Version 1.02

TRUECHAIN：高性能的分散式公共区块链技术

初链研究小组
ERIC ZHANG 1, HENDRIK C2, YANG LIU 3, ARCHIT SHARMA 4, JASPER L 5,

1 ERIC@TRUECHAIN.PRO
2 HENDRIK@TRUECHAIN.PRO
3 LIUYANG@TRUECHAIN.PRO
4 ARCHIT@PM.ME
5 JASPER@TRUECHAIN.PRO

摘要：

在本文中，我们介绍了初链的 Minerva 共识协议的初步设计和其他的一些技术细节。当前，区块链行业普遍认为公链很难同时做到高性能，分布式，和安全。在中本聪所提出的中本链（Nakamoto 链））这类公链（低性能）的情况下也确实如此，或者是目前最受欢迎的区块写入方案-权益证明（pos）也如此。初链的共识方案是设计一个具有一致性，灵活，可最终确认交易的，具有安全保障的混合共识。我们将提出了一个想法：在以太坊的虚拟机的基础上我们增加了基于在一个无权限的环境中许可链的事务处理功能。我们也使用数据分片和可推测交易以及评估的概念在对碎片友好型的虚拟机合约中。最后，我们将从根本上简要讨论一下 ASIC 抗性挖掘算法，Truehash。

1 介绍

随着加密货币的日益普及，区块链技术引起了业界和学术界的关注。我们可以将区块链看作一个共享的计算机环境，通过普遍认可的共识机制达成协议，该环境下的所有节点可以自由地加入和退出。区块链的去中心化特性，以及交易透明性、自治性和不可篡改特性，对于加密货币来说是至关重要的，这些特性为此类系统定下了基调。

然而，早期设计的加密货币，如比特币[23]和以太坊[11]，在交易效率方面已被广泛认为是不可扩展的，而且在效率与经济上也不可行，因为它们需要大量的能源消耗和计算能力。而后出现的一些有力竞争公链项目，如基于代理权益证明（DPoS）共识算法的 EOS，通过牺牲去中心化的理念，从而实现更高的交易吞吐量（tps）。

随着现实生活中基于公链开发应用程序和应用平台的需求不断增长，一种最大可能实现去中心化且支持更高交易效率的安全协议是新一代公链系统的主要关注点。例如，考虑到一个通用的公链系统，它可以在一个非常庞大的用户基础上支持密集型的点对点游戏应用程序。在这样的链中，如果它还给数字广告、在线教育课程和去中心化交易提供智能合约服务，那么我们很容易预料交易确认时间将出现巨大的延迟。

还有一些其他模型，如权益证明的委托机制和实用拜占庭容错协议(PBFT)[13]。PBFT 共识协议，只要系统中每次都不能超过三分之一的参与者是有意或无意的恶意节点，那么它就能够确保该系统的安全性。拜占庭的假设条件在有许可认证的环境下是很容易满足的。然而在一

个没有许可认证的环境下，这是很难得到保证的。DPoS 通过给加入拜占庭委员会设置条件阻碍和建立一套委员会成员作恶的严厉惩罚机制来试图减轻这个问题，效果非常有效。即，利益相关者使用令牌加权投票系统产生拜占庭委员会（通常少于 30 个节点）。然而，去中心化的理念在这里丢失了，因为我们可以预期一些节点每一轮都会被网络中最大的令牌持有者投票选举成功。

在本文中，我们提出了 Minerva 共识，一种混合共识协议[26]，它结合了一种改进版的 PBFT(实用拜占庭)[13]和 POW(工作量证明)共识。POW 共识确保了激励，委员会的选举和委员会审计，而 PBFT 层则承担一种具有瞬时处理高吞吐量事物、交易验证、公平交易贸易委员会的成员轮值功能的高效共识机制，以及作为一种补偿基础设施去处理不同的基础设施。混合共识机制的特性允许它最大限度地容忍三分之一对等节点的腐败。

2 背景

2.1 相关工作

我们共识的核心优势在于对由 Pass 和 Shi 提出的混合共识[26]论文中理论的认知和应用。我们受益于这样一个事实：从那篇论文中，我们明白还有大量的设计空间可以进行更好的优化。DAILYBFT 作为委员会成员的使用，实现了轮值委员会提供验证节点共识更公平的特性。

混合共识遵循一种设计范式，即 PBFT 和 PoW 结合在一起，使其在两个方面的优良特性都得到最好的体现。一般来说，混合共识将利用 PBFT 协议作为快速处理大量传入交易的快捷途径。在默认情况下，PBFT 协议应用于一个许可的设置中，里面所有的身份都是先验已知的。PoW 协议选择 BFT 委员会成员的依据是 csize(挖出的区块数量)和节点权益的结合。这就提供了一种必要的准入系统，以处理动态的成员以及在无认证许可的环境下切换委员会。

2.2 假设

设计一个 Minerva 共识协议，满足在无认证许可的环境下运行，该环境下每个节点通过共识都可以更新他们的日志。将网络延迟考虑进来，每个节点的视图可能看起来与别人不同。因此，一般来说，网络将处于异步状态，同步状态只发生在最后 λ 个区块之前，其中 λ 是自然数。

定义节点 i 在时间 t 下的日志视图是 LOG(t, i)。下列安全要求必须以压倒性的概率保证。

- 一致性：如果 i 是一个诚实节点，那么独立于 t 存在 $\lambda > 0$，这样 LOG(t − λ, i) 相对于 i 是常数。我们用 LOG(t).表示这个值。
- 活力性：定义 TXs(t, j) 是诚实节点 j 在 t 时刻的交易数据。那么独立于 t 存在 $\tau > 0$, 这样 TXs(t, i) $\subseteq$ LOG(t + τ , i)可以表示成所有的诚实节点 i 的交易表达式。

为确保上述安全要求得到满足，以下参数在管理链安全方面将发挥重要作用。

- 公链安全系数 Qfast：在拜占庭委员会诚实节点的占比。

  Qsnail：在 pow 慢链显示最后 λ 个区块被挖矿成功诚实节点的比例。

在认证许可的拜占庭网络，大家都会认为假设链的安全系数大于 2/3 + ϱ，那么这条链是安全的。但是在没有认证许可的网络下，为满足这个安全要求，我们必须发明一个新的协议去保证链的安全系数始终处于压倒似的比例大于 e 2/3+ϱ。

2.3 敌对模型

我们的敌对模型如下[26]中的假设，即允许敌对节点轻度自适应地破坏任何节点，而腐败不会立即生效。在黄皮书的 4 部分，我们将正式地解释敌对模型，并讨论在 Universal Composabililty 模型和初链（高度去中心化的公共账本）中的修改。

3 区块链、区块、状态和交易

现在，我们介绍初链的区块链结果和他们的底层细节。

3.1 区块链

我们链的设计大部分是基于 Pass 和 Shi [26]提出的混合共识开发的，对于这个共识机制，我们做了很多的修改和改进，目的是适应我们关注的应用场景。在这个小节中我们将概述 Minerva 协议，而主要关注区块链的改进。关于共识的细节留给下一部分。

在[26]提出的协议下，混合链是两条链组成的。一个慢链和一个快链，其整个账本历史分别由 LOG 日志和 log 日志表示。快链可以认为是 DailyBFT 链的联合，其中每个 DailyBFT 在其任期内链接拜占庭委员会的日志输出。

快链上的节点作为拜占庭委员会的成员，他们通过 PBFT 投票[13]达成共识。交易和智能合约在快链上执行，实现高吞吐量。每次一旦到某个设定的时间 T，拜占庭委员会的成员将随机的进行轮换。接任的委员会是从优秀的 POW 矿工们选出来的。这将巩固了去中心化理念，因为任何拥有计算机的人都可以参与成为挖矿节点。请注意，在极限情况下，我们将委员会的轮换频率设为一个慢块，委员会的成员个数等于 1，我们将恢复传统的 PoW 共识。

对于本节的其余部分，我们将详细讨论链的每个组件。

3.2 快链

在[13]的经典设置中，认证许可的 BFT 委员会是一组节点，它们能够相互通信、投票同意或不同意主节点提出的请求。我们不假设这些节点彼此信任，事实上，这些节点的一个子集可能是被破坏的敌手。通过对[19]中拜占庭将军问题的深入分析，我们得出结论，当存在 2/3 + ϱ个诚实节点，共识总是能达成的。

在这个假设下，一笔交易在快链中达成过程如下：

•主节点提出一组他认为有效的交易请求 TXs。也就是，发送者对交易进行签名，如果执行交易产生任何的非法状态。主节点将进行数字签名和给其他委员会成员广播。

•接收到请求后，委员会成员将对 TXs 进行验证。如果有效，签名并广播。

•当接收到 2/3 + ϱ 个同意投票后，委员会的成员节点将在他们的快链日志中更新 TXs。

•现在 TXs 被打包到一个快块（快链中的块）中。这个块将作为消息 m 被广播给 PoW 矿工节点，m 中包含摘要和序列号。

在无认证许可的环境下，我们需要改变这个共识，因为保证必须的 2/3 + ϱ个诚实节点是非常困难的。相反，我需要发明一种新的共识去确保 Qfast 在敌对模型（4.2 部分描述）下始终大于 2/3 + ϱ 。我们将更加深入的讨论链的安全管理。

3.3 慢链

现在基于 PoW 共识最大的挑战是效率和可扩展性。缓慢的执行和确认时间使得它不适合开发复杂的应用程序，并且过度的能源消耗使得它不利于环境。不同于以太坊需要所有交易和智能合约在全网上所有节点执行，轮值拜占庭委员会处理大量的主要工作，而 PoW（慢链）仅仅是被用来通过工作量证明的择优程序挑选委员会成员。在本小节，我们将简洁地描述这一流程。

慢链，我们第一个想到的是比特币和以太坊用的区块链结构，通常称之为中本链（Nakamoto 链）。这里，我们定义一个区块 B = (h−1, blockdata, h), 其中 blockdata 表示包含交易和区块数字签名的区块数据；h 是通过挖矿产生的哈希值；h−1 是上一个区块的哈希值。通过哈希数据的连续链接，块形成链式结构，称为区块链。

如果中本链被用作慢链，那么区块数据基本上可以简化为在慢链块期间在快链中发生的所有交易的摘要。其动机在于，虽然快链中的数据内容对每个人都是公共可访问的，但是跨 PoW 节点存储数千或数百万个副本将非常浪费。PoW 节点只需要存储足够的数据来确保快链交易的有效性，而不是交易数据本身。只有希望参与 PBFT 的节点，才需要同步到快链。

中本链存在一些缺陷，其中最重要的问题是自私挖矿。针对这个问题，我们要求慢链的安全系数 Qsnail > 3/4 来确保 Qfast > 2/3，即使假设矿工在有机会时也会自私地采矿。这里，我们只是简短地描述它工作的基本原理。

水果链包含区块和水果，区块与区块之间都有关联，就像中本链一样，同时每个区块包含大量的水果。挖矿难度的设定是用来保证每 10 分钟才挖出一个区块，而水果 1 秒钟挖出一个。网络传输的异步性意味着我们只能期望诚实节点在日志中早于当前时刻 t 的 t − λ时刻之前完成共识，其中λ>0。因此，由于挖掘难度低，节点在挖掘水果时会有不同的日志视图。所以，只要水果所属的区块离现在的区块距离不远，我们就允许它们乱采。

在我们的应用中，每个水果本质上是对应快链块的消化。水果裸露一个由快链给出的序列号，这是为了当尘埃落定时，区块矿工们可以恢复正确的历史交易顺序。所有 PoW 节点必须验证快链块的有效性，在可能执行任何挖掘计算之前，快链块将成为区块头部分的一部分。因此，初链的交易由所有 POW 节点验证，同时仍实现高吞吐量。

明确地讲，定义一个区块 B = (h block −1 , fruitdata, h)和一个水果 F = (h fruit −1 , digest, serial, h), 其中 h block −1 是上一个区块的哈希，h fruit −1 是任意最近 λ 个区块的哈希。我们取值 λ = 17 作为水果的最新参数。最终，摘要是在快链里的交易数据的根哈希。

我们给慢链流程做个总结。
•假设 Qfast > 2/3。一组交易请求发起后，会平均每隔 1 秒钟产生一个快链块。拜占庭委员会将归纳出一个带序号的交易摘要，同时以消息 m 的形式广播给其他 PoW 矿工们。
•PoW 矿工们会通过解决一个简单的数学难题来打包 m 成一个水果。设定挖矿难度，使每隔一秒产生一个水果，并与快链的出块频率搭配。水果可以被无序挖掘，因为它们的父哈希可以被划定到在任意慢链之前λ个区块中。
•慢链区块是交易的最终结果。一个区块每 10 分钟挖掘一次，并且它包含直到那个时期挖掘的所有具有相邻序列号的水果。

3.4 水果链的意图

一个运转良好的 BFT 委员会要求其成员中有 2/3 个是诚实的[13]，因此要求快链安全系数 Qfast＞2/3。使用中本链作为慢链的唯一实现将容易受到明显自私的挖掘攻击策略的攻击。如果一个自私的矿工控制了超过 25%的区块链的算力，那么它可以控制超过 33%区块的产生[24][16]。根据[26]中描述的程序，被选入 BFT 委员会的概率等于一个节点生产区块的占比。因此，自私的矿工很可能控制 BFT 委员会的 1/3 以上。如果它恰好是敌对的，那么 BFT 协议现在将受到损害。

最坏的情况可能是通过在[29]中所示的策略实现的。如果一个自私的矿工控制 Etherum 型区块链中 40%的算力，那么它可以通过优化的自私挖掘程序来控制 70%的块生产。根据[26]中

的委员会选举程序，它将拥有拜占庭委员会 70%以上的控制权。拜占庭委员会不仅妥协，自私的矿工将独裁统治，按照它的意愿宣布任何诚实的委员会成员“不诚实”。

我们选择水果链[25]作为我们底层慢链链的混合共识。水果链的实际形成过程将在后面的小节中详细解释。正如在[25]中的公平性定理所言，水果链更能抵抗自私的挖矿。我们要求的链，在 PBFT+水果链的混合共识下，需要慢链安全系数 Qsnail = 2/3 + ϱ 来维持快链的安全，相对的在 PBFT+PoW 混合共识下，慢链的安全系数 Qsnail = 3/4 +ϱ。中本链所需的额外链安全是为了抵消自我挖矿实践中的安全损失。

但是，如果攻击者直接控制区块链的哈希算力的 33%，那么 BFT 委员会仍然很脆弱。因此，我们将进一步偏离[26]和[25]以减轻的问题。我们需要两种不受欢迎的极端情况找到平衡

有两个不必要的极端，我们需要找到一个平衡点。
•随机性。通过 VRF[22]选择 BFT 委员会成员。这对防御女巫攻击很脆弱。
•任何概率性的选择，可能最终归咎于算力。BFT 委员会对那些富含哈希权力的矿池很容易受到攻击。

我们提出的解决方案如下。当一个诚实的 BFT 节点的链长度达到 λ 时，它将发布链中每个水果的唯一矿机 ID 作为候选（或者，挖掘多于水果的每个矿机 D）。新的 BFT 委员会是从概率一致的候选中随机选出的。在这种方案下，明显的女巫攻击是不可能的，因为我们需要最低级别的 POW 才能成为候选者。对于一个大型矿业公司来说，BFT 委员会达成妥协都不容易。挖掘一个水果将比一个区块更容易被挖掘，而且容易程度是很大数量级的。

关于 BFT 委员会选举的细节将包含在下一节中。在下一小节中，我们将解释快链和水果链的相互作用，以及它们如何处理交易和智能合同。

3.5 区块
3.5.1. 区块结构。在初链中，我们有三种类型的区块需要解释。分别是快链块、水果和慢链块。它们以以下方式相互作用：PBFT 产生快链块，然后以消息 m 的形式广播给 PoW 节点们。PoW 节点们在打包水果到区块之前，将率先将 m 作为水果进行挖矿。

一个水果用数组 f 表示，即 f = (h−1; h ′ ; η; digest; m; h)，而区块的数组 b = ((h−1; h ′ ; η; digest; m; h), F)，参数代表意思是：
•h−1 相应区块的上一个哈希，只对水果认证有效
•h ′ 某个包含水果的区块，只对认证区块有效
•η 是随机数
•摘要是一种抗碰撞哈希函数，用于检查水果的有效性。
•m 是水果里的记录内容
•h = H(h−1; h ′ ; η, d(F); m)是水果或区块产生哈希的函数
•F 是一组有效的水果集，在[25]中有定义

区块链 chain = {chain[0], chain[1], ..., chain[l]}，是通过序号 i 排序的独立区块的集合，其中 chain[i].h−1 = chain[i − 1].h，并且 l 是链的长度。如果链 f ∈ {chain[l − λ + 1].F ∪⋯∪ chain[l].F}，我们将定义水果 f λ-’recent’ w.r.t. 。在我们应用中，取 λ = 17。

3.5.2. 区块中的数据。初链中有两个类型的区块，分别是快链块和慢链块。涉及的产生 hash 的散列算法是 Sha3。
快链块包含以下字段：
•ParentHash：父区块区块头的哈希值
•StateRoot：交易执行后，默克尔状态树的根哈希
•Transactions root: 交易数据的默克尔树根哈希，该树拥有交易列表中每笔交易
•ReceiptHash: 收据默克尔树的根哈希，该树拥有交易列表中每笔交易收据
•Proposer: 交易发起者的地址
•Bloom: 布隆过滤器
•SnailHash: 矿工地址，即挖矿受益者地址
•SnailNumber: 挖矿奖励块的高度
•Number: 祖先块的数量
•GasLimit: 每个区块花费的上下限
•GasUsed: 本区块所有的花费
•Time: 时间戳
•Extra：杂凑使用的预分配空间

慢链结构包含的字段如下：
•Parent hash (h−1): 父区块区块头的哈希值
•Uncle hash: 叔块的哈希
•Coinbase: 挖矿节点地址
•PointerHash: 包裹水果的区块哈希
•PointerNumber: 包裹水果的区块高度
•FruitsHash: 水果的哈希
•Fasthash: 快链块哈希
•FastNumber: 快链块高度
•SignHash: PBFT 成员签名后的哈希
•Bloom: 布隆过滤器
•Difficulty: 区块挖矿难度
•FruitDifficulty: 水果挖矿难度
•Number: 祖先区块数量
•Publickey: 公钥，即提出快链块请求的主节点的椭圆曲线数字签名公钥
•ToElect: 这个区块后委员会要不要换届
•Time: 时间戳
•Extra: 杂凑使用的预分配空间
•MixDigest: 快链块数据摘要
•Nonce: 随机值

3.6 水果链的形成过程
继承[25]中的变量定义（第 14-16 页），水果链由区块链组成，每个块包含自己的一组水果。BFT 所执行的交易最初被打包为记录 m，作为一个水果被挖掘。当下一个区块被挖掘时，比最近参数 R 更近的水果将被打包到一个区块中。

矿工将只运行一个挖矿算法，它从随机预言机产生哈希值 h。当[h]− κ < Dpf 时，将会挖到一个水果，而当 n [h] κ < Dp 时，将会挖到一个区块。其中 Dpf 和 Dp 是对应水果和区块的挖矿难度值。数组(R, Dp, Dpf) 决定挖矿过程。

为了阻止 ASIC 的部署，我们将使最近参数 κ = κ (t)与时间相关。VRF 将使用 VRF 每三个月生成一次新的 κ (t)(落在有效范围内)。这个过程的细节将包含在未来版本的黄皮书中。

更具体地说，挖掘算法如下。
我们尝试性地选择挖矿难度值 Dp 和 Dpf，使得挖掘水果和区块的预期时间分别为 1 秒和 10 分钟。

我们对我们的挖矿过程做出以下结论：
•水果比区块更容易被挖到，因此矿工们加入或组建矿池的积极性大大减少。实现一个更加公平的 PoW 挖矿过程
•既然水果的挖矿难度较低，那么很有可能出现两个水果同时被挖到的情况。通过选择更小的哈希值来决定哪个水果的有效性
•水果直到被写入到区块中才会稳定。因此挖矿激励将发给区块挖矿节点，该节点将对区块包含的水果挖矿节点分发激励。
•水果链的一个很大优势在于挖掘水果可以以任意顺序进行，这样使得挖矿高度一致性。当结合切片技术时，这将是非常有用的。

1 Initialize
2 chain = chain[0]
3 chain[0] = (0; 0; 0; 0; ⊥; H(0; 0; 0; 0; ⊥), ∅)
4 F = ∅
5 if heard fruit f ′ then
6 if f ′ is the unique fruit corresponding to message m′ then
7 F = F ∪ f ′
8 if f ′ .h−1 < f .h−1 for all other fruits f such that f .m = m′ . then
9 F = F ∪ f ′
10 if heard blockchain chain′ and |chain′ .F| > |chain.F| then
11 chain = chain′
12 where |chain.F| is the total number of fruit contained in chain.
13 foreach time step (1 sec) do
14 Heard signed message m, broadcasted by PBFT. Let
15 l = |chain| − 1, so chain = (chain[0], ..., chain[l]).
16 F ′ = {f ∈ F : f recent w.r.t. chain, f 6∈ chain}
17 h ′ = chain[pos].h−1 where pos = max(1, l − κ).
18 h−1 = chain[l − 1].h.
19 while mined = FALSE do
20 Randomly pick η ∈ {0, 1} κ
21 Compute h = H(h−1; h ′ ; η; d(F ′); m)
22 if [h]− κ : < Dpf then

23 f = (h−1; h ′ ; ɳ ; d(F ′); m, h)
24 F = F ∪ f
25 boardcast fruit f
26 mined = FRUIT
27 if [h]: ĸ < Dp then
28 chain[l] = ((h−1; h ′ ; ɳ ; d(F ′); m, h), F ′)
29 broadcast blockchain chain 30 mined = BLOCK

3.7 word state 和 snapshotting。 Word state 是在后台运行的数据库，以 Merkle 树的形式，提供地址和帐户状态之间的映射。作为不可变数据结构，它允许通过相应地改变根哈希来重建任何先前状态。
帐户状态包含以下信息。
（1）Nonce：此帐户进行的交易或合同创建数量。 该数字的值相对于块高度不减小。
（2）Balance：非负值，表示帐户持有的令牌数量。
（3）CodeHash：它存储合同代码的哈希摘要，如果此地址接收到消息调用，将执行该字段摘要。此字段严格不可变，构建后不应更改。
（4）StorageRoot：与每个帐户关联的存储的根哈希
3.8 交易和执行。该事务是由初链客户端的用户发起的加密签名消息。该消息采用以下形式：
(1)AccountNonce：发件人发送的事务数。
(2)GasPrice：此交易的每个单元燃料需要支付的 Wei 的数量。
(3)GasLimit：此交易应使用最大燃料量。
(4) Recipient：此交易的受益人地址。
(5)Payload：要处理的令牌数量。
(6)Code：智能合约部署的虚拟机代码。
(7)Data：混合使用。
(8)V，R，S：与事务签名对应的加密值，用于确定事务的发送者。
交易将通过以下步骤完成最终目标：
（1）交易以及一组其他交易将在 txpool 中累积。
（2）PBFT 委员会的领导者选择它认为有效的 txpool 子集，并向其他委员会成员提出建议。
（3）委员会成员检查领导提出的所有交易的有效性，如果他们同意，则签署并向其他委员会成员报告。
（4）在听取超过 2/3 的赞成票时，形成了共识。 然后将该事务作为消息广播到 PoW 节点。
（5）PoW 矿工首先验证块的事务，该块成为块头的一部分，并将它们打包成果实。果子可以按任意顺序开采。
（6）当下一个块被挖掘时，块矿工将最大的连续集（根据快速块中的水果序列号）打包到达到最终性的块。
（7）我们通过与具有最高水果难度总和的分支对齐来解决蜗牛链叉。 具有相同难度和的两个分支的概率可以忽略不计。
4.共识细节
4.1 对手模型。 在上一节中，我们介绍了基于 Minerva 共识模型的混合区块链架构。我们掩盖的一个重要观点是我们的快速链中的 PBFT 的安全性。如 3.2 节所述，设计的经典 PBFT 在允许的环境情况下运行。 拜占庭假设不到 1/3 的参与节点是腐败的，这是非常保守的。 然而，在无权限的环境中，这意味着我们需要保持快速链质量 Q fast 大于 2/3，以便链条保持一致性和活力。 在本节中，我们将进行讨论

详细介绍一下 PBFT 委员会的形成和安全沟通，观点变化以及保持 Q 值快速超过该阈值的保护措施。

我们假设以下关于我们的操作环境，我们将其称为对手模型。

（1）让 Q snail 成为诚实挖掘节点控制的散列功率的比例。 我们假设 Q 蜗牛> 2/3，这比安全性假设略微保守

由比特币和以太坊等领先项目制作，实际上他们只假设 Q 蜗牛> 1/2。

（2）Weassume 我们的对手是温和的适应性。 通过自适应，我们的意思是一个节点在被选入委员会之前可以保持诚实，并且一旦被选举就突然变成对抗性。 在温和的自适应模型中，我们不假设这个过程是瞬时的，而是在一段时间内发生，这也是我们的假设。

（3）在选择 τ 快链时间步之后精确损坏的节点称为 τ -敏捷对手。 我们不对 τ 应该采用的值或分布做出假设。

（4）BFT 委员会成员是从 PoW 节点中选出的，具有精英，尽管是无权限的过程。 我们不假设节点本身拥有先进的网络安全措施和诚实节点（例如，受委托成员）在被外部方攻击时可能会变成对抗性的。

4.2DailyBFT 协议。

4.2.1 每日 offchain 共识协议。 在快速链中，委员会成员运行一个 Offchain DailyBFT 实例来决定每日日志，而非成员则计算委员会成员的签名。 它将安全性扩展到委员会非成员和后期新的节点。随身携带它是一个终止协议，要求所有诚实节点在终止时同意最终日志。 在 DailyBFT 中，委员会成员输出签名的每日日志哈希值，然后由混合共识协议消费。 这些签名的日常日志哈希满足完整性和不可伪造性。

在 keygen 上，将公钥添加到密钥列表中。 在接收到公共信号时，产生节点作为委员会成员的选举条件。环境将有选择地对委员会开放。

以下是当节点是 BFT 成员时子协议如何工作的情况：一个 BRF 节点将被分叉。 这里的 BFT 虚拟节点，由 BFT p k 表示，然后开始接收 TX（事务）。 如果停止信号已被至少三分之一的初始公共不同公钥签名，则进行日志检查并停止日志写入。 在此过程中，会发生连续的“直到完成”检查，并且在每个步骤全部完成后，将删除所有停止日志条目

以下是子协议在节点不是 BFT 成员时的工作原理：交易的异常，消息被添加到历史记录中，并由初始通信的三分之一公共密钥签名。

签名算法使用前缀 0 标记内部 BFT 实例的每个消息，并使用前缀 1 标记外部 DailyBFT 的每个消息以避免命名空间冲突。

4.2.2 mempool 子协议。 使用 0 初始化 TX 并使用 Union 集跟踪传入事务。 在接收建议呼叫时，它将事务添加到日志并与八卦协议通信。 它还支持查询方法以返回已确认的事务。 通过跟踪集合中的事务，它清除已经确认的事务。

4.2.3。 新生成的节点。 具有隐式消息路由的新生成节点，其携带发送和接收的转录的历史记录。这与以下组件相互作用 - Mempools，Snailchain，Preprocess，Daily Offchain Consensus 和链验证。

4.3。 混合委员会选举。 在，BFT 委员会实例在一段固定的时间后切换（以蜗牛链作为逻辑时钟）。我们的蜗牛链预计每 10 分钟产生一个块，所以我们设置了 144 个块的旋转频率。一个新的委员会由 snailchain 内最新的 csize 数量的矿工组成。

一个简单的实现将容易受到众所周知的自私挖掘攻击策略的影响。自私挖掘的破坏在混合共识设置中被放大，因为权力更集中在前几个高哈希节点中。 如果一个自私的矿工控制了超过区块链哈希能力的 25％，那么她可以控制超过 33％的区块产量。 根据自私矿工所描述的选举程序，可能控制着 BFT 委员会的 1/3。 如果她碰巧是对抗性的，那么快餐链就会失去活力。

我们需要找到两个不受欢迎的极端，
通过 VRF 随机选择 BFT 成员。 这对于 sybil 攻击是易受攻击的。任何选择程序，其中被选择的概率与散列功率成比例。 BFT 委员会容易受到富含哈希权的采矿池的影响。
我们提出的解决方案如下。 当一个诚实的 BFT 节点的链长度达到 λ 时，它将发布链中每个水果的唯一矿工 ID 作为候选者（或者每个挖掘超过 v 果实的矿工 ID）。 新的 BFT 委员会是从具有统一概率的候选人中随机选出的。 由于您需要最低级别的 PoW 才能成为候选者，因此在此方案下无法进行明显的 sybil 攻击。 对于大型矿业池来说，要实现 BFT 委员会的
4.4。 安全的沟通渠道。 在委员会节点之间建立安全通信通道的能力对于构建分散，高性能和安全网络的成功至关重要。 在公众中存在一种常见的误解，即比特币和以太坊等传统的 PoW networks 因为挖掘难题而难以解决。 相反，挖掘难题只不过是一种延长时间机制，因此节点有足够的时间来同步，因为网络延迟。妥协，这两者都不容易。 水果比矿块容易开采的数量级。
更糟糕的是，实现所有节点同步所需的聚合网络流量相对于网络中的节点数以多项式顺序增长。 随着网络的增长，其吞吐量将迅速衰减，使其生态系统的发展变得不可持续。 基于 PBFT 的区块链实现比 PoW 更好的每秒事务的原因是，在事务上达成共识所需的节点要少得多（通常为 10-30），并且无论网络中的节点总数如何都保持不变 虽然我们失去了一定的去中心性。
Minerva 旨在通过使用轮值 BFT 委员会来处理交易来实现高吞吐量，并通过 PoW 来确定谁被选入委员会。 有一个问题，BFT 怎么做
委员会成员就彼此的投票决定进行沟通？ 如果 BFT 委员会使用 PoW 网络的 gossip 协议，则在委员会成员听到之前，投票消息将需要遍历整个 PoW 网络。 该网络的实际实验部署导致吞吐量为 80 tps，比特币现金略有改善。
另一方面，BFT 委员会成员可以向公众宣传他们的沟通渠道，在投票过程中建立直接的点对点沟通。 但是，通过 4.1 节中我们的对手模型，我们可以期望这个节点是 DDoS，快链效果将迅速恶化。 因此，我们需要解决的核心问题是，如何保护轮值委员会成员的身份，同时实现直接的点对点沟通
考虑以下协议，
（1）如果在最后 144 个蜗牛链块上开采超过 v = 100 的 PoW 节点并且愿意参与 PBFT，则它自动成为候选委员会的成员。
（2）从节点 i 挖掘的果实中得到的哈希数据用于区间 I（i）⊂[0,1]。另一个随机数 Γ 也生成来自 snailchain 历史哈希数据（足够远，所以所有节点都有同步）。 如果 Γ ∈I（i），则节点 i 已被选入委员会。 所有诚实节点的 Γ 值应该相同才能达成共识。
（3）重复（2）直到产生足够的委员会成员。
（4）设 Γ 1，...，Γ csize 为生成的均匀随机数。 选定的节点可以从 snailchain 数据本地计算，下一个 com 的其他成员的地址
mittee。 假设这些是 1，...，添加 csize。
（5）假设节点添加 i 是新的委员会成员。 她将广播她的 IP 地址（或建立直接通信渠道所需的其他数据），使用 add 1 的公钥加密，...，使用 gossip 协议添加 csize。
（6）当另一个委员会成员添加 j 听到此消息时，她将使用她的私钥对其进行解密，并找到 csize -1 行乱码和一行包含节点添加 i 的 IP 地址，而非委员会成员将无法 解密这个信息。
该协议允许每个委员会成员在本地构建其他委员会成员的 IP 地址表，这将允许高吞吐量通信，同时从公众隐藏其 IP 地址。 这个程序的一个小缺点如果对手恰好是委员会成员，她可以向公众广播整个委员会 IP 地址列表。 所有委员会节点都可以在短时间内进行 DDoS，并且违反了轻度自适应对手假设。

因此，我们进行以下调整。假设委员会中有 csize 成员，我们创建一个私人 gossip 网络，每个节点连接到 gsize 其他节点。 这些参数的典型值可以是 csize = 31 和 gsize = 4。
（1）我们从历史哈希数据生成矩阵 A，使得 A i，j∈{0,1}，以及 A 的列和行总和达到大小。矩阵 A 应该在所有诚实节点上相同以达成共识。
（2）（2）我们给由随机数 Γ i 选出的委员会成员提供标签 Γ i .
（3）（3）委员会成员我将仅使用 j 的公钥加密她的 IP 地址，如果 A ij = 1 则提供。
我们刚刚建立了 csize 个私人节点网络，每个节点记录 gsize 节点。 如果对抗性节点进入 BFT 委员会，最大的损害是 BFT 委员会失去 gsize + 1 节点。 设置大的 gsize 可以提高区块链的吞吐量，而较小的 gsize 可以提高链的安全性。 通过实验我们发现设置 gsize <16 csize 通常可以提供良好的平衡。这使我们能够在测试网络实时部署设置中实现每秒超过 2000 次的事务处理。
如果 gsize 太小，我们可能冒着进行 eclipse 攻击的风险，特别是当伴随着 A 的错误选择时。粗略地说，如果 A 非常接近低维矩阵的直接和，则得到的图将开始 具有隔离区域，使一些节点成为比其他节点更重要的连接器。 一致地生成 A 的鲁棒选择的算法依赖于对称群的表示理论，这将在即将发表的论文中解释。

4.5 特定应用设计。 我们的共识设计在不影响一致性，活性和安全性的条件下，了解应用程序的特定要求和定制。
4.5.1 物理时序限制。 默认情况下，传统的共识设计允许矿工/委员会成员/领导者在一个小的时间窗口内重新订购交易。 这给一些分散的应用程序带来了问题，例如商业交易所，其中交易公平性要求交易之间的时间顺序要谨慎保留，否则恶意（或甚至是正常的理性）参与者将有动力重新订购交易，甚至 插入自己的交易，以获得额外的利润。 这种激励措施将在高吞吐量下放大。
更糟糕的是，这种恶意重新排序是无法区分的，因为网络延迟自然会导致重新排序，这种延迟只能由接收者本身观察到，因此它有关于网络延迟的数字的最终证据。
为了支持分散式广告交换，我们尝试通过引入另一个称为粘性时间戳的限制来减少此类问题。更具体地说，使用启发式参数 T Δ，在提议事务时，我们要求客户端在元数据中放置物理时间戳 T p 事务的这个物理时间戳与事务的其他部分一起签名。 稍后当 BFT 内的验证器验证事务时，它将执行以下额外检查，如算法中所示。
在 BFT 内部实现日志的阶段，领导者将根据其物理时间戳对事务批处理进行排序，并断开与序列号的关系（尽管非常不可能）。 实际上这个步骤不是必需的，因为我们可以在评估和验证的后期强制执行订单。 但为简单起见，我们把它放在这里。
这套修改给了我们额外的几个特性：
（1）来自任何节点 N i 的事务的顺序根据其物理时间戳在内部保留。 因此，严格执行这些交易的顺序。 这将消除一些涉及来自同一节点的两次转录的恶意重新排序的可能性。
（2）BFT 委员会输出的一批交易中的订单严格按时间戳排序。
（3）由于时间窗口的限制，节点无法操纵虚假的物理时间戳。
这种修改的一个明显缺点是，当参数 T Δ 不适合变化的网络延迟时，由于中止事务导致的吞吐量减少。另一个缺点是，BFT 委员会成员仍被允许撒谎当地时间并拒绝 某些交易。 但是，委员会成员无论如何都可以拒绝某些交易。 但诚实的节点可能会因为时钟不同步而拒绝无知的交易。 通过增加对 BFT 委员会资格的限制可以减少这个问题。 稍后我们将看到，为了进入委员会，节点应该提供同步时钟的证据。
5.计算和数据的硬化和投机交易执行
在本节中，我们将介绍我们的分片方案。对原始混合共识的一个重要修改是我们为它添加了

计算和数据分片支持。甚至更多，
首先，我们设计了一个基于分片的推测交易处理系统。这个想法很清楚。在混合共识中，DailyBFT 实例被索引为确定的序列 DailyBFT [1 ... R]。我们允许多个 DailyBFT 实例序列同时存在。确切地说，我们用碎片 S t 表示第 t 个 DailyBFT 序列。为简单起见，我们将分片数量固定为 C.每个 DailyBFT 是一个普通分片。除了 C 普通分片外，我们还有一个由 csize 节点组成的主分片 S。主分片的工作是完成普通分片输出的排序以及在分布式事务处理系统中实现协调器。而普通的分片，而不是直接与混合共识连接，将日志提交到主要 shard，然后以此告诉混合共识。
我们不允许任何两个分片（正常或主分片）共享公共节点，这可以在委员会选择过程中强制执行。 多个分片的选择类似于章节中描述的选举程序。
我们将状态数据（在帐户范围方面）统一划分为 C 分片。 这将确保对相应分片的每个查询都将返回一致状态。 由于我们要为每个数据单元包含元数据，我们将数据分成数据扇区单元，并为每个数据扇区分配一个地址。 我们有从数据位置到数据扇区地址的映射。 为简单起见，从现在开始，我们只讨论数据扇区。 每个数据扇区 DS [addr]都有 rts，wts，reader，writer 等元数据。
我们假设分区原则是公共的，并且给定地址 addr，我们可以通过调用函数 host（addr）来获取它的主机分片。

Algorithm 2: Extra Verification Regarding Physcal Timestamp

```
Data: Input Transaction TX
Result: A Boolean value that indicates whether the verification is passed
current_time ← Time.Now();
if |current_time − TX.T_p| > T_Δ then
    return false;
    // if the time skew is too large, reject TX.
var txn_history = new static dictionary of lists;
if txn_history[TX.from] == NULL then
    txn_history[TX.from] == [TX];
else
    if txn_history[TX.from][−1].T_p − TX.T_p > 0 then
        return false;
        // To make sure the transactions from the same node preserve timing order.
    else
        txn_history[TX.from].append(TX);
        return true;
```

图 1.用于额外验证的伪代码
请注意，如果我们将每个正常分片（当攻击者的数量不大时）视为分布式处理单元，我们可以在分布式事务处理系统中加入逻辑时间戳的设计，这将使得处理交易.这里我们使用 MaaT 的简化版本，我们不会自动调整其他事务的时间戳。
对于普通分片，它的作用与 DailyBFT 中描述的完全相同，只是以下更改使其兼容并行推测执行。
对于主分片，它从所有普通分片收集输出。 请注意，事务的数据依赖性可以通过其元数据轻松推断。 事实是，如果一个事务访问多个远程分片，它将在所涉及的所有分片中留下痕迹。 当正常的分片提交日志到主分片时，它也会写入慢链。
当主分片从分片中接收（或从 snailchain 中获取）一批 txns 时，它将检查是否已从该批中的所有分片事务中接收到该分片。 如果在某个超时后它没有收到来自特定批次的交易，则表示批次失败。 在这种情况下，整个委员会的开关将在第二天的起点触发。 在收到所有分片的日志后，主分片根据其提交时间戳对事务进行排序（如果某个事务具有较早的批号，则它

将被视为排序中的第一个键，但是，如果其物理时间戳违反了时间戳，则 manyshards，我们认为批次无效，并且该批次内的所有交易都被中止）。 排序后，主分片会过滤所有事务，并在物理时间戳方面保持最长的非减少序列。 将日志记录到混合共识组件中作为当天的日志。
这里仍有许多优化空间。 一个确定的问题是，这种设计中的确认时间不是即时的。

6 运行在虚拟机上的智能合约

6.1 设计原理阐述

在拥有 Ethereum 虚拟机(EVM)[32]的所有原因中，目标之一就是在 POW 共识模型中使用交易费来计量使用量。由于我们初链是一个混合共识模型，我们将更深入地探索这个领域。我们会考虑一下混合云生态系统的可能性。
人们对以太坊黄皮书[32]遇到的一个基本问题是里面的数学符号。因此，我们希望能遵循像 KEVM 黄皮书[18]的做法来列出 EVM 和 TVM(在 4.2 中描述)规范。将来，我们希望通过初链的 github 帐户(https://github.com/truechain)来维护我们自己的规范。

6.1.1 如果将虚拟机替换成容器会怎样

区块链架构中最接近这个概念的是 Hyperledger 的 Fabric 框架[9]。如果尝试着将 fabric 的有准入许可特性转化为无准入许可，最主要的挑战之一将是解决链码问题。这意味着，虽然可以将链码或智能合约保存在一个容器中，但这对公链来说不是一个可扩展的模型。一个公链拥有这样的模型意味着必须在单个节点上运行数千个容器，对应运行几千个智能合约(因为每个节点都维护一个副本)。
社区已经有人尝试限制运行在一个节点上面的最多容器数。正如 Kubernetes 容器编排平台[5]和 Red Hat 的 Openshift 容器平台 3.9 的集群限制[7]所示，目前的极限是每个节点 100 个 pod，每个节点大约能运行 250 个容器。
即使使用最新的存储技术，例如 brick 多路复用技术[1]，容器的最大可能值(比如最大接触值)也不可能达到(至少现在的技术是)1000。在关于 kubernetes 问题的讨论中可以进一步研究这个问题，这个问题可以在 Kubernete 的 github 的 issues 页面[4]上看到关于负载限额决定了一个 pod 上可运行的最大容器数（MAX_CONTR）的更深入的讨论。希望扩展容器的人通常偏向于水平扩展而不是垂直扩展[2,6]，因为后者显著增加了设计决策的复杂度。对于集群规模化配置来说，没有一种通用的规则，因为它完全依赖于工作负载，在我们的例子中，由于它的去中心化，它更依赖于工作负载，对于向扩展工作负载迈出一步来说，并不是很有说服力。此时，它更像是一个创新性问题，而不是简单的技术规范研究问题。目前以太坊已经部署了超过 1000 个智能合约。因此这已经变成优化容生态的设计问题了。
现在让我们扩展一下容器的场景。考虑到上述危机，一种可能的解决方案是在无服务器架构中使用容器。但是考虑一个场景，当 2000 个智能合约是同时在线的，并且并发请求，即，每次调用链码(一个移动窗口)超过 MAX_CONTR 的值时，我们将再次面临相同的问题。因此，只能建议在最大并发请求上增加节流率限制。通过设计，这严重限制了每秒钟的并发交易量。工程不应凌驾于可实现的目标之上。因此，我们选择坚持 EVM 设计，尽管为了我们的目的稍微做了一些修改。

6.2 初链虚拟机(TVM)

在这个领域的一个典型案例是以太虚拟机(EVM)[32]，它试图遵循完全确定，并且尽可能简单，以使激励成为计算的一个简单步骤。它还支持各种特性，如内存的堆栈外存储、合约委托和中间调用值存储。
我们将为慢链复用 EVM 规范，但是在本黄皮书的下一个版本中，在仔细考虑类似于 EVM 的设计原理之后，我们为 TVM 添加了一个新的规范，通过使用 kecak -256 哈希算法和椭圆曲

线加密技术(ECC)派生出基于堆栈的架构。
初链基础架构将整合 EVM 和类似 EVM 字节码执行引擎来运行智能合约。我们仅在快链上运行虚拟机，集成在全节点中，因此它们可以处理按需调用。
TVM 虚拟机基于 DailyBFT 公链技术，与以下组件交互:
复用一些 tendermint 的概念，比如 ABCI(区块链应用编程接口)，其提供了一个抽象层，允许在一个进程中运行的共识引擎管理另一个进程的应行状态；
- 适合 dailyBFT 的另外一个共识引擎。
- 权限化的以太坊虚拟机。
- 保证交易达成的 RPC 网关。

7 激励设计和 GAS 费用
POW 工作量证明协议有一个已被证实的事实，以前所未有的速度吸引计算资源。虽然比特币和以太坊等现有的基于 PoW 共识网络本身就很成功，但它们所吸引的计算资源不过是非常强大的哈希计算器。它们运行起来需要耗费大量的电力，而且没有产生任何有用的东西。在本节中，我们将介绍奖励基础设施的概念，以便平衡 BFT 委员会成员和非委员会成员全节点的工作量。我们基于 PoW 发明了一种新的激励设计，在这里，参与的资源可以被引导到做有用的事情，比如每秒处理的交易(这力称为“TPS”)，并提供链路数据存储。
以太坊 gas 价格是由一个不可能套利的现货市场决定的，类似于[30]研究的电力现货市场。我们认为这个市场是不完整的，因此资产定价的基本定理不适用[14]。因此，潜在的 gas 价格将遵循“喷射噪声”过程，即众所周知的高波动性。我们引入了一个“gas 市场”，在这个 gas 将作为期货进行交易，这个市场是在极小的极限内完成的。与以太坊相比,这预计将显著降低 gas 价格波动。
下面的小节将详细讨论激励设计的每个部分。
7.1 抵抗 ASIC 矿机
初链的挖矿算法是从根本上抵抗 ASIC 矿机的。我们将该功能定义为 Truehash，并在后文中阐述 Truehash 的意义以及解释为什么 Truehash 能够从根本上抵抗 ASIC 矿机。在这里，我们简要概括了目前已经完成的工作。虽然 ASIC 矿机在哈希计算上比通用电脑有明显优势，但是它们只能执行单个程序，是不可编程的。因此，如果有一个自动化程序可以每隔几个月（12000 慢链区块时间）更换一次挖矿算法，那么这种策略就使得制造矿机无法获益。
我们想出如下条件来从根本上抵抗 ASIC 矿机：
1. 存在一个庞大的挖矿算法池（比如 2048！），该挖矿算法池选用对 ASIC 芯片难以硬编码的算法。
2. 挖矿算法具有不需要人干预就可以自动切换的能力，就像一个硬分叉。
3. 新的挖矿算法的正确性必须是可验证的（这样所有节点才能达成共识），并且挖矿过程必须是不可预测的（这样它就不会在切换前被 ASIC 矿工窃取）
经典的 POW 挖矿是重复计算 v(nonce)的哈希值，v(nonce)是一个接受挖矿随机数 nonce、区块头并将其填充到正确的维度的函数。我们使用更新的哈希函数来替换传统挖矿算法：hash(ρ (g) ∗ v(nonce))。这里，G 是一个大的群（例如，S2048）并且 $g \in G$，$\rho : G \to V$ 是一个从 G 到向量空间 V 的同态，因此矩阵乘法 ρ (g) ∗ v(nonce)在向量空间 V 中也是有效的。
可见，通过对每一个不同的参数 $g' \in G$，替换 ρ (g)为 ρ (g)，我们得到了完全不同的哈希算法。由于 G = S2048 是一个大的群，|G | = 2048!，潜在的哈希算法空间有 2048！个，我们可以从中进行选择。
最终，我们可以通过群表示理论的应用从慢链的哈希数据中产生新的群元素 g'。因此，新的挖矿算法是可验证的并且是不可预测的。

7.2 gas 价格和分片

gas 价格在期货市场上交易，期货合约以智能合同的形式出现。具体来说，合同将按以下方式执行。

甲方同意支付给乙方 x 个 True ，乙方承诺在 T0 - T1 期间执行甲方的智能合同，运行费用为 1 个 gas。

乙方将向执行甲方智能合约的委员会 C 相应的资金池提供 x 个 True。这被称为 gas 池。

委员会 C 中的所有成员将平均获得 gas 池中的份额,并返回在池中平均每 gas 的消耗μ。

如果 B 支付的小于μ,她必须通过支付超过μ的第三方来弥补差额。如果 B 支付的超过μ,她将从第三方收到差额。

在这种设计机制下，当流动性提供者正确地预测网络压力，从而在极小的限度内创建一个完整的市场时，他们就会得到奖励。gas 池中的平均机制吸收了价格波动，使价格本身成为网络压力的良好指标。

我们确保 gas 价格在预定区间内交易的意图。因此，如果动态平均价格维持在一定的阈值以上，一个新的 PBFT 委员会将通过量子观测过程产生。另一方面，如果动态平均价格维持在某一阈值以下，则现有的 PBFT 委员会在完成其任期后将不会被授予委员会资格。

挖矿奖励的比例分配如下:设置 n 为 PBFT 委员会在一个特定的场景下运行的节点数量,同时设置 α > 1。挖矿奖励的比例的分配，其中 PBFT 节点= n/(α + n),POW 节点= α /(α + n)。这种分配机制的目的为了在初链发展的后期，新的节点被激励为区块链总的 TPS 做出贡献，从而确保链的可扩展性。当挖矿奖励被对半分时，参数 α 等于 PBFT 委员会成员的数量。

7.3 数据存储

初链的目标是在每个分片上达到 10,000 TPS，并且分片的数量被设计成相对于交易量的需求线性增长(大致与节点数量相关)。在 10 年的时间里，比特币网络在 10 年时间里产生了大约 170 gb 的交易历史数据。从 10 到 10000，相同的数据量预计每 3 天生成一次。在 100 个切分，或 100 万 TPS，我们可以期望它每 45 分钟产生一次。因此，在一个高 TPS 的公共链中，让每个节点存储像比特币这样的整个交易历史就不再可行了。

已经提出了许多解决方案，例如每隔几小时/天检查一次，其中每个节点只需要存储来自前 n 个检查点的交易历史记录。但是我们应该把交易历史的其余部分存储在哪里呢，因为没有人有动机去存储它。

我们的解决方案是在统一的激励基础设施中无缝地将交易处理与 IPFS 的存储能力合并。这将提供存储交易历史的解决方案，并允许在初链体系结构上运行大量复杂的应用程序。初链上的数据存储可以分为三个层次，

层次 1：存储在每个 PoW 节点上，比如比特币和以太坊。 这是最永久的存储方式，也是效率最低的方式。不建议存储短文本以外的任何内容，例如关键信息的 hash 值。用户向 PoW 矿工支付 gas 费。

层次 2：将会有一个类似 ipfs 的文件系统，其中数据的有限副本分布到整个链中的存储节点。这是为存储主网络的大量历史交易和大量数据的去中心化应用程序而设计的。用户将向矿商支付存储和检索的费用。

8 未来方向

即使对最初的混合共识机制采取优化，我们认为在本文所提出的基础上还有更多的优化空间，如：

- 改进所有节点的时间戳同步，而不需要中心化的 NTP 服务器。
- 奖励基础设施里的详细激励技术，使重度投资基础设施的投资方不会遭遇“被忽视”，“亏本”的问题。
- 支持副本创建的分片技术，尽量减少被 BFT 会员会拒绝的交易集。

- 添加零知识证明以增强隐私。
- EVM、TVM 和 Linux 容器技术的混合基础设施。
- 改进虚拟机规范中的二进制数据编码方法，交易签名，收费模型等章节。

9 结论

我们正式定义了混合共识协议和其实现方法，在本草案里，我们介绍了下一版将引用的多种新的理念。

我们建议大家在部署 POW 全节点时采用抗 ASIC 的硬件方案，关于硬件方案的更多细节将尽快给出。

在许可的基于 POW 的网络中的少数节点上运行的许可 BFT 链。

BFT 委员会是一个轮值的委员会，可以及时防止腐败。

BFT 委员会负责交易验证，而 POW 节点只负责根据我们得出和重新定义的一些规则选举委员会成员。

我们推测，新许可的虚拟机可能受 EVM 的启发，但具有不同的块状态和交易执行流程。

在 POW 链中没有权限的 EVM 与这个新的许可 VM（我们称之为 Truechain Virtual Machine - TVM）共存。

TVM 将是验证与共识有关的交易的传统方法，而传统的 EVM 需要重新进行工作才能真正达成共识致，但对于 BFT 选举来说使用变日长是一个难题。

激励模式需要重新开展工作，使其基于 TVM，仍然奖励 POW 链中的矿工。

我们最终会支持 BFT 委员会节点的分工，以提高可扩展性。

考虑到节点配置不一致性（节点池中不同的 CPU/内存/网络带宽）的奖励机制最终将成为共识的一部分，从而加速交易。

因此，智能合约执行将仅在 TVM（BFT 节点）中发生。

*感谢 TrueChain 初链技术社区的支持，将此版黄皮书翻译成中文，让更多区块链爱好者有机会了解到初链的底层技术。