共识机制设计指导

星云研究院

2019 年 3 月
版本号:0.0.1

目录

1 简介 1

2 相关研究 2

2.1 区块提议 2

2.2 一致达成 3

3 系统模型 4

3.1 网络模型 4

3.2 时钟模型 5

3.3 数据模型 5

4 区块提议 6

4.1 基础 6

4.2 性质分析 8

4.2.1 PoW 协议 8

4.2.2 PoS 协议 8

5 一致达成 9

5.1 基础 9

5.2 性质分析 9

5.2.1 链式协议 10

5.2.2 投票协议 10

6 总结 11

附录 A 对 PoW 的进攻方式 13

附录 B 谜题选择 15

附录 C 随机数的生成 18

附录 D 投票系统 19

附录 E 匿名性 20

1 简介

共识 (Consensus) 是布式系统中的基本性质之一，具体来说是指通过消息传递使得系统中所有节点均以相同顺序执行一个命令序列 [1]。而在实际环境中，节点可能出错产生异常行为，或者消息无法被正确传递，使得系统无法达成一致 [2]。早期研究指出，当系统中仅有两个节点时，不存在一种容错协议使得系统达成一致 [3]。1982 年 Lamport 等人将其扩展为多节点下的容错问题，即拜占庭将军问题（Byzantine Generals Problem，BGP）[4]。

拜占庭将军问题被认为是容错性问题中最难的问题类型之一，在区块链技术出现之前，拜占庭容错并没有得到广泛关注。自上世纪 90 年代起，由 Lamport 提出的 Paxos 算法 [5] 被公认为分布式系统共识的经典解决方案 [6]，并得到了广泛的应用 [7, 8, 9]。但 Paxos 等算法仅能实现宕机容错（Crash-Fault Tolerant）而不满足拜占庭容错（Byzantine-Fault Tolerant）。

2008 年匿名发布的比特币白皮书 [10] 提出了一种去中心化的账簿，其从全新的角度阐述了拜占庭环境下的共识解决思路。以比特币为代表的区块链也属于分布式系统，由于存储在这些帐簿中的价值，恶意节点有巨大的经济动机去尝试造成故障，因此这类分布式系统对于拜占庭容错的需求非常高[1]。

区块链技术的出现让拜占庭容错问题重新得到关注，而尝试解决拜占庭容错问题的共识算法也不断被提出。早期的区块链共识机制与传统的拜占庭容错算法存在较大差异，例如比特币中通过算力竞争的方式决定出块权，并且基于最长链原则实现了大规模节点的状态同步 [10]。近几年出现了许多结合经典分布式系统 BFT 机制的共识机制，例如 Tendermint[11]、Byzcoin[12] 等等。

实际上，区块链系统的共识机制涉及到众多领域，包括密码学、分布式系统甚至是经济学等。各种区块链共识算法，其实现机制千差万别，究其原因是其设计者的研究背景各异以至于对区块链共识有着不同的理解[2]。因此在设计出合理的共识机制之前，我们需要回答如下问题：

- 区块链共识机制需要解决什么样的问题?
- 理想的区块链共识机制需要满足什么样的要求?

[1] Understanding Blockchain Fundamentals, https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419

[2] https://zhuanlan.zhihu.com/p/52251671

在此，我们可以先回答第一个问题。

作为广义上的分布式系统，区块链的共识也是为了解决拜占庭容错问题。对应到区块链的流程上通俗地来说，共识协议是在解决“谁负责出块”和“出现分叉如何解决”这两个问题 [13]，论文 [14] 将这两步定义为区块提议（Block Proposal）和一致达成（Agreement）。尽管许多共识机制并未对两者进行严格划分，但这种解耦思想有助于我们理解区块链共识并且在此基础上设计合理的共识算法 [12, 15]。

本文的后续章节则尝试回答第二个问题。

第二章简单介绍现有的区块链共识工作；第三章给出了区块链共识涉及到的系统模型；在此基础之上，第四章和第五章分别给出共识协议中区块提出和一致达成的性质分析。

需要注意，本文仅针对区块链共识机制提供设计指导，而并非完整的共识机制设计方案。

2 相关研究

本章节简单介绍已有的区块链共识研究工作，这里并未按照一些公认的划分方法 [16, 17]，而是分为区块提议和一致达成两部分介绍。同时受限于篇幅，本章并不会详细介绍各种研究，更详细的介绍可以参见星云共识调研报告 [18]。

2.1 区块提议

在传统分布式服务中，一般发起状态更变命令的节点称之为领袖节点（Leader）或者主节点（Primary），此类节点一般由发起请求客户端指定或者按照某种固定算法决定 [19]。在区块链系统中，由出块节点（Block Proposer）负责提出区块，从而发起整个账簿的状态更变请求，但区块链中出块节点的选择和传统的领袖选举仍然存在较大的区别。

以比特币 [10] 和以太坊 [20] 为代表的区块链通常对于出块节点没有准入限制，参与出块的节点可以随时加入离开整个网络，此类区块链系统被定义为无授权区块链（Permissionless Chain）[21]。这类系统通过竞争方式选择出块节点，其中最为广泛采用的是工作量证明（Proof-of-Work，PoW）算法 [22]。需要指出的是，PoW 仅为出块节点选择协议，而并不是完整的共识协议[3]，这一点在 [10] 中已经指出。

对于联盟链（Consortium Blockchain）以及部分采用 DPoS 机制的公链 [23]，参与

[3]通常比特币采用的共识被称之为 Nakamoto Consensus。

节点需要获得授权并且不能随意加入离开，因此被定义为授权区块链（Permissioned Chain）[21]。在此类系统中，一段时间内所有参与节点集合已知且不变，出块节点的选举可以采用轮询机制（Round-Robin），从而整个共识问题退化成传统分布式系统中的共识问题，即在出块节点的选择已经达成共识的情况下（此时出块节点仍可以采取恶意行动），所有验证节点如何针对出块节点的提案达成一致 [19]。

由于 PoW 会消耗大量电力，权益证明（Proof-of-Stake，PoS）随后被提出，不同于 PoW，后者实现出块选择往往与节点的链上资产相关而不依赖于物理算力因此都称之为 PoS，但不同基于 PoS 的共识之间通常差别较大。

早期的 PoS 仍然是基于 PoW 的变种实现 [24]，即对谜题计算中的目标基于权益（Stake）或币龄（Coin-age）进行加权。作为备受瞩目的以太坊演进方案，Casper[25] 被认为是一种 PoS 协议，但实际上 Casper 是利用 stake 实现链式结构的最终性 (Finality)，并不涉及出块提议过程，因此基于 Casper 的共识仍然可以选择采用 PoW 机制。

近几年涌现出一些基于随机数生成（Random Number Generation，RNG）的共识机制实现了基于 PoS 的出块提议方案 [14, 26, 27]。尽管在出块方式和共识实现上存在区别，上述方案均利用伪随机函数（pseudo-random function）实现了出块节点的随机选则。

2.2 一致达成

出块节点打包区块并广播后，其他节点对区块进行决策从而达成一致。

以比特币为代表的无授权区块链采取了最长链原则（Longest Chain Rule）[10]，随后论文 [28] 提出了最重子树原则（GHOST Rule）。最长链原则和最重子树原则通常被称之为链式协议（Chain-based Protocols），在链式协议中，节点基于统一的规则构建链式账簿。

对于链式结构的系统，在达到最终一致性前任何区块状态都可能发生改变（例如某个区块在某个时间后不再属于最长链），通常这种改变概率随着链式结构增长单调递减 [10]，因此链式结构共识无法保证最终性（Finality），或者仅提供概率最终性 (Probabilistic Finality）[16]。

以联盟链等项目为代表的非授权区块链通过缩小节点规模，实现了更强的一致性保证。节点针对每个提案进行投票，通常基于 PBFT[19] 或者某些更为宽松的投机 BFT 算法 [29]，此类机制被称之为投票协议（Vote-based Protocols）或者 BFT 协议 (BFT-based Protocols）[4]。不同于链式区块链，采用投票协议的区块链会针对每个出

[4]https://blog.cosmos.network/consensus-compare-casper-vs-tendermint-6df154ad56ae

块进行一致性确认，因此每个区块状态不可能被扭转，从而保证了最终性。

由于 PBFT 算法以及最差情况下投机 BFT 算法的网络通信开销为 $O(n^2)$，因此采用此类协议的网络往往会控制验证节点规模 [19, 29]。近几年一些研究尝试在投票协议系统中扩展验证节点规模，例如 ByzCoin[12] 和 OmniLedger[30] 使用了集体签名 (Collective Signing) 实现了大规模节点投票；Algorand[14] 采用密码抽签（Cryptographic Sortition）方式可以从候选集合中抽取验证节点再投票。

较为特殊地，Casper[25] 的交易验证采用了一种链式协议和投票协议的混合策略。具体来说，Casper 将某个特定区块高度的区块定义为检查点（checkpoints），在检查点之间的区块采用最长链原则，而对于每个检查点则通过投票来保证最终性。

3 系统模型

本章节主要给出区块链共识涉及到的相关定义，在此基础之上我们给出相关的评价指标。

3.1 网络模型

分布式系统中，节点之间的通信方式主要为消息传递（Message Passing）。根据网络传输延迟的设定，可以将系统划分为同步网络（Synchronous Network）和异步网络（Asynchronous Network）。在同步网络模型下，所有节点的消息传输延迟不会超过某个阈值；而在异步网络中，节点间的消息传输延迟可以是任意值。

根据 FLP 定理 [31]，异步网络环境下，只要存在一个拜占庭节点，便不存在一种共识算法可以同时满足安全性（Safety）和活性（Liveness）[5]。因此现有的共识机制通常采用一些额外的假设，例如 PBFT 允许网络中除领袖以外节点之间的通信可以为异步网络，但主节点和副本节点之间的消息必须可达 [19]；而 Algorand 则假设整个网络可以一段时间处于异步状态，但最终会回到同步状态 [14]。这些设定通常称之为“弱终止假设”或“半异步网络模型”，但本质上仍然属于同步网络。综上，在本文中我们基于同步网络模型讨论共识算法。

在一些传统分布式系统中负责发起状态更变请求的节点被定义为主节点 (Primary)，接受并验证请求的为副本节点（Backups）[5, 19]。在一些区块链系统中，所有参与打包区块以及验证区块的节点，统称为矿工（Miners），而在另一些系统中则区分为出块节点、验证节点和监听节点 [13]。

[5]Blockchain in the Point of View of FLP Impossibility, https://medium.com/codechain/safety-and-liveness-blockchain-in-the-point-of-view-of-flp-impossibility-182e33927ce6

论文 [32] 提出了更具有一般性的分布式共识系统节点模型，即系统中分为提案节点（Proposers）和接受节点（Accpetors）。本文采用类似设定，对于有 n 个节点的系统 U，将验证节点集合表示为 $V = \{a_1, a_2, \cdots\}$，其中 $V \subseteq U$ 且 $|V| = n_v$；对应地将出块节点结合表示为 $P = \{p_1, p_2, \cdots\}$，其中 $P \subseteq U$ 且 $|P| = n_p$。

3.2 时钟模型

由于网络传输延时，分布式系统中的节点无法实现完美同步时钟（Perfectly Synchronized Clock）[6]。一种更严谨的描述方法是采用逻辑时钟（Logical Clocks）[33]，即分布式系统中事件发生顺序关系记录。

一种观念认为，在区块链系统中，每有一个新高度的区块提出意味着整个系统的时钟前进一步[7]。区块高度（Block Height）表示一个区块在区块链中的位置[8]。因此，区块高度本质上是一种逻辑时钟，用于记录不同区块间的顺序关系。

我们可以给出更为准确的一致性描述：所有节点对于提案区块的发生顺序达成一致。因而区块链共识实现的一致性通常是指分布式系统中的顺序一致性（Sequntial Consistency）而并非线性一致性（Linearizable Consistency）。

许多共识机制中也引入了物理时钟概念，比特币中区块和交易都包含时间戳（Timestamp）[10]，以及 [14] 中需要基于本地时钟判断投票超时。

后文我们用 $\hbar$ 表示区块高度，用 t 表示物理时钟。

3.3 数据模型

以比特币为代表的交易网络主要采用了未花费交易输出模型（Unspent Transaction Outputs Model，UTXO Model）；而以 Ethereum 为代表的区块链系统采用了账户模型（Account Model）。通常来说，账户模型设计更接近于传统记账系统（例如银行），并且支持图灵完备智能合约（Turing-complete Smart Contract），因此这里我们基于账户模型给出相关定义，当然，UTXO 模型理论上同样适用于我们的共识设计。

在区块链中，最基本的数据结构是交易（Transaction）和账户（Account），这里给出相关定义。

定义 1. （账户）一个账户 i 拥有一个私钥（Secret Key）sk_i 和基于私钥构建的公钥

[6]更本质的原因是，即便不存在网络延迟，由于相对论的存在也不可能实现完美同步时钟。

[7]Blockchain Proof-of-Work Is a Decentralized Clock, https://grisha.org/blog/2018/01/23/explaining-proof-of-work/

[8]区块高度并不是唯一标识符，即可能有多个区块拥有相同的区块高度。

（Public Key）pk_i。一个账户 i 可以表示为多元组 $\langle pk_i, sk_i, s_i(\hbar)\rangle$，其中 $s_i(\hbar)$ 表示在区块高度 $\hbar$ 时用户 i 的状态[9]。

定义 2. （交易）交易描述了一组用户的状态更变[10]，具体地来说，一个交易包含交易发起账户 i 和接受账户 j 的状态更变。交易表示为 $tx_k = \langle s_i(\hbar), s_i(\hbar'), s_j(\hbar), s_j(\hbar')\rangle$，其中 $s_i(\hbar)$，$s_j(\hbar)$ 表示交易发生前账户的状态，$s_i(\hbar')$，$s_j(\hbar')$ 表示交易发生后账户的状态，并且 $\hbar \neq \hbar'$。

需要注意的是，交易也可能使账户状态不发生变化，即 $s_i(\hbar) = s_i(\hbar')$ 或者 $s_j(\hbar) = s_j(\hbar')$。

在此基础之上，我们给出区块的定义。

定义 3. （区块）区块是一个数据结构，一个区块 $B_\hbar$ 包含了一组交易 $Tx = \{tx_1, \ldots, tx_n\}$ 和区块头，其中区块头包含了该区块的哈希值 $h(B_\hbar)$、指上一个区块的哈希值 $h(B_{\hbar-1})$ 和区块时间戳。

理论上，一个区块内部可能存在多个交易修改相同的账户状态，此时这些交易在区块内部的顺序由负责打包区块的节点决定，通常会依据交易自带的时间戳。

因此在任意区块高度 $\hbar$，我们可以描述系统的状态。

定义 4. （系统状态）系统状态 $\mathcal{L}(\hbar)$ 定义为在 $\hbar$ 时，所有已经验证的区块记录的集合，即 $\mathcal{L}(\hbar) = \{B_1, B_2, \ldots, B_\hbar\}$。

不同共识机制的验证区块策略不同，具体见第5章。

4 区块提议

4.1 基础

论文 [34] 中给出了区块提议的数学描述，类似地，我们给出相关定义。

定义 5. （出块协议 Π）对于参与出块的节点 $p_i \in P$[11]，在时刻 t 执行函数 M_i，M_i 的输入为 (Tx, B', ζ_i)，其中 Tx 为待验证交易集合，B' 为某已知区块，ζ_i 为节点 p_i 的

[9]在交易网络中，用户状态即用户余额，随着智能合约和用户行为的多样化，这里统一称之为用户状态。

[10]这里假设交易发生在一对账户间，对于多对多的交易可以转换为多笔交易。

[11]不失一般性，此处 P 可以为有限或无限集合。

选举参数。M_i 的输出 $\in \Omega = \{B, \perp\}$，样本空间 Ω 为非空集合，当 $M_i = \perp$ 时，节点 p_i 无法出块；当 $M_i = B$ 时，节点 p_i 提出区块 B，并且提案区块 B 成为 B' 的后置区块，即 $\hbar_B = \hbar_{B'} + 1$。

通常来说，$M_i(Tx.B', \zeta) = B$ 的必要条件是待验证交易组 Tx 和区块 B' 有效。对于 UTXO 模型和 Account 模型，交易组 Tx 有效性验证方式不同，而区块 B' 的有效性确定也与一致性相关，本章不做详细介绍。

根据出块是否依赖于物理算力，目前大致有 PoW 和 PoS 两种出块机制。

PoW 协议最早由 [22] 提出，被用于防止拒绝服务攻击，后被比特币应用于出块节点选择 [10]。我们首先给出 PoW 出块机制 Π_w 的一般性描述。

定义 6. （出块协议 Π_w）基于 PoW 的出块协议，有 $\zeta = \langle x, c, d \rangle$，其中 x 为一个临时随机数，c 表示挑战问题，d 为正实数表示困难程度，且：

$$M_i(Tx, B', \zeta_i) = M_i(Tx, B', c, x, d) = \begin{cases} B & \text{if } h(c, x) < \frac{k}{d} \\ \perp & \text{otherwise} \end{cases} \tag{1}$$

$h(\cdot)$ 表示某种哈希函数，k 为某个正整数。

以比特币为例，挑战问题 c 为区块 B 的区块头，$h(c, x) = SHA256(SHA256(c|x))$ 且 $k = 2^{224}$，即函数 $M = B$ 的条件是 $SHA256(SHA256(c|x)) < \frac{2^{224}}{d}$。

目前涌现出许多基于 PoS 的出块协议 [14, 26, 27]，虽然这些协议实现机制略有不同，但本质上都是基于伪随机函数实现了出块节点的随机选择。在此我们给出 PoS 出块协议 Π_s 的一般描述。

定义 7. （出块协议 Π_s）基于 PoS 机制的出块协议，有 $\zeta_i = \langle s_i(\hbar_{B'}), \delta \rangle$，其中 $s_i(\hbar_{B'})$ 表示在已有区块 B' 的高度 $\hbar_{B'}$ 上节点 p_i 的状态[12]，δ 表示可验证伪随机数，且

$$M_i(Tx, B', \zeta_i) = M_i(Tx, B', s_i(\hbar_{B'}), \delta) = \begin{cases} B & \text{if } g(s_i(\hbar_{B'}), \delta) = 0 \\ \perp & \text{otherwise} \end{cases} \tag{2}$$

$g(\cdot)$ 表示某种函数。

执行协议 Π_s 的出块节点需要提供在当前高度的账户权益证明（即 $s_i(\hbar_{B'})$），当满足条件 $g(s_i(\hbar_{B'}, \delta) = 0$ 时，则可以成为出块节点。

[12] 为防止伪造，此处更为严谨的表示是 $SIG_{sk_i}(s_i(\hbar_{B'}))$，即带有签名的状态，这里简单表示为 $s_i(\hbar_{B'})$

4.2 性质分析

论文 [34] 指出出块协议需要满足（伪）随机性，即节点在执行出块操作前无法预知自己或者其他节点是否会成为出块节点。当出块行为可以预知时，会导致一些安全问题，例如自私挖矿（Selfish Mining）[35] 问题。此外，节点在获得出块权之前可能采取消极运行策略 [27]。更进一步地，目前许多链上 DApps（例如博彩应用）的随机性往往来自于出块随机性，因此当出块可预测时，此类 DApps 则会有被操控风险[13]。

同时论文 [28] 指出，由于现有网络的限制，出块频率过高会引起更多的链分叉（Fork），从而降低系统的安全性。

因此，我们认为出块协议需要满足如下性质：

- 不可预测性：任何节点在执行出块操作前无法预知自己或者其他节点是否会成为出块节点；
- 合理出块频率：整个系统的出块频率稳定，不会随着节点规模以及系统状态发生变化。

4.2.1 PoW 协议

首先我们分析基于 PoW 的出块协议 Π_w，根据公式1，对于哈希函数 $h(\cdot)$，在给定 c 情况下，目前没有比穷举更好的算法来寻找 x 使其满足 $h(c|x) < \frac{k}{d}$[36]。即在任意时刻 t 无法预测 $t+1$ 时刻 M 的输出，因此 Π_w 具有不可预测性。对于 PoW 协议，其挑战问题 c 的选择对不可预测性有非常大的影响，更详细的分析见附录B。

在给定 c 的情况下，$\Pr(M = B)$ 与困难程度 d 成反比。以比特币为例，系统会统计每 2016 个区块的总共出块时间来估计全网算力，从而动态调整 d。因此采用 PoW 的出块协议可以保证合理的出块频率。

此外，目前有许多研究分析了针对 PoW 协议的攻击，更详细的介绍见附录A。

4.2.2 PoS 协议

然后我们分析基于 PoS 的出块协议 Π_s，根据公式2，在给定 s_i 和函数 g 的情况下，Π_s 的不可预测性来自于 δ 的不可预测性。通常可验证随机数 δ 来自于当前系统状态 $\mathcal{L}(\hbar)$，例如 [14] 中的随机种子来自于上一个已确定的区块；[26] 的随机种子来自于上个朝代（Epoch）的创世块（Genesis Block）。对于 PoS 协议，其可验证随机数 δ 对不可预测性有非常大的影响，更详细的分析见附录C。

[13] https://www.bitguai.com/block/news/36731.html

在给定 δ 和函数 g 的情况下，$\Pr(M_i = B)$ 与用户状态 s_i 相关。例如 [14] 中 $\Pr(M_i = B)$ 与 ρ 正相关，ρ 为节点 p_i 的资产占总资产的比例。一般情况下，出块节点的出块频率不会随着其资产增加而无限上升，但可能使其连续出块，而产生某些安全问题 [26]。

5 一致达成

5.1 基础

基于第4章，在出块节点提出区块的情况下，验证节点[14]需要对区块的顺序达成一致，该问题可以抽象为分布式系统中的状态机复制（State Machine Replication）问题 [21]。

首先我们给出区块链中拜占庭容错的状态机复制问题定义。

定义 8. （拜占庭容错状态机复制）对于验证节点集合 V，其中最多有 f 个拜占庭节点（即最少有 $n_v - f$ 个诚实节点），所有验证节点必须从提案中最终做出决策，并且满足下述条件：

- 安全性（Safety）[15]：所有诚实（验证）节点对所有提议区块达成顺序一致性；
- 活性（Liveness）[16]：所有诚实（出块）节点提出的区块最终都会被记录；

根据 FLP 定律，在完全异步网络拜占庭环境下，不存在确定性的算法满足上述条件 [31]。因此现有区块链交易验证算法都是同步（或者所谓的半同步）网络拜占庭环境下的状态机复制算法 [14, 19]。更进一步地，在有拜占庭节点的情况下，现有的区块链共识算法在异步网络中应该优先保证安全性而舍弃活性[17]。

5.2 性质分析

除了上述的特性之外，我们需要考虑另一个重要性质，最终性（Finality）。

在区块链系统中，由于网络延迟以及恶意出块因此在同一高度可能会出现多个区块即分叉。因此所有验证节点需要对分叉进行协定，即选择某个区块从而达到共享

[14]部分系统对出块和验证节点不作区分。
[15]部分研究描述为一致性（Consistency）。
[16]部分研究描述为可终止性（Termination）和有效性（Validity）。
[17]区块链共识算法的发展趋势，https://zhuanlan.zhihu.com/p/52251671

状态。最终性是指由验证节点决策通过的共享状态不会被改变，最终性又被称为绝对最终性（Absolute Finality），对应地，存在一定概率使已通过区块状态改变称之为概率最终性（Probabilistic Finality）[18]。

根据是否具有最终性，现有的共识算法又可以划分为链式协议和基于投票协议[16]。

5.2.1 链式协议

在采用链式协议的系统中，通常并不区分出块节点和验证节点。诚实的出块节点会遵从某种规则（最长链原则或者最重子树原则）产生区块，当出现分叉后，根据上述规则所有节点达成一致共享状态。

首先，链式协议仅能保证最终一致性（Eventual Consistency）[19]，即如果不再有新的状态更新，所有节点最终就共享状态达成一致，但在某个时间段内，共享状态在各个节点所存的状态可能不一致 [37]。

同时，链式协议仅满足概率最终性，即随着链结构的增长区块状态改变的概率逐渐降低，但永远无法达到零，即仍然存在被改变可能。比特币白皮书给出了概率最终性的简单分析，基于最长链原则，当恶意节点产生区块的概率 $q = 0.1$ 时，恶意节点从 6 个区块高度前开始重新构造子链从而逆转当前主链的概率低于 0.0003[10]，因此比特币网络的交易确认时间通常约定俗成为一小时即 6 个区块高度。

随之而来的是，采用链式协议的系统需要等待一段时间才能提供足够的概率最终性，因此往往链式协议系统的延迟（Latency）都较高 [17]。

从安全角度来看，概率最终性意味着已经验证的交易会存在修改的可能，例如出现双重支付（Double-Spending）或者长距离攻击（Long-Range Attack）等 [38, 39]，更详细的分析见附录A。

5.2.2 投票协议

为了实现最终性，许多区块链系统采用了传统分布式系统的共识算法。论文 [32] 指出现有的分布式系统共识大多基于经典的 Paxos 模型 [5]，在该模型下，所有接受节点通过投票对提案者的提案进行表决，此类协议被称为投票协议。

对于联盟链为代表的系统 [40]，通常采用 PBFT[19] 和投机 BFT 算法 [29]。在 PBFT 策略中，验证节点为固定的集合，并且所有节点身份公开，在其发布 PREPARE

[18]Finality in Blockchain Consensus, https://medium.com/mechanism-labs/finality-in-blockchain-consensus-d1f83c120a9a

[19]最终一致性是弱一致性（Weak Consistency）的一种形式

消息后其决策也公开可见，因而在其发布正常 COMMIT 消息前存在被攻击或者贿赂可能。同时，此类 BFT 算法在一般情况下网络通信开销为 $O(n^2)$，因此验证节点的规模通常较小。综上对于采用 PBFT 算法的系统，无法适用于无授权环境。

为适用于更大的节点规模，Byzcoin 从出块节点中选择验证节点 [12]，Algorand 采用抽签方式选择验证节点 [14]，本质上来看，两者都通过随机方式（PoW 或者伪随机数）从一个大规模集合中选择更小规模的验证集合用于投票。同时，与 PBFT 等传统分布式共识算法不同的是，Algorand 采用的投票算法 $BA\bigstar$ 并未区分主节点和备份节点，在整个投票过程中，每个验证节点都需要多次执行 $CommitteeVote()$ 和 $CountVotes()$ 操作，即进行多次的投票和计票。Algorand 假设在整个 $BA\bigstar$ 过程中所有验证节点不会改变投票选择，在保证一定比例诚实节点的情况下，如果收不到足够的投票则问题一定出在网络上[20]。

关于投票的研究有很多，论文 [41] 指出电子投票系统需要满足若干特性，详细分析见附录D。

此外，对于共识中的投票协议，其投票效益的计算也不应被忽视。许多共识系统采用的是“一人一票”制度，即任何有投票权的验证节点只能投一票，但验证节点获取投票权的概率和其持有（或抵押）的资产相关 [14, 11]，这可能导致系统容易收到女巫攻击（Sybil Attack）[42]。在非授权的投票系统中，攻击者可以通过创建大量假名节点来增加其获取投票权的概率。因此投票过程必须能够抵抗女巫攻击，解决方法可以是通过设置准入门槛 [26]，或者采用类似于星云指数的计算方式 [43]。

6 总结

我们可以发现，区块链的共识和传统共识机制存在许多相似之处。例如对于区块一致性的达成本质上也是状态机复制，同时区块链共识在拜占庭环境下对安全性和活性的需求也并不能违背 FLP 定律。

但区块链相比传统分布式系统更为复杂, 主要表现在以下几个方面。

- 以比特币为代表的区块链系统摆脱了传统分布式系统的领袖-备份节点的设定，后者被认为是共识达成的瓶颈之一 [32]。在区块链系统中，出块节点负责发起对整个系统的状态更变，因此区块提议作为区块链共识机制中的重要部分被广泛研究，具体来说，理想的提议机制需要满足不可预测性以及合理出块频率。
- 区块链共识对于拜占庭容错的需求非常高，在大规模的异步网络下，应该优先保证安全性。同时，区块链共识需要保证已经达成一致的状态不会被改变（或

[20]Algorand 假设网络最终会收敛于同步模型

者发生改变的概率非常低）。

与此同时，我们看到越来越多的研究关注于具有扩展能力（Scale-out）的区块链共识算法，在并没有严谨定义的 Layer 1&Layer 2 概念中，分片技术（Sharding）被认为属于 Layer 1 机制，而闪电网络（Lightning Network）、状态通道（State Channel）等技术被认为属于 Layer 2 机制[21]。上述技术并不对区块链共识产生本质上的影响，我们将在后续工作中进一步阐述。

[21]https://lucidity.tech/layer-2-blockchain-technology/

附录 A　对 PoW 的进攻方式

自比特币被提出同时成为最具影响力的区块链项目以来，对其核心的 PoW 协议的种种进攻方式的提出，以及相应的对策，相应安全性的分析从未停止过。目前大家熟知的就有“51% 攻击”（主要能实现“双花攻击”），以及 2015 年 Eyal 等人提出的“私自挖矿”攻击等等，同时还存在着鲜为人知甚至潜在的尚未被发掘的进攻方式。作为共识的设计者，深入了解此类进攻方式的工作原理及影响，并分析相应的对抗策略，无疑能为所设计共识的安全性提供极大帮助。

近期，Ren Zhang 等人关于 PoW 进攻方式的研究的论文发表在安全领域顶级会议 2019S&P 上（暂无引用链接）。这里我们以这篇论文为基础，简要概述其结论，并为本设计指导提出一些对 PoW 进攻方式见解与思考。

就目前而言最为广泛的进攻方式可大致分为两大类

- 分叉攻击

  这类攻击方式就包括我们熟知的**双花攻击**：在攻击者 a 支付数字货币的交易 $a \to b$ 被区块 B 打包，并且攻击者 A 获得实际（如线下收货）收益后，以 B 区块的父区块为根进行分叉产生新区块 B'，同时 B' 包含攻击者将同一笔数字货币支付给其小号的交易 $a \to a'$，进而达到否定区块 B 及其中交易 $a \to b$ 的目的。结果 b 没能收到款但 a 收到了货。

  通常认为双花攻击需要攻击者达到全网 50% 以上算力。

  同时也包括所谓的**私自挖矿攻击**。当攻击者挖出新区块后，并不选择立即公布这个区块，而是私自在新区块上继续挖矿并不断加长更新，即，维护自己的一条“私链”。当私链长度大于主链长度时，攻击者选择一直在私链挖矿。只有当私链长度等于主链长度时，攻击者才选择公布其私链并期望私链能赢得之后的算力竞争。结论表明只要攻击者算力大于全网的 1/3，私自挖矿即可以让攻击者有利可图。具体分析见原论文 [35]。

  私自挖矿可以和双花攻击进行结合，即先在主链进行交易后再公布私链用以否定主链交易。

- 定点攻击

  这类攻击有个专业名称叫 feather-fork，最早见于比特币论坛上[22]。

  此类攻击允许攻击者在拥有即使小于 50% 算力的情况下完全隔绝任何来自某特定地址（即所谓黑名单，如 Alice）的交易。具体操作如下：

[22]https://bitcointalk.org/index.php?topic=312668.0

攻击者事先做出一个承诺（commitment[23]）：我永远不会在任何包含来自 Alice 的交易的区块上进行挖矿。攻击者会一直遵循他所做的承诺。

作为一个普通矿工，当他听到攻击者的承诺后，作为一个利益最大化的个体，他在打包交易的时候也不会包含任何来自于 Alice 的交易：如果他的区块包含了 Alice 的交易，那么在攻击者拥有算力为 α（全网百分比）的情况下，至少有 α^2 的概率攻击者连续挖到两个区块，这两个区块将接在该矿工区块的父区块上，使该矿工挖出的区块成为孤块而丧失奖励。而矿工不打包 Alice 的交易仅仅损失少量交易费。其结果是，所有理性矿工都会隔绝 Alice 的交易，达成所谓定点攻击。一般而言 α 越大能拉拢的普通矿工越多。

造成定点攻击的原因在于理性矿工与协议矿工（reference miner）的区别：理性矿工总会最大化自己的利益，而协议矿工会至始至终按协议运作。定点攻击只有在协议矿工的比例少于 50% 时才作效。

- 其他攻击（跳链，矿池）。

  跳链严格来说不是一种攻击：因为比特币的挖矿难度是根据 2 周内平均挖矿时间动态调整的，那么，很多大矿工大矿池可以选择在比特币难度较高的时候转去其他 PoW 公链挖矿，待比特币难度降下来（必然结果。因为矿工跳槽了，总算力减少）之后再回归比特币。来源见于 Bitcoin-NG [15]

  所谓矿池相关攻击不属于共识层面，是因为矿池的引入导致各式矿工行为。这里稍微介绍下仅供参考。

  Pool-hopping [44]: 类似，有的矿池是根据单位时间收益（出块奖励/挖矿时间）来分配奖励。那么，矿工可以在某矿池挖了一段时间没挖出矿后跳到别的矿池去挖矿，因为在原矿池继续挖即使挖出来了因时间太长收益也低。一般为达到平均时间的 43.5% 即跳槽。

  派间谍（Miners' dilemma [45]）：简单而言，矿池 A 可以派一部分矿工，所谓间谍，去矿池 B 挖矿，但是间谍挖到真正的矿不会提交给 B 矿池，只提交挖矿的证明。（提交 share，难度为矿的千分之一）。相当于，间谍从 B 矿池领工资但不真正挖矿，工资分给 A 矿池的人。（当然 B 矿池也会向 A 矿池派间谍，形成一个类似囚徒困境的局面）。

文章接下来分析了某些著名的 PoW 项目针对这些攻击的安全性，同时提出了几项评价指标。这里不详细介绍。其重点结论在于，针对上述进攻安全性不能同时满足：存在一个安全性悖论："rewarding the bad and punish the good"。具体而言，对于区块链分叉，一般存在两种处理方式：

[23]commitment 是博弈论中一个重要概念。见 https://en.wikipedia.org/wiki/Stackelberg_competition

- 对所有分叉同给予同样奖励，所谓“reward-all”。举例包括 fruitchain，EthPoW（叔块）等。此类项目由于分叉没有损失，会加剧分叉攻击。
- 对分叉进行惩罚，所谓“punishment”，如将出块奖励均匀分发给各个分叉。举例包括 DECOR+，Bahack‘s idea。此类项目由于分叉损失过大，理性矿工会更加担心自己挖出的块成为孤块，进而加剧定点攻击。
- 还有一类叫做“reward lucky”。此类协议奖励某些区块，定义比较模糊。举例如 Subchains,Botail。但是文章认为 lucy 不等于 good，也不能达到效果。

所以，该论文给我们的思考在于，设计共识应达成上述安全性的一个平衡。

文章最后给出的共识参考建议也值得一提：

- 设计的协议不应该太复杂。
- 不应只针对特定的攻击来进行安全性分析。(应全面考虑)
- 不应针对特定攻击者奖励进行安全性分析。(应全面考虑)

另外，文章指出安全性能基于下面几项要素得到提高

- 网络环境更好
- （弱）全局时钟的存在
- 可信赖的第三方
- 责任外包制度
- 基于“Layer 2”的抗攻击手段。

附录 B 谜题选择

谜题（puzzle）在 PoW 协议中扮演重要角色。通常，PoW 协议规定只有解决给定谜题的矿工拥有出块权，是一种抵抗女巫攻击的有效手段。有文章指出 PoW 本质是通过谜题实现一个分布式时钟[24]。

谜题的选择同样也面临各式各样的取舍：

[24]https://grisha.org/blog/2018/01/23/explaining-proof-of-work/

- 谜题固然需要一定的难度来防止女巫攻击，但另一方面，有研究表明对于任何算力竞赛模型，高难度的谜题存在马太效应，更容易造成大户垄断(51%dominance)。

- 谜题的难度固然需要动态调整以适应不断升级的算力，但另一方面，动态难度会造成跳链现象（见章节A)。同时，动态难度也会遭受所谓长程攻击（long-range attack)，即攻击者从某远古区块开始一直以极低算力挖一条私链，因难度是动态的私链增长速率可以和主链一致，然后攻击者一定时间段突然加大算力，使私链长度大于主链。通常解决长程攻击的方式为矿工检测到分叉时，除简单的采取最长链原则外同时也要检测区块难度，以摒弃难度过低的链。

鉴于谜题在 PoW 协议中的重要作用，作为区块链共识设计者，了解包括比特币在内的多个 PoW 所采用的谜题及工作原理，优势劣势等，也是设计合理的 PoW 共识必不可少的一部分。

本章节主要针对 Mimblewimble 共识协议（Grin 项目）所采取的谜题进行介绍并展开思考。该谜题名称叫 cuckoo，发表在 2015 年 FC 上 [46]。

比特币的挖矿工具经历了 CPU，GPU，FPGA，ASIC 四个阶段。现今有比特大陆等矿机公司已经本质上实现了比特币的算力垄断。而 cuckoo 旨在提出一种新的谜题，使得挖矿工具的更新停留在 GPU 这一步——只有 1060 以上显卡才能进行挖矿。

谜题的本质是验证（verification）与探索（proof attempt）的不对称性。比特币所采取的 SHA256 由于哈希函数的难逆性无疑符合条件。cuckoo 采取的是随机图找环算法，通过引入内存带宽限制构建谜题的难度。具体步骤如下:

- 图的生成。

  二部图的 N 个点已经给定，通过哈希函数随机生成二部图的 M 条边（大约 $N/2$)。生成边的方式要满足一定的要求，可以理解为每条边是根据 $(k, nonce)$ 的 SHA512 值决定，其中 k 为编号，$nonce$ 为矿工尝试的数字。验证时，一旦给出 nonce 则可还原出图中所有的边。

- 谜题目标。

  给定一个图，矿工需要给出一个长为 L 的环。验证时，一旦给出图以及 L 个点的编号，可以轻易验证环是否能形成。

  值得一提的是，从一个图里面寻找长为 L 的环是多项式时间可解的[25]。然而由于图的生成是完全随机的，且每个图大概率不存在长为 L 的环，故矿工仍然需要暴力搜索。

[25] L 为偶数时，时间复杂度为 n^2。L 为奇数时，时间复杂度为 $M(n)$，其中 $M(n)$ 为计算矩阵乘法所需的时间复杂度。

- 找环推荐算法。

  文章推荐算法包含两部分。

  - 减支部分

    所谓减支本质上是完成一个拓扑排序问题：去掉所有度数为 1 的以及相邻的边，重复上述过程直到所有点的度数 ≥ 2。由于上述减支过程需要存储每个点的度数，故需要进行大量内存读取。这就是该谜题能引入内存带宽限制的原因。

    文章同时也提出了其他的减支算法，$BFS(L)$ 和 $BFS(L/2)$，能避免对每个点都记录信息，但会消耗更多的时间，是一种时间和存储的平衡（TMTO，time-memory trade-off）

  - 找环部分。

    文章推荐的找环算法维护一个有向图森林，以类似并查集的方式将边逐条加入。一开始，所有孤立点各自都是一座森林。一旦一条边加入，如果两个端点属于两个不同森林，则将两个森林合并，通过维护森林中边的指向与每个节点的 root 值。当且仅当新加入边的两个端点属于同一森林时，则必存在一个环，可根据有向图路径找到该环并确定长度。

    值得一提的是，如果找出来环的长度不为 L，则忽略该条边继续上述操作。这样虽然可能导致有的长为 L 的环被漏掉，但这种情形概率不高，作为一种概率性的算法仍能保证高效性[26]。

文章推荐的数据规模 $N = 2^{25} + 2^{25}$，$L = 42$

文章接下来给出了很多实验图表。这里不一一列出，仅简要介绍结论。

- 计算哈希函数的时间开销随着节点规模增大而减小，最终低于 15%。（大部分时间用于找环）
- 存在 42 环的概率在 $M/N > 1/2$ 时剧烈增长。
- 内存的读开销随着已尝试 nonce 的百分比指数级上升，但写开销持平。
- 存在环的概率与 L 大约成反比。

总结：就目前而言 cuckoo 能限制矿机，但同时也需要考虑到新型矿机的可能性（如基于路由器的大带宽）。

[26]高效概率性算法在实际运作中比比皆是。一个经典的例子是质数判定问题 $Prime()$。有研究已经证明该问题是多项式时间可解，但时间复杂度仍然很高。实际运行时人们仍然选择用费马小定理进行判断。后者不能保证 100% 正确但更快。另一个例子是线性规划问题，虽然已被证明椭圆算法能在多项式时间解决，但人们更多的还是采用单纯形法，后者不能保证多项式时间解决，但实际平均运行时间往往更低。

附录 C　随机数的生成

所有涉及委员会选举，权益证明的区块链都离不开随机数的应用。而区块链上的随机数与人们传统理解又有所不同：区块链本质是实现一个状态机复制的问题，故要求所有的节点以相同条件生成随机数都会得到同样的结果，这就杜绝了以物理方式生成随机数的可能（包括宇宙辐射，random.org，等等）。

区块链上的随机数基本要求是不可预测性与可验证性，否则不能满足区块链的需求：首先，出块权这种不能被预测，否则会引发一系列问题（DDOS 攻击）。同时，区块链一切算法是公开的，不存在一个中心节点秘密生成随机数。这就要求所有人都能验证随机数的合法性。

此章节以 Randao 白皮书为参考[27]，指出随机数（或随机种子）生成需要权衡的几个问题：

最后演员问题"last actor problem"

当随机数的生成需要多人合作时，最后一个做出行动的成员在其他成员行动后可以预知随机数的值，而其余成员因缺少最后行动者的行动无法获知，造成信息不对称。而一旦最后行动者发现随机数对自己不利，他可以选择拒绝行动。

Algorand 生成随机种子的方法为，以上一轮出块者的 VRF 函数（可验证）作为下一轮的随机种子。因为 Algorand 算法无法预测出块者，故能满足不可预测性。

但我们认为 Algorand 中的算法存在最后演员问题：出块权是根据账户优先级决定的，而优先级来源于每个出块者的 VRF，需要进行广播。而一旦一个矿工已经接受到所有其他矿工的广播，然后发现自己的优先级是最高的，他可以提前知道自己是出块者（只要他在限定时间内广播并且所有节点正常运作）并知道下一轮随机种子。进而他也可以选择不出块来改变区块链的结果（outcome）。

Dfinity 和 randao 采用 BLS（(t,n) 门限签名）方式生成随机数。因为在得到 t 个独立签名之前无法恢复出组签名，故所有用户均无法预测组签名值，满足不可预测性。但是也存在最后演员问题：当一个用户收到 $t-1$ 个独立签名之后，他可以提前恢复出组签名，获取随机种子，进而也可以根据结果选择拒绝运作。当然，只要剩余 $n-t+1$ 个人不都怎么做，组签名还是能成功运作。所以，该算法能抵御非法成员少于 $n-t+1$ 的最后演员问题，且 t 越大越难抵御。

但是，如果 t 太小的另一个明显的问题是，任何 t 个成员可以共谋，无视剩余 $n-t$ 个成员进行所有组签名操作。故 BLS 能抵抗 $\min\{n-t, t-1\}$ 个非法成员。所以一般选取 $t=(n-1)/2$。

同时，BLS 的另一个问题是生成的随机种子随机性不能满足。所谓随机性指任何

[27]https://www.randao.org/whitepaper/Randao_v0.85.pdf

一个对手无法在多项式时间内区分算法返回的随机数和一个真正的随机数。Algorand 的 VRF 签名能满足随机性要求，然而门限签名由于限制颇多，并不能理论保证返回结果的随机性。Randao 采取的方式是选取多个组串行进行签名，即，前一组的输出结果作为下一组的输入进行门限签名，但仍没有理论证明。

附录 D 投票系统

目前关于电子投票（Electronic Voting）的研究中 [41]，除去传统的隐私性（Privacy）、公平性（Fairness）和健壮性（Robustness）需求，理想的电子投票还需要满足如下要求：

- 可校验性（Universal-Verifiability）：任何第三方都可以验证最后的投票结果是否正确统计了合法选票[28]；
- 无收据性（Receipt-Freeness）：投票者无法向第三方证明其所投的选票内容；
- 无争议性（Dispute-Freeness）：任何第三方都可以验证协议的参与方是否正确执行了协议；
- 自计票性（Self-Tallying）：任何第三方可以进行计票，而不需要可信第三方或者投票者的参与；
- 完善保密性（Perfect Ballot Secrecy）：假设存在 n 个选民，任何 t 个（$t < n$）投票者的投票结果只有剩余 $n - t$ 个投票者串通起来才能知道。

而对于共识机制中的投票系统，由于不存在可信的第三方机构，每个验证节点既是投票者也是计票者(即所谓的"all voters are talliers")，因此其必须满足可校验性和自计票性。论文 [41] 指出在大规模的投票系统中，并不需要满足完善保密性。

现有大多数基于投票的共识机制都无法满足无收据性和无争议性，具体地来说，对于拜占庭验证节点，虽然无法伪造或篡改其他人的消息[29]，但仍可能出现如下恶意行为：

- 恶意投票：恶意验证者不发布任何投票或者对其他验证者发布不同的投票，例如对部分验证者发布 $\langle pk_i, sign_{sk_i}(t, h(B_1), \pi_i)\rangle$，而对另一部分验证者发布 $\langle pk_i, sign_{sk_i}(t, h(B_2), \pi_i)\rangle$，$B_1 \neq B_2$。

[28] 另有原子可校验性描述仅投票者可以验证投票结果是否正确统计了合法选票。
[29] 通常而言，我们认为现有的签名算法可以保证信息无法篡改或者伪造。

- 割裂网络：在采用 Gossip 协议传输的网络中，恶意节点可能在收到其他节点的投票后不向其他节点转发该信息，导致原有投票信息无法广播到所有节点。
- 共谋：恶意节点在投票前或者投票过程中得知其他验证者身份，从而贿赂其他验证者使其投票决策发生变化。

作为分布式系统，区块链中所有验证节点通过 P2P 方式进行通信，因此任何节点在投票过程中都无法检验其他节点是否正确执行协议，即在投票过程中无争议性无法保证。[30]在这种情况下，恶意投票和割裂网络将变得可行。这两种行为会导致决策无法收敛，从而影响共识的活性。

共谋行为则违背了无收据性，现有大部分投票共识都没有考虑拜占庭节点的共谋行为并且假设系统中拜占庭节点比例低于某个阈值（例如三分之一），而实际上，由于共谋行为的存在，系统中拜占庭节点比例会更高。

附录 E　匿名性

通常，委员会选举，投票过程，包括某些共识过程都对匿名性有要求。匿名性能抵御贿赂现象，DDOS 攻击，打击报复等等。Algorand 通过每次抽取不同的委员会成员来投票的方式实现了部分的匿名性。

（未完待续）

[30]尽管 Casper 通过 Slash 机制实现了检点的互相监督，但恶意行为被发现是基于交易已经上链的前提。

参考文献

[1] L. Lamport, "The implementation of reliable distributed multiprocess systems," Computer Networks (1976), vol. 2, no. 2, pp. 95–114, 1978.

[2] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," Journal of the ACM (JACM), vol. 27, no. 2, pp. 228–234, 1980.

[3] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber, "Some constraints and tradeoffs in the design of network communications," in ACM SIGOPS Operating Systems Review, vol. 9, pp. 67–74, ACM, 1975.

[4] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 4, no. 3, pp. 382–401, 1982.

[5] L. Lamport et al., "Paxos made simple," ACM Sigact News, vol. 32, no. 4, pp. 18–25, 2001.

[6] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: an engineering perspective," in Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing, pp. 398–407, ACM, 2007.

[7] M. Burrows, "The chubby lock service for loosely-coupled distributed systems," in Proceedings of the 7th symposium on Operating systems design and implementation, pp. 335–350, USENIX Association, 2006.

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, p. 4, 2008.

[9] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," 2003.

[10] S. Nakamoto et al., "Bitcoin: A peer-to-peer electronic cash system," 2008.

[11] E. Buchman, Tendermint: Byzantine fault tolerance in the age of blockchains. PhD thesis, 2016.

[12] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 279–296, 2016.

[13] "Ultrain 共识黄皮书," 2019.

[14] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68, ACM, 2017.

[15] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in 13th USENIX Symposium on Networked Systems Design and Implementation NSDI'16, pp. 45–59, 2016.

[16] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," IEEE Access, vol. 7, pp. 22328–22370, 2019.

[17] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," arXiv preprint arXiv:1711.03936, 2017.

[18] Y. Zeng, "Consensus survey," 2019.

[19] M. Castro, B. Liskov, et al., "Practical byzantine fault tolerance," in OSDI, vol. 99, pp. 173–186, 1999.

[20] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, pp. 1–32, 2014.

[21] R. Pass and E. Shi, "Rethinking large-scale consensus," in 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pp. 115–129, IEEE, 2017.

[22] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in Annual International Cryptology Conference, pp. 139–147, Springer, 1992.

[23] I. Grigg, "Eos-an introduction," 2017.

[24] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper, August, vol. 19, 2012.

[25] V. Buterin and V. Griffith, "Casper the friendly finality gadget," arXiv preprint arXiv:1710.09437, 2017.

[26] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 66–98, Springer, 2018.

[27] T. Hanke, M. Movahedi, and D. Williams, "Difinity technology overview series–consensus system (rev. 1)," 2018.

[28] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in International Conference on Financial Cryptography and Data Security, pp. 507–527, Springer, 2015.

[29] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in ACM SIGOPS Operating Systems Review, vol. 41, pp. 45–58, ACM, 2007.

[30] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598, IEEE, 2018.

[31] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process.," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1982.

[32] Howard, Distributed Consensus Revised. PhD thesis, 2019.

[33] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.

[34] J. Brown-Cohen, A. Narayanan, C.-A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," arXiv preprint arXiv:1809.06528, 2018.

[35] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," Communications of the ACM, vol. 61, no. 7, pp. 95–102, 2018.

[36] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," in International workshop on selected areas in cryptography, pp. 175–193, Springer, 2003.

[37] E. A. Brewer, "Towards robust distributed systems," in PODC, vol. 7, 2000.

[38] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 281–310, Springer, 2015.

[39] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," arXiv preprint arXiv:1312.7013, 2013.

[40] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in Proceedings of the Thirteenth EuroSys Conference, p. 30, ACM, 2018.

[41] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in International Workshop on Public Key Cryptography, pp. 141–158, Springer, 2002.

[42] J. R. Douceur, "The sybil attack," in International workshop on peer-to-peer systems, pp. 251–260, Springer, 2002.

[43] "Nebulas yellowpaper." `https://nebulas.io/docs/NebulasYellowpaperZh.pdf`.

[44] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," arXiv preprint arXiv:1112.4980, 2011.

[45] I. Eyal, "The miner's dilemma," in 2015 IEEE Symposium on Security and Privacy, pp. 89–103, IEEE, 2015.

[46] J. Tromp, "Cuckoo cycle: a memory bound graph-theoretic proof-of-work," in International Conference on Financial Cryptography and Data Security, pp. 49–62, Springer, 2015.