机，本质上来说，Java 的 JVM、以太坊的 EVM、EOS 或 GXChain 的 WebAssembly 虚拟机都是一样的道理，只是在智能合约里面能调用的 API，是区块链节点本身的数据。

预言机

如上文所述，智能合约是一个沙盒，其能力范围只能通过虚拟机提供的 API 主动访问节点本身的数据，那么如果能让智能合约能访问更多链外的数据，我们需要把这些数据搬上链，把链外可信数据搬上链的机制，我们叫预言机。

预言机不仅仅是把数据搬到链上这么简单，更重要的是做到可信，我们除了通过技术的手段，比如数字签名这样的技术外，还可以使用多方计算的方式增加数据的可信度和输入的

稳定性。

典型领域的知识和储备

分布式一致性

在分布式计算领域，我们常讨论强一致性和弱一致性，而对于区块链的系统来说，我们通常实现的是最终一致性，也就是说每一笔交易不是立即确认的，而是有一个最终确认的过程，这个最终确认的标准和过程，是在共识机制中研究和设计的重点，也是我们常说的 BFT、PoW、PoS、DPoS 这些共识机制的主要区别。

TPS 和 QPS 的区别

TPS 指的是系统每秒处理的交易数，QPS 是指系统的吞吐量，因此 QPS 看起来是比较广义的，而 TPS 才是区块链的终极性能指标。

不可能的三角 (Impossible Trinity)

不可能的三角也称为三元悖论，在我们生活中各个领域，都面临着鱼和熊掌不可兼得的艰难抉择：

- 比如国际金融领域：资本流动性 (Capital mobility)、汇率稳定性 (Exchange rate)、货币政策独立性 (Monetary policy)，三者是无法兼得的，追求资本稳定性，要保持汇率的稳定，国家需要制定各种经济政策，来应对国际经济形式的变化，当下中美贸易战就是一个最直观的例子了。
- 分布式计算领域，我们常说的 CAP，即一致性、可用性和分区容错性，也是一个难以取舍的问题。
- 同样区块链系统中去中心化、安全、性能，也是一直以来社区的一大争议，所以回到本文的论述要点，如何选择一条公链，或许你就需要在去中心化、安全、性能上有所取舍。例如，追求绝对的去中心化和安全，不考虑性能，比特币和以太坊的社区是一个很好的选择，而如果你希望合理地收敛去中心化程度来实现更高的性能，基于 DPoS 和各种 XPoS 共识机制的公链将是你的不二之选。

扩展阅读：值得参考的项目和资料

技术

- 入门必读之《比特币白皮书》<https://bitcoin.org/bitcoin.pdf>

- 为了鼓励一下作者的事业，我们来阅读一下《GXChain 白皮书》https://static.gxchain.org/files/GXChain_WhitePaper_v3.0_CN.pdf
- 能把密码学这么学术的问题阐述地最简洁明了的，当然是 Wiki 了《椭圆曲线密码学》https://en.wikipedia.org/wiki/Elliptic-curve_cryptography
- 安全，是智能合约最大的敌人，也是这个行业最尖端的人最关注的事情之一《智能合约安全》<https://medium.com/@merunasgrincalaitis/how-to-audit-a-smart-contract-most-dangerous-attacks-in-solidity-ae402a7e7868>
- 如果文中 UTXO 和账户模型的解释让你无法理解，可以阅读这篇 <http://8btc.com/article-4381-1.html>

非技术

- 如果我们非得非常学术地理解“通证经济” https://en.wikipedia.org/wiki/Token_economy
- “智能合约” <https://github.com/EthFans/wiki/wiki/%E6%99%BA%E8%83%BD%E5%90%88%E7%BA%A6> ♦

蓝昊翔，公信宝区块链研发总监。全栈开发工程师，精通区块链技术，Graphene 社区活跃成员和代码贡献者，5 年金融领域从业经验，2013 年接触区块链，并持续从事区块链技术的研究和应用。曾就职于美国道富、51 信用卡、大树金融，在密码学、数据库、服务端和移动端等多个领域都有丰富的实践经验。

通证简史——安全与灵活的博弈

文 / 尚书

通证的本质

区块链发展到今天，经历了三个里程碑：

- 2008 年比特币诞生，创建了真正意义上的去中心化点对点转账系统；
- 2015 年比特币启动，开创了去中心化应用程序的新框架；
- 2017 年 ERC20 爆发，引领了一种全新的价值载体，即 Token。

Token 一词常被翻译为“代币”，但这远远不能体现 Token 作为区块链上通用价值载体的潜力，叫做“通证”更为妥当，即可流通的价值证明。

通证不代表法币，它只是基于区块链技术衍生出的一种创新价值载体。对区块链而言，通证不是必须的。但如果缺失了通证，区块链将缺失价值载体，构建价值互联网的宏图将寸步难行。尽管离开通证，区块链依然可以成为一个极具特色的分布式数据库解决方案，虽同样具有应用价值，但难免让人叹息。

早在 2015 年 11 月 19 日，Fabian Vogelsteller 便在以太坊改进提案（EIP，The Ethereum Improvement Proposal）中以编号 20 提出了一组通证规则，允许用户发行一组彼此无差别的通证，并可以拥有和交易这些通证，就如同我们日常使用的货币一样，这就是后来大名鼎鼎的 ERC20 协议的雏形。在 2017 年，随着 ERC20 类型通证作为价值载体在众筹中的广泛使用，我们看到，通证信任成本和流通成本极低的特点，使得项目募资成本被大幅下降，新一轮的投资热潮迅速在全球引爆。

紧接着，2017 年 12 月 2 日，Axiom Zen 打造的 CryptoKitties 横空出世。随着 CryptoKitties 名声大噪的不仅是区块链游戏，更是另一种全新的通证规则，允许用户发行一组通证，其中每一个通证都是独一无二的，就如同我们日常使用的会员卡一样，这即是之后以编号 721 收录到 EIP 的 ERC721 协议。

回顾那些疯狂的岁月，ERC20 向我们展示了通证高效的一面，而 ERC721 则向我们展示了通证多样化的一面。作为一种全新的价值载体，通证不同于金钱，它提供更多维度的价值体现，如身份标识、权益比例、系统积分等。一个没有通证的区块链世界，多少会显得暗淡无光。

通证面临的两难博弈

那么，目前各大主流区块链平台通证使用的现状如何呢？链下资产通证化上链的进程是否顺利呢？答案是悲观的，在政策和技术两方面，区块链的通证之路都面临着安全与灵活性的博弈。

一方面，从政策角度看，通证是基于区块链衍生出来的一种全新价值载体，由于区块链不受控制的开放性，导致链上资产监管十分困难。自由主义对资产灵活性的诉求，与各国金融监管部门对资产安全性的要求背道而驰。金融监管是维护实体经济平稳发展的重要工具，我们切不可为了自由主义乌托邦式的理想而放弃监管，没有法律的约束，一个通证的权益和责任边界都不明确，这对于用户和项目方都具有极高的风险，极其不利于区块链技术落地。诸如项目方因资金管理不善跑路，以及投资用户因亏钱而骚扰项目方的负面事件，都无形中抹黑了区块链技术本身，对行业发展造成伤害。

所以，对于政策上安全和灵活性的博弈，我们理应优先保障金融监管的安全，但现实却完全相反。从密码朋克运动中发展出来的区块链技术，带着浓厚的自由主义气息，这一思潮几乎影响了目前所有主流区块链平台，导致各大平台在架构层面没有任何机制来帮助平台上的通证配合监管，通证发行方需要自己承担所有合规的成本，这将使区块链的通证之路走得十分坎坷。

另一方面，从技术角度看，通证是基于代码定义出来的一串规则，理论上我们可以使用图灵完备的智能合约语言定义出任意形式的通证，但编程语言不受约束的灵活性也导致了通证安全性的困扰。智能合约是业务逻辑的表现形式，追求的是灵活性；而通证是价值载体的表现形式，追求的是安全性。这两个需求的技术目标是彼此冲突的，极度安全一定不灵活，而极度灵活一定不安全。这就是为什么在 Ethereum 上，一旦项目方尝试自定义通证而不使用标准模板，就很容易因为智能合约漏洞而导致用户和项目方的资产损失。诸如 BEC 和 SMT 等通证出现的黑客盗刷事件，通证总数量一夜间被翻了数千万倍，严重的通胀导致通证价值归零，有些项目方可以挺过去，有些则直接夭折了。

因此，对于技术上安全和灵活性的博弈，我们理应优先保障通证的安全性，但现实依旧让人遗憾。由于 Ethereum 在设计之初，通证的概念还没有出现，所以并没有在架构层面为通证设计安全保障机制，而 Ethereum 之后的众多项目也是在通证价值慢慢体现的过程中研发出来的，在它们做架构设计时也没有考虑到通证的巨大价值及其高安全性诉求。所以我们看到，目前无论是第二代公链 Ethereum，还是第三代公链 EOS、Tron 和 Nebulas，各家在智能合约虚拟机上，从 Solidity 到 C++，再到 JavaScript 各有千秋，但在通证定义上的策略却空前一致，都没有为通证设计单独的层级，而是直接放在智能合约层和合约统一处理。而当我们试图使用智能合约同时满足业务逻辑的灵活性和通证的安全性时，必然顾此失彼，在此岸看彼岸，两头不到岸，也将使区块链的通证之路隐患重重。

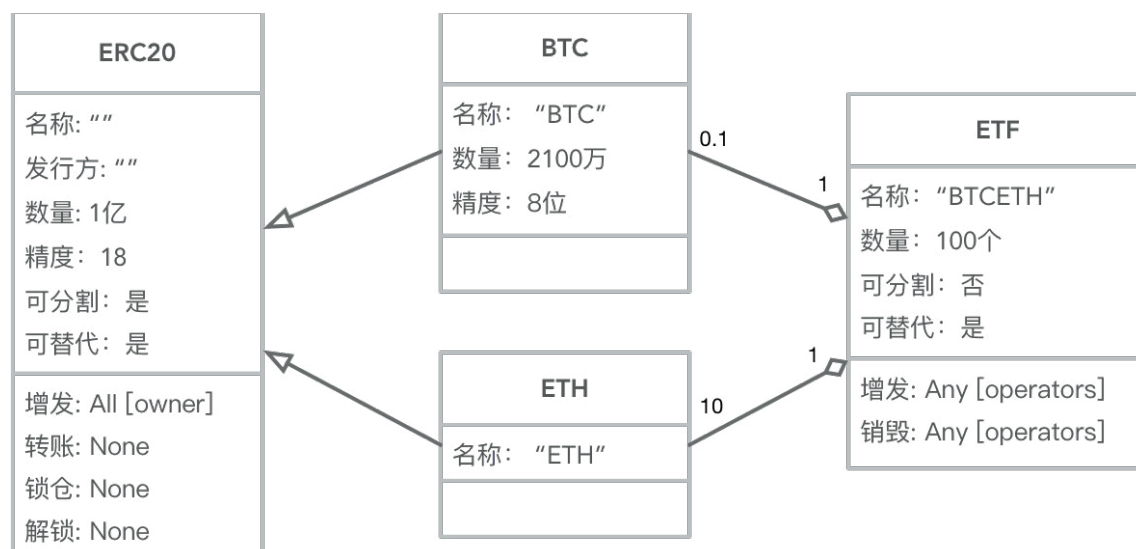
通证之路如何破局

很幸运，在经历了“区块链 3.0”的热潮之后，站在今天这个时间节点重看区块链的通证之路，我们能考虑得更加周全。为了妥善解决通证在政策和技术上灵活性和安全性的矛盾，我们鼓励各大主流公链平台将通证从智能合约中解耦出来，并在通证层提供支持权限定义的形式化通证建模语言取代图灵完备的智能合约编程语言，形成通证层和合约层的双层结构。

在这样的架构设计之下，合约层的灵活性将不受影响，而借助形式化语言易于做形式化验证的特点，通证层的技术安全性将极大提高，再加上通证层丰富的权限定义功能，平台上的通证发行方将有足够的工具来配合金融监管，保护通证政策上的安全性。当然，在这样的架构之下，通证定义将不如原来那般灵活，但形式化建模语言依旧保有一定的灵活性，这对于通证建模来说绰绰有余，在现实需求下真正做到鱼和熊掌兼得。

在认识到区块链通证之路的坎坷并找到解决方案之后，我们便开始付诸实践。在我们的新项目 Zerohm (zerohm.io) 中，我们便是基于这样分层解耦的基本思路来设计我们底层架构。为了充分解耦各种需求，我们首创了公链底层的四层架构方案，分为高性能可拔插的核心层，重安全可穿透的通证层，高扩展可升级的应用层，以及超易用可信赖的账户层，通过高内聚低耦合来满足各层级并不统一的技术要求。在通证层，我们在 UML (Unified Modeling Language, 统一建模语言) 的基础上，为通证量身定制了形式化建模语言 TML (Token Modeling Language, 通证建模语言)。

借助 TML，通证建模就和在 UML 定义类图一样简单。甚至和 UML 一样，通过拖拽的方式定义一个通证。举例说明，我们可以按照如下步骤构建下图所示的 ETF 通证：



由于操作符，操作符权限，通证间关系，基本属性都是形式化定义好的，我们可以很方便地对 TML 提供的这些能力做形式化验证，保证 TML 的高安全级别。在 TML 的高安全级

别保障下，基于 TML 定义出来的通证安全性也都有保障，这相对于通证和合约耦合结构下，每个自定义通证都需要支付昂贵的安全审计费用比，成本会得到大幅改善。而且 TML 支持升级，可以根据需求做安全地扩展，通证建模的灵活性也有保障。

结语

区块链正在经历着互联网曾走过的迷茫期，有人因失望而离场，有人因信仰仍在坚守。区块链的落地之路定然坎坷崎岖，前路不好走，愿与你同行，穿越这黎明前的黑暗！ ❖

尚书，Zerohm 联合创始人，Zerohm 白皮书主编。前星云链首席研发工程师，技术白皮书主编，链块学院技术顾问，公众号《区块勿语》作者，前 TOTFREE CTO，前阿里巴巴研发工程师，清华大学硕士。

公链设计与开发实践和经验

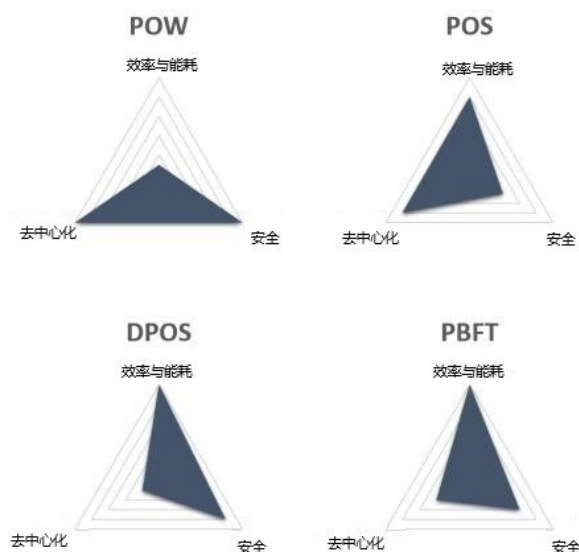
文 / 冯英飞

公链面临的问题

开发一条公链,对于任何一个技术团队都是一件极具挑战性的事情。假如糟糕的网络通信这种被动的恶意,还能让人接受,那么不择手段的双花、DDoS、智能合约攻击等等,让我们必须谨慎的设计每一个逻辑环节。

我们先来看看公链设计中的“不可能三角”问题:Scalability、Decentralization、Security。

- Scalability 可以理解为可扩展性,例如分片技术,也可以理解为网络能耗低,保证网络的性能最优,总的来讲就是优异的 TPS。
- Decentralization, 去中心化,如何保障网络的去中心性,这就要求该网络需要是一个对等网络,该网络中的机器的地位都是平等的,不存在任何特殊化的中心节点,同时为了保证该网络的去中心性,该网络需要是一个开放的无准入的网络,从而可以让人人都能够加入该网络,且该网络不会被一个或多个中心控制。
- Security, 安全性,是保障该网络足够安全,不能被坏人破坏。在一个开放并与经济利益挂钩的网络中,不仅会有好人购买机器加入这个网络,也会有更多的坏人企图希望通过破坏该网络获利。那么,如何在网络内部存在坏人的情况下,保证网络的安全性,这已经突破了传统意义上的安全架构,是安全设计的挑战。



PoW 的算力浪费以及单节点产块让它的性能低下, PoS 面对众多安全问题, 例如无厉害攻击, 远程攻击等, DPoS 的产块节点有限, 虽然有选举制度, 但是它的去中心化就是让人耿耿于怀, 而 PBFT 的同步通信让它的节点数注定会受到制约。

以上是现有共识的优缺点, 而我们 alabs 的 ADAG 公链团队, 选择了 Hashgraph 作为基本共识算法来设计我们的公链, 因为我们通过严谨的分析该共识算法, 并在此基础上做了重要的改进, 在不破坏算法严谨性的同时, 让它面对公网环境时, 也能具有很强的说服力。我们在它身上, 看到了画出一个尽可能最大三角的可能, 下面就谈谈它的性能, 去中心化和安全三个维度的情况。

ADAG 为什么选择 Hashgraph 共识

Hashgraph 的性能与去中心化

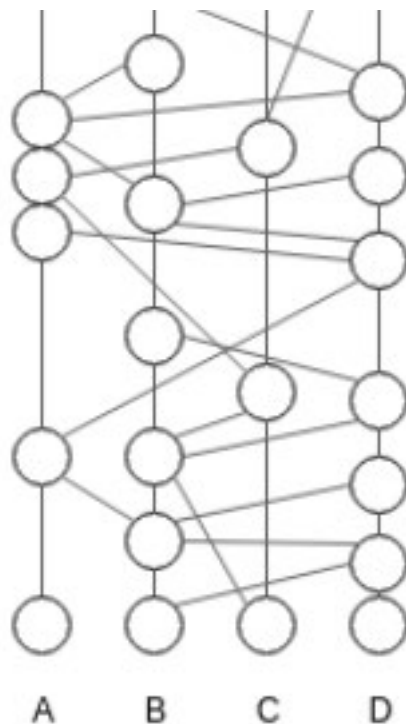
Hashgraph 算法是 swirllds 团队基于 DAG (有向无环图) 设计的一种完全异步的类 BFT 共识算法。该算法有两大特点:

1. Gossip about gossip (八卦协议)
2. Virtual Voting (虚拟投票技术)

事实上, 在选型之初, 我们很快意识到 Hashgraph 是一个类 BFT 算法, 而且不像 PoW, DPoS 那样, 完美的避开了拜占庭将军问题。这让它显得没有那么有魅力, 而就是这两个特性, 让我们看到了它的独特之处。

概括的说, 它们指的就是, 节点间随机的发生通信, 通信的载体叫 event, 图中的圆圈。A, B, C, D 为不同的节点。例如 A 与 B 通信, 通信的内容就是 A 节点知道的, 而 B 节点不知道的交易列表。假如 B 节点创建一个 event 来记录这次通信内容, 该 event 除了要包含交易, 还要有两个非常重要的字段, 即 Selfparent (B) 和 Otherparent (A), 由此随着交易的发生, 每个节点本地都会存在一个这样由通信历史组成的哈希图 (Hashgraph), 而每个节点根据本地的 Hashgraph 都能够独立的计算出一个块, 而且块中包含的交易序列以及顺序都是一致的。

节点间的通信我们叫 gossip, 而 gossip 的内容简单来说就是我知道的, 而你不知道的, 所以很形象的称为八卦协议。而 self parent 和 otherparent 又记录下所有的历史通信记录, 这是算法再进行 witness 和 famouswitness 选举时最重要的信息 (具体算法可以参照 swirllds 白皮书), 有了这些信息, 节点就可以独立的计算并获得块 (产块), 并保证块的一致性 (这叫做虚拟投票)。所以你会发现, 通过这种看似随意的通信, Hashgraph 避免掉了传统



BFT 类共识中常见的消息传递风暴 (N^2)，很大程度的降低了网络能耗，而且能够很好的支持并发。用 Hashgraph 发明者的话来说就是：“Hashgraph 具备投票算法的一切优点，然而避开了它的最大缺陷。”

从算法中我们也可以看到，这是一个异步的类 BFT 算法，所有的节点是对等的，不存在其他身份的节点，所以它是一个去中心化的系统。当然作为 BFT 类共识，它依然要面对拜占庭问题，即 $1/3$ 恶意节点问题，我们需要额外的奖惩机制来约束网络环境。

Hashgraph 的安全性

首先 Hashgraph 作为一个完全异步的拜占庭容错，这意味着它并不对消息在互联网上传递有多快做任何假设。该功能使得它能够抵御 DDoS 攻击、僵尸网络。另一个经常被讨论的问题就是 Hashgraph 能否经受 Sybil 攻击，即攻击者通过创建大量假身份来破坏对等网络的信誉系统，并利用它们获得不成比例的巨大影响力。

我们设计了一种让所有共识参与者共同维护“共识白名单”的机制，让所有参与者能够验证彼此对同一个块的签名，再配合奖惩机制，让它能够应对假身份攻击的危险。

该算法能够通过定义网络中的大多数为 $2/3 * N \sim 3/4 * N$ ，从而能够抵抗 $1/3 \sim 1/2$ 的恶意攻击，当然大多数越接近 N ，对性能会有负面的影响。

Hashgraph 存在的问题

当然 Hashgraph 并不是完美的，作为一个完全异步的 BFT 类共识，它在公网环境下，能

够支持多少个节点？完全的异步让它对网络环境的要求没有那么苛刻，我们知道它最终能够达成一致，但是，如果这个时间窗口过长，那么它将无法面对很多业务场景，而且，作为 BFT 类共识，它必须知道全网节点数 N ，而这些都是公网环境下无法保证的。于是我们设计了基于 Hashgraph 的快速同步功能以及共识节点的动态验证集策略。

在传统链式结构中，大家对这两个功能并不陌生。像以太坊的快速同步功能，以及 casper 中的动态验证集。基于链式的数据结构让我们很容易能够想象，这两个功能如何去实现，所以我们需要把 Hashgraph 链式化。

如何解决

链式账本结构，它更适应于表示不可变的事务的有序列表。在我们的系统里，交易顺序由 Hashgraph 一致性算法控制，但是最终的“块状”交易被映射到由块组成的线性数据结构。每个块包含交易的有序列表，前一个块的哈希，对应的应用程序状态（statehash），以及来自动态验证集的签名集合。

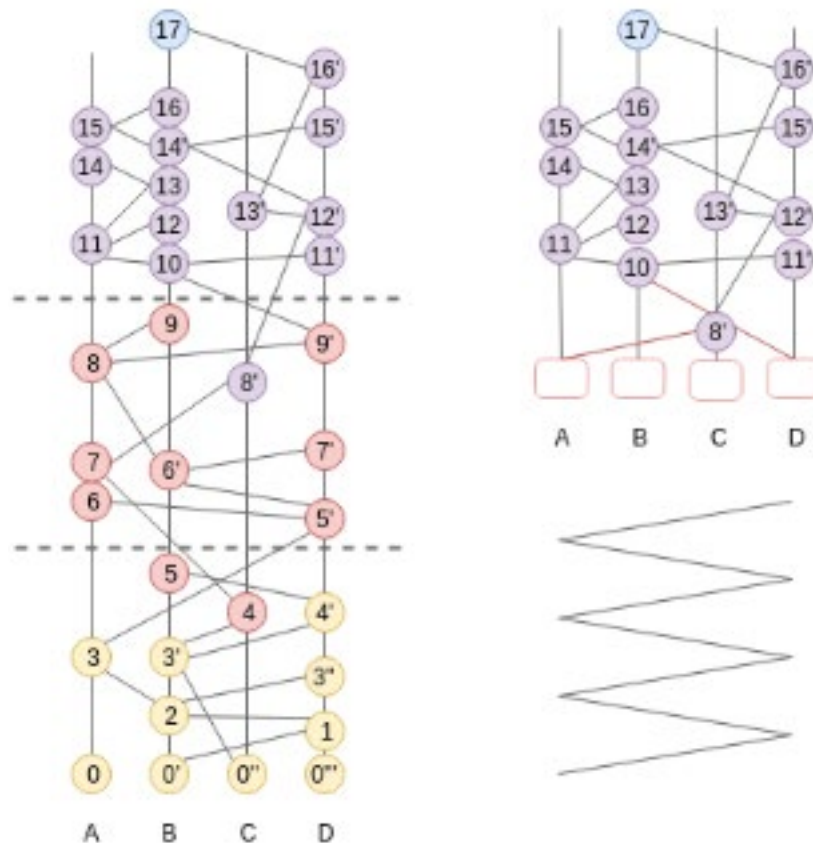
在区块链世界里，任何一个共识系统的输出都是一个有序的交易列表。我们最终决定使用链式结构来建模最终数据，是因为它是高效的，具有良好区块链兼容性（分片，跨年，layer2 等）。由批量，有序的交易序列，hash，签名，组成的线性数据结构让它很容易验证很多事务。

Hashgraph 是一种基于同义数据结构的，优秀的一致性算法。然而 swirlds 团队也仅仅是给出对交易进行排序的一致性算法，然而哈希图数据结构在表示线性交易序列时并不容易使用。它是一个有向无环图（DAG），其顺序必须通过一些复杂的一致性函数来提取，为了验证给定交易的一致性索引，必须重新计算 hash 图的子集上的一致性方法。而链式结构不需要进一步的处理来提取交易的有序序列，并且通过简单的加密原语足以验证块。

如下图所示，我们根据 swirlds 算法中的 round 以及 roundreceived 概念，将 Hashgraph 映射为链式结构。在链式结构上，我们通过设计新的数据结构来截断 Hashgraph，以便支持快速同步，并在不同的 round 中支持不同的动态验证集，这让我们的公链在面对恶劣的公网环境，能够有很好的适应能力。

同步功能，我们设计了两种同步方式，一种是基于全图谱的全数据同步模式，另一种是基于某一个特定的“frame”，以及对应的 block，对应的账本 snapshot 的快速同步模式。快速同步功能虽然并不陌生，但是它对与一个完全异步的 BFT 类共识就显得格外重要了，因为由于网络原因，某些节点会落后于当前大多数节点，在一些特定得应用场景中，我们鼓励节点积极的使用快速同步，这样可以有效得控制并防止异步过程中，全网数据状态不一致的时间窗口越来越长。当然需要全数据的节点可以通过不断的 gossip，来同步自己的全数据，然后不断产块，来追赶最新的数据，方式是灵活的。

动态验证集功能，公网环境下，我们很难控制节点的行为，当然有效的奖惩机制是个不错



的选择，这是另外一个很大的话题，这里暂且不说。节点的添加与删除，我们通过设计一种特殊的交易类型来实现。这样的交易会像普通交易一样被封装在 event 里，并 gossip 到全网中，这样根据一致性的共识算法，这个交易会被大多数节点放到同一个块里，那么它们就可以在逻辑上一致性的去维护一个共识白名单。我们的策略是一个节点可以随时在线或者断线，但是我们将它在逻辑上与对应的 round 周期相对应，这样在投票算法中，就不会破坏共识算法的严谨性。

回到文章开始的地方，我们之所以选择 Hashgraph+ 链式账本结构的方式来开发我们的公链，就是因为，通过严谨的研究与分析，以及核心创新功能的开发，我们在 ADAG 身上看到了很多可能。

关于 layer2 的一些思考

在区块链的技术栈中，跨链和 layer2 的方向正被越来越多人重视，相关技术的发展也越来越成熟。我们团队也一直很重视 ADAG 对相关技术的支持。如果不是很严谨的说，Inter-Blockchain Communication (IBC) 是关于一个链充当另一个链的轻客户端，而为一个链式结构构建一个清客户端明显要比在一个 hash 图上构建要简单的多。为了在块链之间实现相互操作，几个倡议已经被提出，如 Cosmos、Polkadot 和 EOS。而 ADAG 的链式账本可以与这些网络体系结构很好的集成。

Layer2 的很多技术都可以通过智能合约来实现，ADAG 采用通用的链式账本结构，能够轻松的支持智能合约，当前支持 EVM，支持 WASM 的分支也在开发中。

分享与建议

公链并不是万能的，它需要适配自己的业务场景。这对技术选型，未来的成长至关重要。

假如你使用的共识算法并不成熟，而你又希望开发一条公链，那么建议团队在全力投入之前，先确定你的共识算法的安全，容错边界，以免搭建空中楼阁。

无论中本聪类共识，还是 BFT 类共识的容错边界都是有限的，所以也不要对公链初期的准入与控制行为耿耿于怀。

篇幅原因，文章中没有对经济模型和奖惩机制做更多的介绍，而它们与共识一样对公链的成败至关重要。❖

冯英飞，Alabs ADAG 主链项目技术负责人。区块链领域开发专家，致力于区块链主链技术的研发及应用落地，毕业于江南大学，曾工作于 futurmaster 中国，长期从事分布式计算，网络通信等领域的研发工作。

公链设计与开发细节揭秘

文 / 钟文斌

背景介绍

2018 年被称为“公链元年”，各种形形色色的公链如雨后春笋，一时间几乎所有项目都在开发各种各具特色的公链。Qtum 量子链作为国内最早的一批公链之一，在 2017 年 9 月上线了其主干网络，其全球全节点数仅次于以太坊和比特币，成为全球第三大去中心化区块链网络。

Qtum 兼容比特币和以太坊这两大生态，可以说集合了主流公链开发的所有技术，并且在此基础上提出并实现了多项技术创新。本文以 Qtum 为例，介绍了公链设计和开发需要注意的诸多细节，供所有公链开发者参考。

Qtum 量子链的基本特性

Qtum 是首个建立在 UTXO 模型之上，采用 PoS 共识机制和去中心化治理模式，且兼容多虚拟机的智能合约平台和价值传输网络。

为了读者有直观理解，先罗列 Qtum 区块链网络的基本特性和参数：

- 底层模型：UTXO（Unspent Transaction Output）
- 区块大小：2MB
- 平均区块间隔：144 秒
- 共识机制：PoS（Proof-of-Stake，权益证明机制）
- 链上理论最大 TPS：70~100 笔交易 / 秒
- 初始发行量：100,000,000 QTUM
- 当前挖矿奖励：4 QTUM，每四年减半
- 挖矿奖励总额：约 7,884,000 QTUM（作为区块生产者的奖励）
- 基础代码框架：比特币 Bitcoin core；
- 支持智能合约，支持 EVM，x86VM 等多虚拟机；

- 支持去中心化链上治理机制；

这些特性和参数的确定并非一时兴起，而是 Qtum 创始团队在综合考虑公链生态的现状、应用场景、技术可行性以及未来发展方向等因素之后确定的。在正式开始介绍各部分设计前，不妨先回顾一下 Qtum 设计初衷，有助于理解后续的许多实现细节。

Qtum 量子链设计初衷

Qtum 量子链的想法诞生于 2016 年，当时的比特币网络已经稳定运行了近 8 年，期间比特币账本从未出过一笔错账，足以看出比特币 UTXO, PoW 等设计的优越性。当时大部分项目都是对比特币的简单修改，本质上还是中本聪最初设计的雏形。然而比特币的美中不足之处在于其非图灵完备的脚本，无法支持真正意义上的智能合约。

Qtum 创始团队最初的想法很简单：让比特币支持智能合约。

然而，为什么团队创造了一条新的公链，并且具有如此多与比特币完全不同的特性呢？主要出于以下几点考虑：

1. 出于安全性、稳定性、匿名性和可扩展性等方面考虑，选择 UTXO 模型，而不是账户模型；
2. 比特币的升级需要 Core 团队和比特币社区的共识，如此重大的升级几乎不可行，因此需要一条新公链；
3. 支持智能合约需要虚拟机，EVM 具有良好的生态，首先兼容；
4. 不重复造轮子，并保持和比特币及以太坊两大社区的兼容性，复用比特币和以太坊部分代码；
5. EVM 有其固有的缺点，因此 Qtum 也在自主研发更具兼容性的 x86 虚拟机；
6. 同一条链上支持多种虚拟机，需要设计可扩展的账户抽象层；
7. PoW 共识的缺陷，兼顾去中心化，因此采用 PoS 作为共识机制；
8. 已有的区块链项目都缺乏有效的链上治理模式，需要设计一套去中心化治理机制。

简而言之，Qtum = UTXO+PoS+ 智能合约 + 多虚拟机兼容 + 去中心化治理，这与开篇提到的：“Qtum 是首个建立在 UTXO 模型之上，采用 PoS 共识机制和去中心化治理模式，且兼容多虚拟机的智能合约平台和价值传输网络”完全吻合。

上述设计涉及到较多技术实现细节，接下来我们详细解释。

Qtum 技术实现细节

账户抽象层 — UTXO 和比特币脚本如何支持智能合约？

UTXO vs. 账户模型

UTXO (Unspent Transaction Output, 未花费的交易输出) 是比特币采用的底层账本模型。而以太坊则采用了账户模型。对于普通人来说, 后者相对好理解, 就是从一个地址到另一个地址, 一笔 ETH 的传输, 类似银行账户。而比特币交易则相对复杂, 它是由若干 input 和 output 组成的, 每个 output 包含特定”锁定脚本”, 该脚本规定了该 output 的花费规则, 提供符合规则的”解锁脚本”即可花费该 output, 作为下一笔交易的 input。而尚未被花费的 output 就是 UTXO。比特币网络上的交易都是由 input 和 output 串联起来的。UTXO 的优点在于:

- 安全性(解决双花问题的逻辑和处理方式都及其简单, 且支持原生多重签名)
- 匿名性(多人多出的交易, 钱包可生成任意多个地址, 每次生成新的找零地址)
- 可扩展性(可并行处理, 生成离线交易无需像以太坊一样维护 nonce)
- 轻钱包(SPV, 支持轻钱包去中心化验证交易的合法性, 特别适合移动设备)
- 兼容性(兼容诸如闪电网络, 跨链原子交换, 隐私交易等其他基于 UTXO 的项目所采用的技术)

UTXO 虽然有诸多优点, 但当时基于 UTXO 模型的各种项目的底层脚本语言都是非图灵完备的, 并且与以太坊虚拟机所采用的账户模型无法兼容。为此 Qtum 开发账户抽象层 (Account Abstraction Layer, AAL), 实现 UTXO 模型与账户模型的适配。这种分层设计实现了底层账本和上层智能合约的完全解耦, 也使 Qtum 后续兼容多种虚拟机 (EVM, x86, WASM 等) 成为可能。

AAL 要实现的工作其实很简单: UTXO 模型 <-> 账户模型。由于篇幅有限, 本文只对其基本原理和遇到的问题做简要解释, 感兴趣的读者可以进一步阅读《深度解析 Qtum 量子链账户抽象层》, 文章从代码层面对 AAL 进行了详细剖析。

扩展比特币脚本 — UTXO 模型 -> 账户模型

比特币的 UTXO 模型采用了一套非图灵完备的脚本语言, 该脚本自带的操作码并不支持合约交易。Qtum 在比特币脚本的基础上增加了 3 个操作码:

1. OP_CREATE : 创建智能合约;
2. OP_CALL : 调用智能合约 (向合约发送 QTUM);

3. OP_SPEND : 花费智能合约中的 QTUM

在产生新区块时，除了对交易脚本做常规的检查外，还需要检查是否包含上述的操作码。OP_CREATE 用于向 EVM 传递合约字节码。OP_CALL 将 data、gasPrice、gasLimit、VM version 等运行智能合约所需的关键参数通过交易脚本发送，最终传递到 EVM 中。OP_SEND 则用于合约执行结果的 UTXO 转换，稍后解释。

通过引入上述三个脚本，Qtum 的 UTXO 模型具备了识别和处理智能合约相关交易的能力。

从 UTXO 获取的合约如何在 EVM 中运行？— 账户模型 -> UTXO 模型

合约的执行会引起状态改变，对于合约的状态，Qtum 沿用了 EVM 的定义，所以能兼容所有的符合 EVM 规范的智能合约。

从上述 OP_CREATE 或 OP_CALL 提取出合约交易参数之后，即可构建运行环境。合约的运行过程基本采用了 EVM 的引擎，为使其与 Qtum 区块链进行正常交互，Qtum 完成了以下工作（这里只截取部分），供公链开发者参考：

1. 针对每个独立交易构建运行环境，最大限度的把不同交易的合约执行过程隔离开，避免合约执行过程中的交叉影响；
2. 实现 EVM 的永久存储，当区块成为孤块或断开时，可以实现状态的回滚；
3. 实现 EVM 和 Qtum 区块链间的接口，使 EVM 能够获取诸如 BLOCKHASH, COINBASE, DIFFICULTY, BLOCKNUMBER, GASLIMIT 等区块基本信息；
4. 实现已部署在链上的合约间的相互调用；
5. 实现合约地址获取自身余额的功能；
6. 为 EVM 生成 debug 信息，方便调试智能合约；
7. 重新设定各操作码的 gas 价格，因为 QTUM 和 ETH 的价格差异较大，且 EVM 的 gas 模型设计略不合理，因此需要重新优化；
8. 支持自毁操作码，使合约可以自毁；
9. ……

感兴趣的读者可以阅读《[Qtum 设计文档](#)》获取更多设计方面的细节。

在解决了上述问题之后，从 Qtum 交易传入的智能合约代码即可在 EVM 中成功运行了。该实现理论上适用于所有基于 UTXO 的项目。

运行的结果会引起状态改变，这时需要使用上述的 OP_SPEND 用于花费合约中的余额，上

面已经提到，无论是比特币还是 Qtum，都是通过私钥”解锁“UTXO 脚本来花费 UTXO 余额的，而 EVM 的执行涉及不同账户之间的转账，所以需要通过 OP_SPEND 实现这些转账到 UTXO 模型交易的转换，最终转化为 Qtum 标准交易脚本（对比特币交易脚本不熟悉的读者建议阅读《Mastering Bitcoin》）。

账户抽象层的安全性

上述的 UTXO 和账户模型的互相转换虽然实现了 Qtum 区块链与 EVM 的交互，但 AAL 也引入了一个安全问题。由于合约地址本身也是一个 Qtum 的合法地址，并且底层都是 UTXO 模型，这就意味着一个合约可以拥有多个 UTXO。这本身没有问题，但却给攻击者提供了 DoS 攻击的可能性。攻击者可以通过给合约发送多个小额的 UTXO，然后一次性用 OP_SPEND 花费大量 UTXO，从而导致区块大小超过限制，新区块无法被接收，使得网络无法产生新的区块。这是 UTXO 与智能合约相结合所带来的必然问题。

Qtum 的解决方案是在共识层面做优化，规定合约地址只能拥有一个 UTXO。每次合约运行结束时，都会将原来的 UTXO 和新加入的 UTXO 进行合并，从而保证合约地址始终只有一个 UTXO。Qtum 称之为“Condensing transaction”。

为了保证合约状态以及 UTXO 的共识，Qtum 的区块头除了包含与比特币相同的字段外，还需要额外加入 hashStateRoot 以及 hashUTXORoot 两个字段，感兴趣的读者可以阅读 Qtum 的源码。

账户抽象层带来的好处

比特币的 UTXO 模型相较于以太坊的账户模型有诸多优越性，以太坊有一套图灵完备的语言，可以实现比较复杂的智能合约逻辑。两者都有强大的生态作为支撑，并且聚集了区块链行业最优秀的开发者。Qtum 选择不重复造轮子，而是在比特币和以太坊的基础上开发了账户抽象层 AAL，打通了两个原本分离的生态。这一设计带来的好处包括：

1. 获得与比特币一致的稳定底层基础设施（UTXO）；
2. 兼容比特币后续所有包括性能上和安全性上的升级；（目前已同步升级到 0.16 版本，正在适配 0.17）
3. 兼容现有智能合约生态，以太坊上的智能合约可以零成本迁移；（QRC-20，QRC-721 等均兼容）
4. 比特币和以太坊的开发者自动成为 Qtum 的开发者；
5. 兼容所有基于比特币和以太坊虚拟机的技术，比如闪电网络，雷电网络，plasma，加密隐私，原子交换，分片等等；
6. 兼容以太坊上所有智能合约开发工具（当然有些需要做 RPC 的适配，Qtum 已经实现了这样的适配），降低开发者学习成本。

小结

Qtum 首创的账户抽象层 AAL 实现了 UTXO 模型到账户模型的适配，从技术层面打通了比特币和以太坊生态，并未后续兼容多种虚拟机（如 x86，WASM）提供了可能，其设计思想和实现细节值得公链开发者参考。

PoS 共识机制

为什么需要挖矿？

说到共识机制，我们首先从字面上回顾一下什么叫“区块链”，所谓区块实际上是一段时间内交易的集合，每个区块都有一个指向上一区块的哈希指针，从而组成了一条链。所以简单来说区块链就是一个不可篡改的分布式数据库，那为什么需要挖矿呢？

这里以 PoW 挖矿为例。首先，区块链主要分为：公有链、私有链和联盟链。它们的核心区别在于记账权，公有链是去中心化的，赋予网络上的每个节点记账的权力，而私有链和联盟链的记账节点为少数几个指定节点。只有公有链才需要挖矿，其目的在于：

1. 安全性：依靠区块奖励和手续费激励节点记账，去中心化地维护区块链网络的安全性；
2. 随机性：保证选出的记账节点的随机性，否则固定的记账节点很容易被 DoS 攻击，出现单点故障；
3. 代币分发：挖矿也是一个代币分发的过程，从而把币随机地分出去，而不是只在少数人手中。

整个比特币的精髓就在于它的激励机制，但 PoW 共识机制也并非没有问题，比如：

- 加入网络的门槛较高，当前需要花费很多钱买矿机才能成为比特币的全节点进行挖矿，普通设备几乎没有机会；
- 越来越趋性中心化，大部分的挖矿所得被少数几个大矿池占据；
- 能源消耗巨大，矿机耗电甚至超过某些小国的耗电量（当然这些电量支撑起这么大的网络并不算多，但我们应该寻找更好的方式替代）。

基于这些问题，2012 年时 Sunny King 提出 PoS（Proof-of-Stake，权益证明机制），其原理这里不做赘述，感兴趣的读者可以参考[这篇文章](#)。简单来说，只要用户持有该网络的 Token，就可以参与记账。另外，代币持有人作为利益相关者也更有动力维护网络的安全。PoS 机制同样也能达到比特币的随机性、安全性和代币分发等功能，而且相比于 PoW 来说显得更加去中心化（前提是代币初始分发足够分散）。中本聪在设计 PoW 共识机制时并没有想到会有矿机的出现从而使得比特币越来越中心化，而 PoS 的挖矿根本不需要使用矿机，只需要一台普通电脑、树莓派就可以参与。

除此之外还有几种共识机制比如 DPoS（代理权益机制）选出一些代理人来进行记账。pBFT，dBFT 等则是改进后的拜占庭容错机制解决方案。这几个共识机制虽然效率较高，但都比较中心化的，容易导致一些中心化的问题。对于计算节点比较少的网络，甚至一个节点挂掉都有可能导致整个网络瘫痪，但这些在比特币或者 Qtum 上都没有发生过。

Qtum 项目出于去中心化程度和能源消耗方面的考虑，选择采用 PoS 共识机制。

关于 PoS 的误解

关于 PoS 共识机制，大多数人的理解还停留在其最原始的版本，即 PoS1.0。实际上 PoS 共识机制已经经历过 3 次大的迭代：

- PoS1.0：依赖“币龄”，长期不在线，双花问题等；
- PoS2.0：移除“币龄”，增强安全性；
- PoS3.0：针对“short-range”攻击，采用区块时间和交易时间确认 UTXO 的“年龄”。

“币龄”这个概念其实早就从共识机制中移除，现在的 PoS3.0 系统已经可以解决很多初始版本遇到的安全性问题，甚至在抵抗 51% 攻击上比 PoW 系统更具优势（[为什么？](#)）。

PoS + 智能合约带来的问题

Qtum 没有直接采用 PoS3.0，不是因为共识机制本身有问题，而是由于 PoS 和智能合约的结合可能带来潜在的攻击可能。攻击者可以通过支付比较昂贵的 gas 发起一系列“垃圾合约”。虽然这将消耗很多的 gas，但由于矿工可以获得交易中的 gas 作为奖励，只要攻击者的合约设计得当，就能保证其他节点无法正常处理该交易，而攻击者则可利用先验知识成为区块的生产者，从而将攻击成本全部收回，实现零成本的 DoS 攻击。

但是为何现有用 PoS 共识机制的区块链项目就不会有这个问题，因为它们大部分只支持非图灵完备的脚本语言，无法支持上述的“垃圾合约”（实际上需要循环操作）。但是在 Qtum 上既支持 PoS，又支持智能合约，使上述攻击成为可能。

Qtum 的解决方案是：通过和其他节点分享收益并将收益延迟化，增加攻击的成本。

Qtum 在 PoS3.0 的基础上修改激励返还机制，实现了 MPoS（Mutualized Proof-of-Stake）。

MPoS 的基本原理：每个区块奖励由 10 个矿工平分，其余奖励延迟 500 区块。即 1/10 区块奖励立刻获得，其余 9/10 奖励在 500 个区块之后连续 9 个块中获得。挖矿奖励 = 区块所得 + 手续费 + 运行智能合约 gas 费用。这个简单改进在不改变 PoS3.0 的核心逻辑的前提下，使攻击者无法预测获得区块奖励的多少，也无法立即获得区块奖励，从而极大提高了发动上述“垃圾合约”攻击的成本（仅存在理论可能性，实际操作中完全无法实现）。

PoS 的实现细节

Qtum 的底层代码源于比特币，其采用的是 PoW 共识机制。如何将 PoS 应用到其中，并进一步实现 MPoS 呢？Qtum 团队做了以下实现，供公链开发者参考：

实现 PoS 区块验证和挖矿功能。具体细节如下：

- 同时支持 PoW 和 PoS 区块，因为在初始阶段需采用 PoW 生成初始代币，在一定高度后变为纯 PoS；
- 每个区块的 stake modifier 都会发生改变；
- coinage（币龄）不能改变 PoS 哈希值；
- 交易费不会作为共识规则；
- 需要最近的 500 个区块用于验证 PoS，这里区块不宜太多，否则数据量太大难以处理；
- 没有币龄概念，只需要币达到一定的确认数（如 500 个区块）就可以进行 PoS 挖矿；
- 增加 PoS 相关的共识参数：区块签名，区块类型（PoW 还是 PoS），前一个 stake 位置，以及 staking 交易创建时间等；
- 区块出 coinbase 交易外（实际上 coinbase 交易已经没有实际作用），还需要增加 coin stake 交易，用于生成和分配区块奖励；
- 接收区块时要检查区块头，因为处理 PoS 区块时，需要处理重复哈希和孤块问题；
- Staker 只需要才成功发现 stake 后才继续处理交易，否则将导致交易被多次处理，特别是对于合约交易，多次处理可能会浪费大量时间，这点与 PoW 区块的交易处理略有不同；
- 由于 AAL 引入了智能合约，PoS 区块的交易不仅需要估计交易本身自带的 sigop 的大小，还需要考虑由于智能合约产生的“condensing 交易”所引入的额外交易，否则将导致有可能生成超过区块大小限制的交易，影响系统安全性。

关于 MPoS 的实现细节包括：

- 发送给 MPoS staker 的区块奖励必须来自于 staking 区块；
- stake 交易必须包含至少 10 个输出（outputs），1 个给区块创建者，9 个给 MPoS staker；
- stake 交易的第 1 个 10 个 outputs 是共识相关的，必须准确地按照 MPoS staker 挖矿的区块高度正确地排列，且创建者为第一个 output；
- stake 交易可以包含额外的 outputs，例如，将一个大的 staking UTXO 分成多个 UTXO，input 也可以包含多个输入，比如讲零散的 UTXO 自动合并成较大的 UTXO。在第 10 个 output 后，对于 output 不应该有任何特殊的共识规则，只要交易的 output 值不会

超过输入值 + 区块奖励 + 费用；

- 前 500 个 PoS 区块不使用 MPoS，因此区块的创建者获得全部的奖励，不需要使用前面的规则。

更多实现细节请参照 Qtum 原始设计文档及源码。

小结

Qtum 是首个采用纯 PoS 共识机制的智能合约平台，并解决了其潜在的安全隐患。公链开发者在采用新的共识规则时应该首要考虑不同的技术组合带来的安全性问题，不能完全照搬照套。

分布式自治协议

基本原理

区块链治理要解决的是在一个去中心化网络怎样对软件系统进行升级、迭代等等问题。比特币之前分叉成 BTC 和 BCH，分歧仅在于区块的大小（当然这背后包含很多利益纠葛，但技术上只是一个参数的分歧），正因为比特币是一个没有完整链上治理机制的去中心化网络，所以导致这个争论持续了很长一段时间。又比如 BCH 分叉的算力之争。分叉并不能说是绝对意义上的坏事，但基本的区块链参数完全可以在无分叉的情况下实现升级，因此 Qtum 设计了 DGP（Decentralized Governance Protocol，分布式自治协议）。

影响分叉的因素大致可分为以下三类：

- 算法、功能的改变（共识算法、加密算法、交易脚本、虚拟机）
- 策略、参数改变（区块大小、出块时间、交易数量、Gas 策略）
- 关键漏洞的修复、回滚（DAO、Parity 多重签名钱包）

策略方面的因素其实是最容易达成共识的，另外两类有时必须通过分叉来解决。

DGP 本身的框架是通过若干部署在创始区块的智能合约来实现的，其基本的治理结构是这样，在整个社区内部的矿工、区块生成者和持有者都是区块链治理的参与者，通过投票去完成治理的过程。最终让区块链能够实现自我管理、升级和迭代的系统。

DGP 实现

DGP 的实现需要某种可编程技术，UTXO 和 EVM 其实都提供了这种特性，所以理论上有两种实现方式：

- 一是基于比特币交易脚本，通过在交易脚本上实现协议逻辑。然而由于比特币脚本非图灵完备，这种实现会比较复杂；

- 二是基于智能合约，具有图灵完备的可编程能力，可以灵活实现复杂的逻辑。

Qtum 选择了后者。

DGP 核心逻辑的实现，是由一系列的智能合约（包括框架合约，特性合约）组成。

DGP 框架合约实现以下功能：

- 提案和投票 – 每个参数改变包括内部治理席位管理需要先被提议，然后对它进行投票。如果投票符合所选择的条件，则该提议被接受，并执行该操作。投票使用 “msg.sender” 计算，这样公钥哈希地址或合约地址都可以作为参与者参与投票。
- 治理席位管理 – 可以添加和删除参与者，也可以修改治理席参数，比如一个提案被接受需要多少个治理席位同意，添加一个治理席位需要多少个治理席位同意等等。
- 发送正确格式的数据给 DGP 特性合约
- 允许自己被禁用，这样在不使用硬分叉的情况下就不能进行进一步的 DGP 修改（防止重大的漏洞或问题）
- 一次只允许一个提案，提案只能由参与者提出。每个提案的有效期不超过 5000 个区块。每个提案在到期后，或者在投票结束后，完全可以拒绝或者批准。
- （可选功能）维护管理员的列表，管理员可以删除提案，管理员也可以是唯一允许添加提案的人。

DGP 特性合约则更简单，它只需要完成以下两件事情：

1. 只从合适的 DGP 框架合约接收信息 / 数据（使用 msg.sender）；
2. 使用 “SSTORE” 以一种标准化的形式存储共识数据，以便区块链在不运行 EVM 的情况下就能在 RPL 中检索和解析该数据。

区块链核心代码在共识过程中执行协议的智能合约，获得当前的共识状态。同时它能够通过 Transaction 完成区块链网络的状态转换，升级无需区块链网络软件更新。理论上，采用了图灵完备的智能合约可以实现任意复杂度的协议设计，甚至是区块链的核心协议，如共识部分的代码等；权衡效率和安全性方面考虑，当前协议仅适用于在安全范围内对特定参数进行更改，同时对参数生效时间采取一定的时间限制。未来 DGP 可以不断迭代，实现更多更复杂的治理。

回到具体实现，创世块嵌入了常见的区块链参数治理的智能合约，每个治理的主题都由独立的框架合约控制（模板），这意味着每个功能有独立的治理、授权机制以及内置限制条件 Block size, Min GasPrice, Block GasLimit, Gas Schedule。此外 DGP 合约还具备自毁功能，能在提案治理上发生意外时启动，治理参数退回到默认状态。更多实现细节请参阅这篇来自 Qtum 社区开发者的[分析](#)。

小结

链上治理往往是公链开发者容易忽略的一环，然而其实际上非常重要，是去中心化网络后续升级迭代的重要基础设施。Qtum 的分布式自治协议是行业内真正意义上的链上治理机制，其设计思想可以引入到很多其他项目中。

x86 虚拟机

Qtum 已经兼容 EVM，为何还需要 x86VM？

虽然 EVM（以太坊虚拟机）是当下最流行的智能合约虚拟机，但正如绝大多数新生事物一样（比如 Javascript），它存在诸多缺点。并且由于它的设计比较非主流，很难有主流的编程语言能够移植到 EVM 上。这种设计可以说对于近 50 年来的大多数编程范例来说都不太友好。

这里罗列一些其明显的缺点（详细描述可以参考[这篇分析](#)）：

- 编程语言局限性（Solidity）
- 缺少标准库
- 256bit 整数，大部分处理器不能原生支持，运行效率降低
- Gas 模型不合理，难以估计 Gas 消耗
- 生成的 bytecode 较大，浪费区块存储资源
- 难以测试和调试

Qtum-x86 虚拟机的设计目标

正因为 EVM 存在诸多缺陷，Qtum 决定开发自己的虚拟机。x86 虚拟机兼容了被工业界充分验证过的 x86 指令集，对基于 x86 架构之上的所有技术和基础设施都有很好的兼容性。Qtum-x86 的基本特性包括：

- 支持多种主流编程语言：C/C++/Go/Rust 等等
- 丰富的标准库，提高开发效率
- 更加优化的 Gas 模型 – 为标准库函数设定合理的 gas 模型，可以准确估计 gas 消耗
- 解锁 AAL 的强大功能 – 支持合约的 P2SH 交易，segwit 交易等
- 冯·诺依曼结构，加强版的智能合约 – 代码即数据，多任务协作，支持中断和恢复
- 第一类预言机 – 无需运行合约即可获得某些合约数据

- 区块链动态分析 – 更全面地分析区块链状态
- 选择性数据存储 – 节省宝贵的区块链上资源
- 清晰的依赖关系树 – 有可能并行运行智能合约，降低 gas 费用

由于 Qtum-x86 还没正式发布，其具体设计文档暂时无法透入太多细节。但其设计目标的描述可以参考[这篇文章](#)。

小结

Qtum 的账户抽象层使兼容多虚拟机，使集成 x86 成为可能。未来 Qtum 将同时支持多种虚拟机，充分发挥不同虚拟机的优势。虚拟机是支持智能合约的区块链系统不可或缺的部分，Qtum 的 x86 虚拟机无论从设计上还是实现上都对公链开发者有极大的参考价值。其发布后也可输出到其他区块链项目中。

Qtum 的可扩展性

上面已经介绍了 Qtum 区块链的底层设计和实现。由于 Qtum 兼容 UTXO 以及 EVM，使得很多已有的区块链相关技术都能够很容易地移植到 Qtum 上，这些都得益于 Qtum 优秀的架构设计。

下面的一些技术已经成功在 Qtum 上实现，感兴趣的读者可以在 <https://github.com/qtumproject> 上查看所有实现细节。

- Qtum 闪电网络
- Qtum 支付通道
- Qtum-IPFS
- Qtum-Plasma
- qtumjs
- QRC20, QRC721 等（与 ERC20, ERC721 保持一致）
- Qmix（类似 Remix）
- Qrypto（类似 metamask）
-

总结

Qtum 是首个建立在 UTXO 模型之上，采用 PoS 共识机制和去中心化治理模式，且兼容多

虚拟机的智能合约平台和价值传输网络。

- Qtum 创新性的账户抽象层打通了以比特币为代表的 UTXO 以及以太坊为代表的智能合约生态，兼容大部分主流区块链项目的技术，可扩展性极高；
- Qtum 采用了 PoS 共识机制，在保证去中心化的前提下解决了 PoW 固有的能源消耗问题，并在此基础上进一步提高安全性，支持灵活的智能合约；
- Qtum 提出的分布式自治协议完成了去中心化链上治理，实现无分叉的区块链升级，未来可应用于更多场景；
- Qtum-x86 虚拟机支持主流的 x86 指令集，将主流语言和主流开发工具引入到智能合约开发领域，大大降低了智能合约和 DApp 开发的门槛。

Qtum 作为较早上线的主链，是典型的结合了区块链领域多种主流技术，并包含多项技术创新的公链项目，其良好的扩展性可以支撑后续技术迭代，同时也为其他公链项目的开发提供了良好的范本，其 AAL，DGP，x86 虚拟机等都可以适配到其他公链项目，对公链开发者来说是很好的参考。更多信息欢迎登陆 qtum.org 查看。❖

钟文斌，量子链技术负责人、中国首席开发工程师。主要负责 Qtum Core 和 Qtum x86 虚拟机等研发，对区块链基本原理有较为深入理解。在加入 Qtum 之前就职于 SYNOPSYS 任高级研发工程师，有超过四年的大型软件开发经验。本科毕业于上海交通大学，中国科学院硕士。

底层公链设计开发实操指南

Aurora 极光链案例

文 / 强科臻

区块链发展到现在，已经被大家喻为下一代互联网，其发展也经历 1.0 和 2.0，在区块链 1.0 阶段，以比特币为代表的分布式账本实现了不可篡改的点对点转账功能。在区块链 2.0 的时代，通过图灵完备的智能合约，实现了代码即法律，最具代表的就是以太坊。随着区块链从一项技术演变成一个行业，越来越多的企业开始使用区块链技术，但是很快就暴露出了区块链 2.0 问题，性能严重不足。区块链 3.0 就是为企业应用而生，来应对区块链在大规模应用中的结合和落地问题。

公链之争从未停止过，怎样设计一条公链成为大家关注的话题。

区块链的不可能三角问题

要设计一条公链，首先要了解区块链设计中“不可能三角”问题，即无法同时达到“高效低能 (Scalability)”、“去中心化 (Decentralization)”、以及“安全 (Security)”这三个要求。

- 最大程度的“去中心化”和“安全”则无法达到“高效低能”，比特币、以太坊采用 POW 共识机制便是目前最去中心化的共识机制。从数据结构上看，它们采用“区块 + 链”的结构，在可追溯、防篡改上具备安全优势，但是 PoW 需要很大的算力支持，所以注定无法达到高效。
- 追求“高效低能”和“安全”则无法完全实现“去中心化”，无论是比特币的 PoW 还是 AOA 的 DPoS，都是用自己的方式选择公共的服务节点，只不过方式不一样，DPoS 利用 N 位受委托人通过投票实现的股份授权证明，实际上都是对“去中心化”的退让，最大程度的降低去中心化就会形成联盟链和私有链，变成一个中心化的数据库账本。
- 优先“高效低能”和“去中心化”则“安全性”将会降低，区块链最大的特点就是在密码学的基础上实现了可追溯，防止篡改，在区块链的处理交易过程中，加密解密也是很耗时的。

公链开发实践

要设计一条公链，还要有一个近乎完美的经济体系。设计良好的经济体系，一方面可以保

护区块链安全稳定运行，一方面可以让更多的人以很低的成本使用区块链，在降低使用区块链成本的同时，我们还要尽可能的保证公平性，如果不能兼顾公平性，就会出现以太坊拥堵问题。Aurora 公链开发，主要又分了两个大的阶段。

第一阶段

快速的开发第一个版本，来填补区块链 2.0 到区块链 3.0 之间的一个过渡阶段，这个版本要解决好区块链下面的问题：1. 重新设计经济系统和交易上块策略，在新的经济体系中，我们更加关注交易处理的公平性，使用应用智能隔离的方式，通过可变规则对交易进行分析，动态完成。分类和类内排序，很好保证了不同类型交易互不影响，类内交易可以通过提高费用方式提高处理速度，有效的避免以太坊拥堵问题。并且我们最大程度的降低了交易过程中手续费的收取，因为我们认为，区块链体系中，收取手续费是对整个经济体系的保护，并不存在免费的系统，EOS 所谓的交易免费，只是换个方式而已，它们通过抵押 EOS 换取资源方式，交易过程中需要消耗资源，反而增加了用户使用区块链的门槛。

通过对不可能三角问题深刻理解，我们在第一阶段选择了以 DPoS+BFT 为基础重新设计实现共识机制，全新的抵押投票机制，既保证了每个人有参与投票权利，也保证了参与投票每个人真实有效。通过融入 BFT 产块批量验证机制，减少了区块确认数问题，也降低了自然分叉的几率，提高了交易性能，新的共识机制没有 PoW 产生的高能耗，又要保证相对公平性，和最大程度的去中心化原则。

优化了 P2P 网络，提高了广播速度和到达效率，为后期升级到 P2P 立体网络做准备，我们所希望的区块链，是一个秒级确认的系统，网络优化是个持续改进话题。

引入了多资产功能，多资产不需要写智能合约就可以快速发行 Token，没有合约代码，就没有 bug，提高了安全性。并且多资产还有通过设置股东投票的方式完成增发，超级管理员完成冻结解冻的附加功能，来应对不同场景对 Token 的要求。还增加子地址功能，来解决 Token 资产归集难，归集成本高的问题。

智能合约全面支持多资产操作，让非主链 Token 操作像主链 Token 操作一样简单、高效。

第二阶段

寻找高效、简单的共识机制 Aurora 公链第一阶段版本采用的 DPoS+BFT 的共识机制，就是为了保证共识速度足够快，提高确认速度，将确认区块降低到一个，但是这也不是最佳的共识机制，我们也在积极的探索新的共识机制，新的共识机制应该像 PoW 一样高度去中心化，但是非常简单的、不需要大量计算，同时可以到达秒级共识速度，一旦我们技术成熟我们会替换现在共识机制。

例如：我们正在研究的 VRF 可验证随机函数，随机抽取共识节点，具有不可预知可验证特点，它既保证了公平公正，黑客无法攻击，又能够比较好的让所有人都有权参与产块过

程，产快速度还可能做到秒级确认。

多链并行是我们第二阶段重点，现在比较流行的区块链设计：单链，双链，或者 DAG，他们在一定程度上都缓解了现在区块链存在的问题，尤其是性能问题，但是问题还是会慢慢会暴露出来。

- 随着使用用户增加，行业范围越来越大，它们又要重新面对性能问题，所以区块链处理能力应该是横向动态增加，纵向提高始终有上限。
- 区块链上很多不相关的数据，用户不希望保存，因为增长很快，成本随之增加。
- 不同行业 / 领域接入区块链，根据流量不同，一定会有拥有一条独立区块链的需求，但是又不能产生信息孤岛，链与链的价值无法传递尴尬情况。

Aurora 公链设计的多链结构，可以允许用户通过申请的方式提交发链的请求，Aurora 节点投票方式进行审批，如果通过，即可加入到多链的网络体系中，新增加的链是独立运行在网络中，并且随时和其他链产生价值交换。

P2P 立体网络，是为了更好的支持多链架构：

- 建立网络的立体分层分区域结构，通过对网络的优化，超级路由器，可以准确的识别出数据的到达区域，来提升通信、广播的速度。
- 更好的数据压缩信息处理方式。
- 智能化的布隆筛选，直接接收转发有效的数据。
- 智能合约

Aurora 公链第一阶段选择 EVM 作为我们优先支持的智能合约虚拟机，是因为，EVM 也是以太坊智能合约虚拟机，它的优点也是显而易见：

- 经过大量用户的测试和验证；
- 用 Solidity 编写的智能合约都会被编译成指令去执行，比较高效；
- 支持以太坊社区的开发者很多，更容易被大家接受。

但是智能合约最好要支持用 C++、JavaScript、Java，Go 等我们大多数程序员都能编的语言去写智能合约，这样才能够得到普及。或者说，假如开发 DApp，也能够让普通开发人员来去写这些智能合约，才能让 DApp 更快的普及起来，所以我们也开发集成 WebAssembly 来支持新的智能合约，这样很多高级开发语言都可以直接编译成 WebAssembly 的字节码，这种字节码和底层机器码很相似可快速装载运行，性能也是非常高。

如何逐步的做到极度安全

现在绝大多数区块链项目采用的均为椭圆算法，但在量子计算面前，基本上几秒钟甚至是一瞬间就会被破解，这将导致我们所有账户余额、私钥都会被破解。在量子计算面前也是毫无隐私，破解后可以造假、转账，做任何事情。所以我们的加密计算方式必须要能抗量子计算，保障无隐私泄露的危险。

- 加密方式要和用户数据分离，这样加密方式升级对用户数据完全没有影响。
- 零知识证明是能够解决，当转账的时候，只能通过验证交易是否正确，并且整个过程不泄露隐私。
- 格计算是一种基于多维空间的数学转换过程，这种多维的空间随着维度每增加一维，它的计算的难度，不仅是指数级的增加，更是一种复杂级别的增加。这种增加使得未来的技术，即使量子计算，也无法破解。

对同行的建议

区块链出现，最大的贡献就是用低成本方式解决信任问题，所以我们在做好公链的技术支持的同时，要不断的探索不同行业 / 领域结合方式，不能夸大公链能力，造成区块链滥用，做好 layer1 和 layer2 的结合，不同企业需要的不仅仅是区块链技术，更需要的是一个完成区块链生态支持，需要生态中的交易闭环，帮他们解决传统互联网方式解决不好的问题，做好区块链生态，更好的服务于各个行业。❖

强科臻，Aurora 极光链 CTO，吉林大学计算机与软件工程专业，高级软件架构师和技术爱好者，拥有丰富的互联网和区块链开发经验。

DApp 开发者公链选择指南

文 / 刘虔铭（青冥子）

DApp 是指一种新型的基于区块链的分布式应用，其核心代码不运行在某个服务器上，而是运行于区块链共识节点的虚拟机上。DApp 的运行基于所属公链的共识机制、合约规范，DApp 的代码和数据是链数据的一部分。不用担心 DApp 因为某个服务器奔溃而无法使用，也不用担心 DApp 官方会偷偷篡改逻辑和数据。实际上，DApp 并不仅仅限于智能合约，而是包括智能合约、后台链上链下通讯处理、前台呈现的一系列代码的集合，是一个能提供完整功能的整体。

那么，对于希望投身 DApp 开发的开发者来说，最关键的一个问题是什么呢？那就是如何选择选择一个合适的公链。本文将从多个方面介绍公链的选择思路，供广大对 DApp 开发跃跃欲试的开发者参考。

比较公链的效率

首先，我们来看公链的共识机制，也就是公链的宪法。它决定了节点间以什么样的方式实现相互信任，什么样的交易操作合法，什么样的交易能够被（优先）打包出块。目前共识机制大体分为几类：PoW 工作量证明（BTC、ETH 现在）、POS 股权证明（ETH 未来）、拜占庭容错类（NEO、EOS）。他们的资源消耗（包括但不限于电力）、参与共识的节点数、相对理论安全性为递减关系，而效率基本为递增关系。

作为 DApp 来说，一般建议选择效率更高的公链，以获得相对较快的确认速度，进而获得更好的用户体验。DApp 一般包含多个方法，需要在一个相对可接受的时间间隔（比如 NEO 的 15-20 秒）内相互承接，而并不像 BTC 一样只做一个最简单的转账处理（所以 10 分钟问题也不大）。当然无论如何，DApp 速度达到和传统 App 一样的效率是不现实的，因为 DApp 的主要特性是安全和分布式，而不是效率，所以把区块链简单的当作分布式数据库的开发者将遇到非常大的挑战。

此外，一些公链特有机制也会影响 DApp 的使用效率。比如 EOS 的 CPU、RAM 市场化租用和购买机制，会导致 DApp 可用资源不确定性，也是公链选择不可忽略的因素。

关注公链的安全性

安全性方面，我们首先看分叉问题。分叉是区块链世界一个非常有趣的事件，它会导致公

链社区分裂为两条或多条发展线。既有人为分叉，也有技术性分叉。人为分叉，比如著名的 ETH 在 DAO 被攻击后分裂出了 ETH 和 ETC 两个独立的公链。根据对于不可篡改的信仰，在 ETC 上所有数据是没有任何回滚的。但在 ETH 上，根据作恶应该被制止的原则，被 DAO 攻击后的数据进行了回滚。除了人为分叉以外，某些类型的共识机制的特性导致的同步问题也会导致分叉。可能 A 和 B 在相同时间出了同样高度的块，而作为一条区块链是不允许出现这种情况的。此时，就要看谁的块被更多的节点接受，而不被接受的块则成为孤块从而被认为是无效的（这个不被承认的块就成了主线上一个支叉），这种分叉对于 DApp 设计的影响可能比认为分叉更大一些。在选择的时候，你首先要确定你的 DApp 对于分叉是否敏感，是否会因为分叉而产生运行障碍和安全风险。如果非常敏感，则应该选择例如 NEO 这种从共识机制上避免分叉的公链，而如果并不敏感的话选择面就比较宽。

此外，我们也要看一个公链的安全审计做的如何，其 DApp 安全审计环境又如何。公链代码本身是 DApp 安全的第一道墙，如果公链本身存在漏洞，那么 DApp 再如何设计安全，也是无济于事的。我们建议尽量选择有可靠安全审计的公链，比如是否经过国内外有影响的安全公司（如 red4sec 等）的安全审计。

如果公链本身安全，我们又要再回到 DApp 本身视角。DApp 的一大特性就是不可篡改性，也就是即使开发者也不能随意改变智能合约既有的逻辑。即使要改变，一般只能新发一个合约，并通过某种方式迁移旧合约数据。很明显，这个过程是非常繁琐和中长期的，同时处理过程一般也是不得不公开的。这个特性决定了，DApp 一旦被发现存在漏洞，攻击者将有更多时间利用它（比如掏空合约的资金）。所以 DApp 代码应该被更加严格的反复测试和第三方安全测评。在选择公链的时候，建议首先调查下是否有安全机构提供相关安全审计服务，以及是否丰富与易获得。比如 NEO 智能合约目前有 NEL、red4sec、奇虎 360 提供合约安全审计服务。

注意公链稳定性

在安全性之后，我们要关注的是公链的稳定性，看看是否能够稳定出块，是否会被恶意攻击所影响。一个公链的平均出块时间和平均出块稳定性将影响 DApp 中的一个操作多久可以完成，以及操作的持续稳定性。比如 BTC10 分钟比较难以忍受，但 ETH、NEO15 秒就好得多。此外，还需要关注公链的网络拥堵应对能力，是否内置有效的防瘫痪攻击机制。如果没有有效的防拥堵机制，将会带来非常严重的安全问题。比如曾有一个知名的游戏，其中一个机制是最后一个参与的人将获得奖励。黑客就利用疯狂发送交易的方式，在他参与后几乎瘫痪了公链，使得他必然为最后一个玩家，而最终获得了奖励。

这里针对公链防拥堵机制举个例子：NEO 虽然目前基本交易操作都是免费的（10GAS 免费额），但是对于免费交易设置有防拥堵与防滥用机制：当单个区块的交易数达到 500 时，将限制只打包最早的 20 个免费交易；当交易池未发送交易达到 5 万时，将限制免费交易

提交。这些措施旨在避免低成本免费(手续费 <0.001GAS)交易成为拥堵网络的垃圾交易, 保证收费交易优先发送, 同时在网络不拥堵的时候又能使得交易免费执行。

体验公链易用性

如果一个公链即高效又安全还稳定, 那么我们还要看看这个公链的易用性。也就是实现类似的功能逻辑, 你需要花费多少学习、实现成本。有的公链有自己的特有语言, 如 ETH 的 Solidity。有的公链则直接支持多种常用语言, 如 NEO 支持 C#、Java、Python、Go 等语言。一般, 每个公链都会提供智能合约开发工具, 但是使用难易也各有不同。比如 NEO 提供 VS 插件、Nuget 包, 更加可以选择在线行级云端合约调试排错工具。DApp 开发还需要公链生态提供丰富的 API 和 SDK, 我们要考察 API 的规范性、易用性和稳定性。传统程序必须通过 API 和 SDK, 才能和公链上的智能合约进行交互通讯, 公链节点的通讯接口是公链与链下世界交互的大门。

接下来, 是要考虑公链的基础设置是否齐备和丰富。主要是区块浏览器和钱包两大类。区块浏览器一般是直接读取公链节点数据或通过爬取缓存数据将链上数据呈现给用户。它是区块数据对普通用户的一个重要窗口, 它可以把区块上面复杂难懂的二进制码变为普通用户易懂的一个个图文信息。钱包一般分为普通钱包和轻钱包。轻钱包多为 web 或移动形式, 是不用同步区块数据就能执行操作的钱包。它是区块链和用户的桥梁, 用户通过钱包和区块链建立交互操作, 用户无论是简单的转账还是复杂的合约调用操作都离不开一个好的钱包。一个 DApp 一般是一个钱包内部的一个插件, 钱包提供地址余额查询、交易签名发送能支持, 是 DApp 的重要伙伴。例如 NEO 就有官方开发的 GUI 钱包(普通钱包)、NEL 开发的 Web 钱包(内嵌 NNS 系列 DApp)、COZ 开发的 Neon 桌面轻钱包、O3 开发的移动端双平台钱包(支持 DApp 调用 JS 包实现钱包基础方法)。

此外技术支持也是必备之物。第一个要看是开发文档的数量、质量、语言, 它是为开发者打开 DApp 开发的大门第一把钥匙。一份好的文档可以帮助新入门开发者少走弯路, 帮助老开发者拓展思路。另外, 实在来说, 对于中国的开发者, 一份中文的开发文档总是好过其他语言的。文档之外, 我们还需要动态的技术支持。如果公链有活的技术论坛, 我们就可以快速解决文档中缺失或未及时更新的问题, 并及时解决我们急需的问题。有些大型的公链项目及其社区, 甚至会提供线下、线上技术教学课程, 那对开发入门和疑难解答来说更加是如虎添翼了。目前这方面来说, 做的比较有代表性可以看 NEO, 它提供准确完善的中文(当然也有其他语言) 文档、有活跃的技术论坛(<https://bbs.neldev.net/>)、还有和 NEL 合作定期举行的技术公开课。

最后, 我们还要看看公链是否形成了获得广泛共识的通用规范。由于区块链的去中心化文化, 区块链中很少有自上而下的规定, 更多的是众多社区参与者的共识。所以, 能够形成广泛共识的通用规范, 意味着这个公链已经相对成熟。同时, 通用规范是不同 DApp 互相兼容、互相交互的基础。比如, Neo 的 NEP5 通证规范, 其同意定义了包括余额获取、转

账、通知格式等一系列规范，保证我们可以非常方便地用统一的方式处理任意 NEP5 通证的各种操作，这点非常重要。一个公链的基础通用规范，一般包括私钥的保护机制、钱包格式规范、通用通证规范、合约间调用规范等。我们建议选择，通用规范比较多并已经使用广泛的公链。

综合评估公链生态环境

在以上内向化考虑点之外，我也要看看公链的生态发展如何。也就是看公链对 DApp 开发的支持力度大不大、是否有官方或者专项技术基金、是否提供经常性的有效性的技术支持、是否会定期举办开发者比赛等活动。

目前大多数 DApp 开发团队或公司多为初创，也因为区块链行业的新兴状态承担着很大的试错风险，非常需要资金的支持。如果公链官方或其生态社区建立有针对 DApp 优秀项目的基金，甚至那么无论你是否能否享受到都是有巨大助力的。有些大型的公链项目，甚至还会定期组织大型赛事活动，提供该公链的 DApp 知名度并发掘新兴优秀创意。比如。NEO 就会每年定期举办应用开发者大会和游戏开发者大会，展示优秀项目，并对特别优秀的项目进行投资。

针对行业选择方面，各个公链都有自己一定的特色，有些是金融特色，有些是供应链特色，有些是游戏特色，有些是普适的，应该要根据自己的应用的特色，选择合适的公链。目前，公链发展呈现两个发展趋势，一个是追求高效化，一个是发展专业化特色化。比如你是金融应用，就应该要以安全为第一要务去选择公链，同时也要关注是否支持高精度数字计算、是否支持数字化身份。而如果是游戏，那么就应该选择 TPS 高的、支持幂运算的公链。

公链的 DApp 合约的部署和使用成本也是生态的重要一环。合约部署成本影响你的 DApp 开发和运营成本，而使用成本则是你 DApp 的门槛有多高。比如 ETH 每次发送交易（对 DApp 来说就是执行方法）都需要一定的 ETH 作为燃料，而 NEO 则对一般的交易执行免费政策。这样，在相同 DApp 功能的情况下，用户会更偏向于门槛低的公链平台。

此外公链上 DApp 质量与种类丰富度，就是公链生态最直观的体现了。质量非常重要，它是公链成熟度的一个标志。如果一个刚起步的公链，其 DApp 一般都是非常简单的存入和取出数据类应用。而一个成熟的公链，你一定能看到逻辑非常复杂，代码成百上千的合约。再看合约种类丰富度，则表现了公链的技术延展性。有的公链，设计比较局限，可能只能支持一些特定的简单的应用方向。而一个设计更加包容的公链，则有足够的纵深，支持任意类型的应用。例如，目前 NEO 就已经有了去中心化交易所 NEX、别名服务 NNS、稳定币阿基米德、同质化代币游戏疯狂角斗士等多种类、高质量的应用群。这些应用，对用户是优秀体验，对开发者则是绝好的参考教材。

最后，我们要考察的是公链开发者社区的质量与分布。众人拾柴火焰高，有一个社区一起

发展 DApp 技术，一定比一个人单干效率高得多，走更少的弯路。要看社区分布广度，是某些国家还是全球。要看是否有本地社区，能够实现更加高效的沟通。要看社区是否有高质量的项目，要看社区的活跃度。要看社区的成员质量与数量。比如 NEO 既有发源于美国的 COZ 社区、也有服务国内着眼全球的 NEL 社区，同时在巴西、俄罗斯等国家和国家也均有技术社区组织。这些组织围绕公链生态发展相互协作、又互有竞争，使得公链生态呈现了百花齐放、百家争鸣的良好态势。

结语

目前国内外各公链他方唱罢我登场、八仙过海各显神通，希望本文能够给予小伙伴们以一点点迷雾中的光亮，最终大家都能凭借自己的智慧选择到最合适的公链平台。❖

刘虔铭（青冥子），新经济实验室 NEL（NEO 中文技术社区）项目总监，主要负责 NEL 官方重点项目如轻钱包 API、thinSDK、NNS 等项目的方案设计、DApp 系统合约、后端、前端三层实现规划、核心功能开发、新技术研发等工作。加入 NEL 之前为上海三甲医院信息工程师，有超过 9 年的信息化系统建设经验。本科毕业于上海应用技术大学，中国海洋大学项目管理硕士，前医疗信息化工程师。

从节点钱包，谈打磨易用的区块链产品

文 / 罗鹏

众所周知在传统互联网领域，已经拥有种类繁多的优秀产品，给人们的生活带来了前所未有的改变。而对于区块链行业而言，还处于一个发展早期，基于区块链的产品并不算丰富，对于大多数公链来讲，钱包则应该是他们的标配产品。纵然是标配，做出一款好的区块链钱包却并没那么简单。

在区块链的世界，钱包是一个重要的入口，它的核心用途是存放数字资产和进行转账交易。通常是通过私钥 + 密码的方式保护资产的安全，如果私钥丢失，便会造成数字资产的丢失，因此在区块链世界中，用户一定要保存好自己的私钥。

区块链钱包根据安全性和易用性通常可分为：

节点钱包

- 该类钱包需要下载整个区块链数据，是一个完整的区块链节点
- 由于节点钱包可作为链上的一个节点，有所有交易的详细信息，所以其稳定性和安全性都较高
- 节点钱包会参与网络维护

SPV 钱包

- 无需下载整个区块数据，只下载区块头数据即可
- 使用 SPV 钱包需要理解两个词：“交易验证”和“支付验证”。交易验证涉及验证余额是否充足，交易是否双花等。支付验证只判断该笔交易是否被验证过，得到了多少个确认
- 虽然 SPV 钱包避免了下载大量数据的问题，但它只能做支付验证

轻钱包

- 无需下载任何区块数据，发起交易时只需对交易签名然后广播即可
- 轻钱包的数据实时地从节点获取，因此，节点服务器的性能决定了轻钱包的性能，但该类钱包的便捷性较高
- 硬件钱包

- 硬件钱包是把私钥放在 USB 设备里，安全性极高
- 该类钱包需要软件和物联网硬件结合，使用比较麻烦

目前使用较多的是节点钱包，不过很多节点钱包都存在一些相同的问题，例如

- 使用麻烦，更换设备就需重新下载节点程序，并且同步数据时间比较长
- 如果要使用多种数字货币，就需要下载多个节点钱包
- 节点钱包 UI 更新困难，如果要更新 UI，就要重新下载或更新节点程序

本文将主要针对节点钱包进行介绍和剖析。由于篇幅有限，其余类型的钱包这里就不展开讲了，有兴趣的同学可以自行查资料了解。

节点钱包的基本功能和原理

节点钱包需要下载整个区块链的数据，节点程序运行起来后会不断去同步区块数据，保证本地节点的区块数据和链上其它节点数据的一致性。当用户发起一笔交易，节点钱包会用本地的私钥对交易进行签名然后广播到网络中，整个签名过程都发生在本地，因此安全性会得到极大的保障，此外当本地节点收到其他节点广播的消息后也会将该消息广播给它所连接的节点。节点会不断去连接最长链，若本地节点当前处于分叉链，则会自动回滚，然后去同步最长链的区块数据

节点钱包实际就是一个客户端程序，需要占用硬盘空间和内存资源。对于钱包用户来讲，节点钱包可以提供的基本功能通常如下：

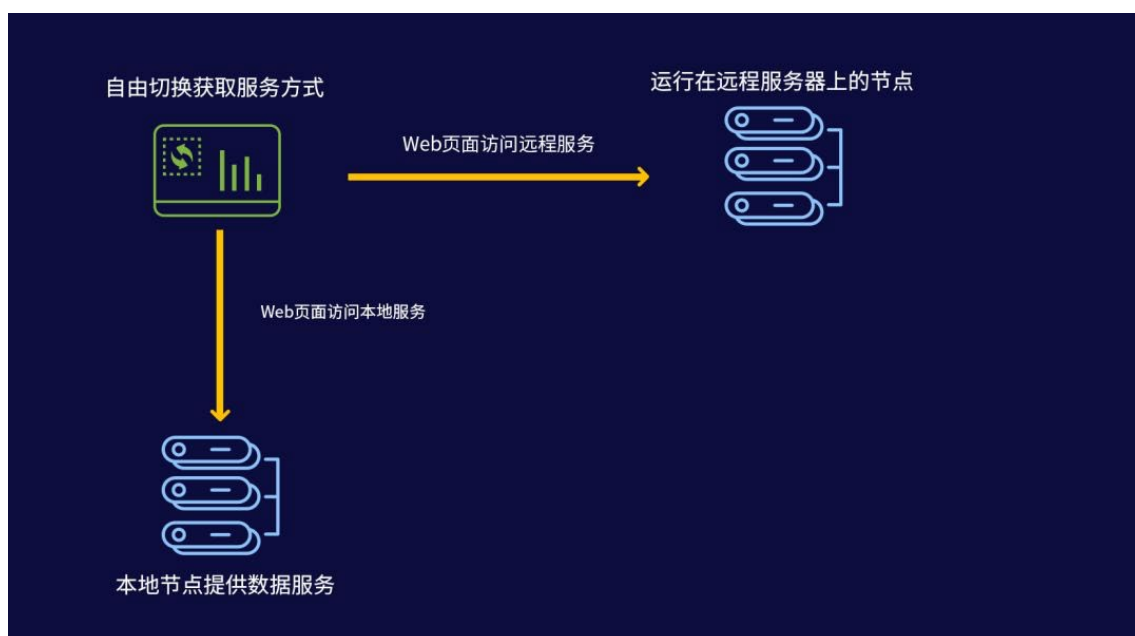
- 生成和备份私钥，保存所有账户
- 实时展示资产情况
- 进行转账收款等操作
- 参与链的网络维护

其中参与链的网络维护是节点钱包和其他类型钱包最大的区别，节点程序是其他类型钱包能够稳定运行的基础。不少用户选择节点钱包的原因也是想通过参与网络维护来获得奖励。节点钱包的基础功能并不多，但对每一项功能的稳定性要求很高，满足这些基本功能后，那该节点钱包可以被定义为可用。在这个阶段，通过交互设计带来的体验提升是有限的。交互设计提升的体验是视觉和操作上的，但一切的交互体验都是建立在功能完备的基础上。因此我们想要节点钱包在体验上有质的改变，还应该先从功能上下手。

可作为本地服务器的节点钱包

通过节点钱包的工作机制我们可以看出，它相对较重，当用户更换设备或设备不在身边时，需要重新下载节点程序，然后等待同步数据完成。这样对于用户使用产品的即时性就大大降低，那么我们是否有办法可以解决这样的问题，让用户不受地点和时间的限制呢？

其中一种处理方式就是：中心化结合非中心化。其中中心化指的是我们可以提供一个 web 版本的钱包给用户，前端页面通过 https 请求连接链上的某个节点来获取整个链的数据来展示给用户。对于交易的签名，在本地浏览器中用 Javascript 完成，这样也保证了私钥不会被发送到网络中，保证了资产的安全。非中心化指的是当用户可以使用常用设备时，可以打开 web 钱包然后把连接的服务器切换到本地的节点程序，这样可以极大加快响应速度，提升用户的使用体验。这种方式就和之前直接运行节点程序一样，只是将 UI 单独剥离出来了。用户还可以切换到自己认为更加稳定或带宽更高的远程节点服务器上，这些都是可配置的。



通过中心化结合非中心化的方式，在之前节点钱包基础上增加了以下功能提供给用户：

- 直接通过网页即可使用区块链钱包
- 根据实际情况切换所连接的节点程序

通过这两个功能便解决了节点钱包 UI 更新困难的问题，UI 资源存放在中心化服务器上，可随时更新。同时也解决了钱包更换设备需重新下载钱包的问题。用户可在任何地方连接自己运行的节点或者可信节点。当然，最好的方式肯定是连接本地节点，避免了可能由于网络带来的不稳定问题。通过中心化结合非中心化的方式就让节点钱包的体验开始走向易用方向了。

基于模块化设计的节点运行多条链

随着区块链领域的产品落地，区块链产品会逐渐走入我们的日常生活，然而我们需要的产品服务，是一条链没办法全部解决的。那我们肯定会涉及到使用多条链，这就意味着我们需要在自己的设备上去下载并运行多个钱包，这显然是不合理的。我们需要寻求某种方式，让我们可以更方便地使用区块链服务。

NULS — 一个可定制的区块链基础设施，给我们提供了一个解决方案，通过模块化设计让节点钱包同时支持多条链。关于 NULS，有兴趣的同学可以直接去官网 nuls.io 进行了解，本文我将直接讲解决方案。

首先，大家需要理解一种程序设计思想 - 模块化设计，它指的是对众多功能按照某种维度进行划分，设计出一系列功能模块，通过对模块的选择和组合可以构成不同的产品，这个过程有点类似于使用不同形状乐高积木来搭建一些建筑。

其次，大家还需了解一个产品 — 链工厂（链工厂是 NULS 的一个产品），链工厂就是基于模块化架构开发的一个产品，这个产品提供的主要功能就是快速低成本造链。在链工厂中，用户可以根据实际的业务需求按照以下步骤快速生成并运行一条新链。

- 在模块仓库中挑选合适的模块，例如共识模块、网络模块、账本模块、业务模块等等
- 配置初始参数，例如发行量，初始持有 Token 的地址
- 打包生成链的程序包
- 程序生成后可选择直接通过链工厂的云节点来运行新链，这样可以节省用户自己搭建环境的时间



那么链工厂的节点是如何办到同时作为链工厂和其他链的节点钱包呢？

我们知道链工厂是模块化架构，因此其节点钱包是通过不同的模块组合而成的，而通过链工厂造出的链也是模块化架构。链工厂的节点只要包含新链节点的模块，那么新链的节点钱包就相当于链工厂节点钱包的子集，那链工厂的节点自然就可以作为新链的节点钱包运行了，且用户还可以在链工厂节点钱包中自由安装或卸载某些链，也就是说用户可以自由选择节点钱包支持哪些链。

通过模块化设计 + 链工厂的解决方案，节点钱包就又在之前的中心化结合非中心化的方式上增加了以下功能提供给用户：

- 支持同时作为多条链的节点钱包
- 自由安装或卸载某些链

链工厂的节点钱包可同时作为多条链的节点钱包，解决了用户需要下载不同链的节点钱包问题，更加符合用户的实际使用场景，能给用户带来更好的使用体验。

总结

随着业务的发展会持续不断产生新的需求，我们需要通过调整或者改进我们的产品去快速应对这些变化，正如在本文所分享的一样，在区块链行业初期，用户只需要一个可管理资产，转账的钱包，于是我们提供了包含这些功能的钱包给用户，但随着发展，用户对钱包的使用频率增加，于是对钱包的稳定性和易用性要求开始提高。我们就应对变化，用中心化和非中心化结合的方式解决了即时性的问题。在可预见的将来，区块链产品会逐渐走入我们的生活，我们的节点钱包需要支持多条链，我们便提出单个节点钱包支持多链的解决方案。我们需要通过一种循序渐进的方式来逐渐打磨一款高易用性的区块链钱包，让用户能够体验到一款优秀的区块链钱包应该是什么样子。❖

罗鹏，NULS 产品经理。2015 年开始接触并了解区块链，对区块链的产品和技术一直在不断关注和研究。2018 年 8 月从华为离职加入 NULS 团队，任职产品经理，负责 NULS 相关的钱包，浏览器，链工厂等一系列产品。本科就读于重庆大学计算机学院，对产品和技術充满热情。