混合共识：无许可模式下的有效共识

Rafael Pass和Elaine Shi

CornellTech，Cornell，CryptoCurrency和合同倡议（IC3）*

摘要

共识或状态机复制是分布式系统和现代密码学的基础构建块。 在经典，许可环境中的共识已经在分布式系统文献的30年中得到了广泛的研究。 比特币和其他分散加密货币的最新发展在“无许可”环境中普及了一种新形式的共识，任何人都可以动态加入和离开，并且没有对共识节点的先验知识。 尽管取得了令人振奋的突破，但今天的无党派共识协议（通常称为“区块链”）已知具有可怕的性能，这引起了热烈的讨论，并且在社区中引发了激烈的争论。 首先，我们表明，不幸的是，对于任何能够抵御至少三分之一破坏力的协议来说，性能损失是固有的。具体来说，我们正式定义了一个称为响应的新性能度量，并且表明任何响应式的无权限共识协议都不能容忍哈希功率的1/3或更多的损坏。 接下来，我们展示一个紧密匹配的uppper绑定。具体来说，我们提出了一个称为混合共识的新的无权限共识协议，该协议具有响应能力，并能够抵御1/3的破坏（大约）。 混合共识的想法是通过将低效区块链协议与快速许可的共识协议相结合来引导快速的无许可共识。 混合共识使用区块链不同意交易，但同意轮流委员会，轮流委员会依次执行许可的共识协议以就交易达成一致。 我们的论文是第一个正式处理如何正确结合工作区块链和传统共识以实现响应式的无权限共识协议的工作。 我们的努力揭示了许多对协议正确性至关重要的不重要的技术细节和挑战，但早期并行的临时方法忽略了这些细节和挑战。

*hTTP：//www.initc3.org

1 介绍

传统上分布式系统和密码学文献主要关注参与者先验已知的协议。 比特币迅速崛起成名，这是一个令人振奋的突破：比特币凭借经验证明，通过利用诸如工作证明等假设，非重要的安全应用程序可以建立在完全分散的网络之上，节点自由动态地加入和离开，参与者之间没有预先建立的信任。 在本文的其余部分，我们将分别将这两个网络设置称为许可设置和许可设置。

非正式地说，比特币的核心共识协议，通常被称为Nakamoto共识 [46],实现了“复制状态机”抽象，其中无权限网络中的节点就所提交的一组事务以及它们的排序达成一致。 由于协议依赖于交易块的链接，因此它通常被称为“区块链”。

在传统的许可模式中取得共识后，成为经典的分布式系统问题，并且寻求设计和优化拜占庭共识协议的研究有很长的路线 [21,25,42]. 我们可以在无权限模式中获得共识（依靠工作证明）这一事实是比特币的新贡献。 从某种意义上说，比特币推广了分布式系统的新模式，这在30年的传统分布式系统文献中很少被考虑。

已知的无许可共识协议，比如比特币的Nakamoto共识 [46],然而，需要付出代价。 由于节点的身份不是先验已知的，因此必须防御Sybil攻击，攻击者组成任意多个身份以胜过诚实节点。 比特币协议严格依赖于工作证明来粗略地实施“每票平均一票”的想法。 不幸的是，比特币的表现非常糟糕。 正如克罗曼等人。 [22] 指出，比特币网络最多可以支持7 tx / sec，交易确认时间为10分钟以上（比如Visa这样的主流支付处理器可以处理平均速率为2 000 tx / sec，峰值速率为59,000 tx / sec）。此外，如果我们在所有交易中确认网络的总电量消耗，每笔确认的交易的成本大约为1美元至6美元。如今，这笔费用在某种程度上受到比特币投机者的补贴。

这自然会引起一个重要的问题。

是否可以在无权限模型中设计高效的共识协议？

我们在本文中正式探讨这个重要问题。

1.1 我们的结果和贡献

了解限制：性能与安全性。 为了正式理解这一点，让我们先试着理解为什么Nakamoto共识协议 [46]（被比特币采用）效率低下。 正如Garay等人 [30] 和Pass等人。 [47] 指出，Nakamoto共识协议至关重要的依赖于网络延迟上限的先验知识（以下称为Δ）为
参数化其难题难度，并且协议的事务确认时间大致为$O(\lambda\Delta)$以实现$\exp(-\Omega(\lambda))$安全失败 - 一种考虑这种情况的方式是块间隔需要为$O(\Delta)$至实现对任何恒定部分腐败（哈希功率）的安全性，并且必须等待$O(\lambda)$块获得$\exp(-\Omega(\lambda))$安全失败。 Nakamoto显然效率低下，因为先验参数Δ需要保守地设置以保证安全性
协议；并且交易确认时间在估计Δ中存在松动。 虽然目前还有其他可能的效率指标，但我们将重点关注这一指标。

因此，一个自然而然的问题是，我们是否可以有一个协议，其交易确认时间仅取决于网络的实际性能，而不是任何先验已知的上限。 我们正式定义称为响应的性能指标[1] 它捕捉了这种直觉：如果协议的交易确认时间仅取决于网络的实际延迟δ，但不取决于任何先前已知的上限Δ（或者根本不知道先验的上限是已知的）。 具体而言，实际上δ实际上通常（小得多）小于上限Δ。 在这种情况下，响应能力将成为衡量绩效的有效指标。

定*理1. （非正式的）没有安全和响应性的协商一致协议可以容忍1/3或更多的腐败，即使当对手被限制为静态腐败时 - 这在经典的许可设置（即使假定为PKI）时也是如此，作为以及在工作证明（其中腐败是用散列强度表示）的无许可情况下。*

坏消息是，我们表明没有响应共识协议可以容忍三分之一或更多的腐败，即使对手受到限制，只能静态地腐败党。 这个下限结果在经典的许可设置（即使假设为PKI）以及在工作证明的无许可设置中都保持不变 - 在此设置中，损坏根据哈希功率计算而不是节点数量。 为了说明这一点，请注意Nakamoto没有反应能力，但可以容忍1/2的破坏力 [47].

这个下界的证明与Dwork等人的下界有关。 [25] 他表明，在一个异步（或部分同步）网络中，没有传统的许可共识协议（即使使用PKI）也能容忍1/3或更多的拜占庭式腐败。 在我们的论文中，我们将展示如何使他们的证明适应无许可证和工作证明的设置以及响应协议。

具有（几乎）最佳弹性的响应式协议。 下一个显而易见的问题是：假设我们愿意放松模型，并假设哈希功率只有<1/3的损坏，我们能否在无许可的情况下制定一个响应共识协议？ 我们以积极的态度回答这个问题。

定*理2. （非正式的）假设一个工作证明随机预言。 存在一个响应式的无任务的共识协议，可以抵御轻度自适应对手的三分之一破坏。*

为此，我们提出混合共识。 混合共识为无许可共识提供了“效率引导”，就像众所周知的混合加密和OT扩展是“效率引导”结构一样。 自古典的许可共识 [15,19,21,25,39,40,42,45] 已经被研究和优化了几十年，并且已经被证明能够对1/3的腐败做出反应，我们的想法是使用缓慢的区块链协议（称为snailchain），例如Nakamoto共识 [30,46] 以引导快速许可的拜占庭共识，最终的结果是在无权限模式中的可扩展共识协议。 出于这个原因，我们把我们的协议称为“混合共识”。

混合共识是第一个已知的响应式无许可共识协议，甚至是启发式的。 我们正式证明混合共识达到了针对具有以下能力的恶意（即拜占庭式）攻击者的安全性：1）总计算能力的大致或[1]部分； 2）可以自适应地破坏节点，但腐败需要一段时间才能生效（从此以后） 3

[1]请注意，响应性不会被Garay等人定义的活性所误认。 [30] 和Pass等人。 [47] 在无许可的情况下。 活跃度要求在由活性参数T指示的有限时间内确认交易；而响应性又要求T仅取决于网络的实际延迟δ，而不取决于（可能松散的）先验上限Δ。

方案	TX conf。 时间	加工/ TX	% 诚实
中本聪 [46], BitcoinNG [27]	Θ（λΔ）	上）	$\sim \frac{1}{2}$
Fruitchain [48]（同时工作）在	Θ（λΔ）	上）	$\sim \frac{1}{2}$
Nakamoto Hybrid杂交共识上的杂	Opt：O（δ），最差：O	O	$\sim \frac{3}{4}$
交共识	（λδ）	（λ	$\sim \frac{2}{3}$

表1：我们的结果总结。 n表示节点的总数（假设所有节点具有相等的散列功率）； Δ表示网络传输延迟上的预定上限； δ表示网络的实际延迟； λ是获得2的安全参数-λ
安全失败。

被称为温和适应的对手）； 和3）并且可以在传输期间重新排序消息，并且延迟消息达δ时间步长的界限。

其他实际好处。 除了提供响应能力之外，混合共识在实践中还带来了额外的好处：对于智能合同应用，它将处理成本降低到O（λ）（即，独立于矿工总数），而现有区块链如Nakamoto要求所有矿工执行智能合约。 最后，混合共识可以帮助提高共识协议的整体吞吐量，因为我们预计选举委员会（与完整的对等网络相比）的带宽吞吐量更高。 尽管混合共识不会消除工作证明，但它通过在较高的事务吞吐量上摊销工作成本证明成本来降低每个确认交易的成本。

1.2 技术细节与密切相关作品的比较

尽管在社区中已经讨论了将无许可的共识和许可共识相结合的想法（例如，Decker等人最近的工作。 [23]和并行和独立的工作ByzCoin [37]),据*我们所知，以前的工作没有提供正式的处理。*

技术细节。 正如我们的工作所表明的那样，在构建和证明安全性方面，将许可和无权限的协议相结合并不是微不足道的。 我们的努力揭示了许多微妙之处，如果没有正式的对待就很容易被忽视：例如，如何防范自私的采矿并实现最佳的韧性； 如何依靠链上加盖技术来防止追溯性的委员会腐败行为； 如何处理委员会的对抗性选择性开放； 如何正确处理委员会切换，以确保在切换期间可以在短窗口上同时运行多个传统BFT实例时确保组合安全。

与密切相关的作品比较。 相比之下，Decker等人的先前工作 [23]和同时工作的Byzcoin [37]采取一种启发式的方法来实现安全：经过仔细研究，这些作品似乎忽略了上述一个或多个微妙之处（以及其他细节）。 首先，两者都有效 [23,37]声称获得最佳韧性（即容忍1/3腐败），但他们的说法是不正确的 - 因为这些作品使用Nakamoto作为潜在的snailchain，由于众所周知的自私挖矿攻击 [28],他们最多可以容忍1/4瑕疵缺席不平凡

改变他们的协议。 除了这个值得注意的问题外，Byzcoin协议还有其他明显的漏洞 [37]: 例如，他们的协议不会删除委员会选举中的尾随块，因此可能不会就委员会达成一致意见 – 这可能会损害其协议的一致性和/或生活性（其他人也独立地观察到了这些Byzcoin的脆弱性 [1]）。 事实上，自从拜茨币 [37] 发表在Usenix Security'16上，作者自己已经在后续的博客文章中承认了其中的一些漏洞 [3,4]. 值得注意的是，在他们最近的博客中 [4], 他们引用我们的论文作为实现最佳弹性的潜在解决方案 – 但是，迄今为止，他们仍然没有提供正式的正式协议。 由于他们的工作似乎忽略了我们的努力所揭示的多个微妙之处，因此不清楚他们提供的安全保证或他们的协议提供任何保证的安全模型。

因此，我们的工作非常独特，因为我们提供了第一个证明工作区块链和经典共识以实现响应能力的正确方法。

1.3 纸张组织

我们的正式待遇是一个相当重要的努力（并且如上所述是必要的） – 因此为了可读性，我们首先在Section中提出半正式的技术路线图 2 在我们给出正式的定义，协议描述和证明之前。

2 技术路线图

2.1 预赛和积木

在这一节中，我们提出半正式或非正式的定义，以获得无许可的共识，这是我们旨在实现的抽象，以及我们依赖的构建块。 正式的定义将在后面的文章中介绍。

2.1.1 许可协议

我们的目标是在无权限模型中实现状态机复制抽象 – 从此称为无许可共识[2].

在一个无许可协议的协议中，每个节点在每个时间步输出一个LOG--这个LOG表示一组已提交的事务。 必须以绝对的可能性来保证两个重要的安全要求，即一致性和活跃性。

- 一致性：一致性包括以下内容：
  - *通用前缀*。 假设一个诚实节点i在t时刻输出LOG给Z，而一个诚实节点 j（相同或不同）在t时刻输出$LOG^{t'}$给Z'，它认为$LOG \prec LOG^{t'}$或$LOG^{t'} \prec LOG$。 这里的关系≺表示“是前缀”。 按照惯例，我们假设对于任何x，x ≺ x和x ⊀ x。
  - *自洽*。 假设节点i在时间t和$t' \geqslant t$是诚实的，并输出LOG和$LOG^{t'}$分别在时间t和t'，它认为$LOG \prec LOG^{t'}$。

[2]经典分布式系统文献中也使用术语共识来表示单次协议协议。 在本文中，共识意味着状态机复制，其中节点想要就随时间增长而线性排序的日志达成一致。

- *生存力*：假设事务TX在$t \geq T$时输入到一个诚实的节点$_{暖身}$。 那么，如果在时间$t' \geqslant t+ T_{确认}$处诚实的任何节点在时间t'输出LOG，则认为TXs⩾LOG。

直观地说，活跃性表示事务被包含在$T_{确认}$时间内的诚实节点的LOG中。 有两个活性参数$T_{确认}$和$T_{暖身}$。$T_{暖身}$是协议的预热时间；和$T_{确认}$是交易（在$T_{暖身}$之后提出的）最大等待时间待确认。

2.1.2 Blockchains

混合共识依赖于工作区块链（以下称为snailchain）作为构建块。 工作区块链可以被看作是一个无权协议的特殊情况 – 事实上，我们将在本文的完整版本中正式展示这一点。 Garay等人提出了区块链的正式抽象。 [30] 和Pass等人。 [47]. 我们在本节中提供了区块链的半正式定义，同时将正式定义推迟到后面的章节。 准确地说，一颗螺旋链满足以下特性。

- 一致性。 所有诚实节点的链（任何时候）都一致，除了尾随 λ块，其中λ是安全参数。 此外，节点链与其未来的自我一致。
- *链*质量。 在一个诚实节点链中的任何连续的λ区块中，足够的一小部分区块是由诚实的矿工开采的。
- *逆*锁增长。 诚实的节点链条以稳定的速度增长，既不快也不慢。

2.1.3 许可BFT

在经典的分布式系统文献中已经对许可的BFT协议进行了广泛的研究。 通常，已知的许可BFT [21, 25,42] 完全符合前面定义的一致性和活性保证 – 但这里是为了获得许可的设置。

混合共识将作为构建模块使用许可的BFT协议。 事实证明，在我们的上下文中，一个有趣的技术微妙出现在正式化被许可的BFT所需的抽象中。 基于财产的安全定义（正如通常在分布式系统文献中所做的那样）由于选择性开放式攻击而变得不足。 特别是，攻击者可以先看看节点的公钥，然后自适应地影响委员会的选择方式。 在Section中 4.2, 我们认为存在一个（有点人为的）许可的BFT协议，在基于财产的定义下证明是安全的，但如果遭受敌对选择性开放，则会被完全破坏。

因此，我们为我们基础许可的BFT构件定义了一个加强的安全性概念。 我们不仅要求上述一致性和活性属性以压倒性的概率得到满足，还需要以下更强有力的说明：

存在ppt减少B，使得给定任何ppt对手可以破坏BFT的任何公钥集合的安全属性，使得黑盒调用该对手的缩减B可以代表诚实方伪造签名。

在本文后面，我们将形式化上述概念，并通过增加PBFT算法来展示如何实现这种抽象 [21].

2.1.4 执行模型

我们考虑一个开放的招生网络，其中包含一个工作证明的预言模型，与之前的作品一样，模型是一个随机预言 [30, 47].

网络假设。 我们假设一个部分同步模型，其中由诚实节点发送的任何消息被保证在δ时间步骤内到达所有诚实节点。 允许对手按照上述限制对消息进行重新排序。

我们的协议需要知道δ的一个可能松散的上界来参数化该方案（特别是，参数化底层snailchain的难题）。 我们此后使用符号Δ来表示这个预先确定的上限。 我们的协议实现了响应性：即使我们使用先验上限Δ作为输入参数，我们的协议实现的交易确认时间取决于网络的实际延迟δ，而不是可能松散的上限Δ。 这个要求使得我们的设置与同步模型有着根本的不同 - 因为如果协议简单地将Δ时间步骤作为同步轮，则协议将不会响应。

轻度适应性腐败。 尽管我们允许攻击者自适应地决定哪些节点损坏，但腐败并不会立即发生。在我们的模型中，当攻击者向节点发出"目标损坏"指令时，节点实际上会损坏τ（τ将在后面的完整版本中进行参数化）需要τ时间。 一旦节点实际上变得腐败，攻击者可以杀死该节点。 最后，新节点可以随时产生。

允许的参数。 此后，我们将假设底层区块链选择合适的难度参数，使得预期的区块间时间是比真正节点之间的网络延迟大的合适的不变部分。 本文后面我们将这些可接受性要求正式化。

2.2 混合共识协议：概述

底层的snailchain。 我们构建一个潜在的区块链协议（表示为snailchain），其正式抽象由Garay等人定义。 [30] 和Pass等人。 [47]. 这种蜗牛抽象有两种可能的实现方式，即原始的Nakamoto consenus [30,46,47], 和最近的Fruitchain [48] 协议。 为了概念上的简单，在我们的展览中，我们会
假设Nakamoto是潜在的snailchain。 然而，正如我们稍后解释的那样，将Nakamoto作为snailchain使我们能够抵御1/4 - E腐败（在hashpower中）。
Fruitchain [48] 实现与Nakamoto snailchain相同的抽象 - 然而，作为Fruitchain
可以证实自私的采矿攻击，它实现（几乎）理想的连锁质量。 正如我们后来所说的，如果我们改为采用Fruitchain作为潜在的snailchain的替代品，我们可以针对任意小的常数E来抵御1/3 - E腐败（在hashpower中），因此
实现（几乎）最佳的弹性。 顺便说一下，这也显示出引人注目的优势
模块化协议设计和组成。

2.2.1 热身：静态委员会

从根本上说，诸如Nakamoto共识（以下称为snailchain）的区块链依赖于工作证明的难题，使得节点可以建立Sybil弹性身份。 我们的第一个想法是利用蜗牛来选举一个静态委员会。 为此，诚实的节点运行区块链的csize +λ区块，其中csize =Θ（λ）表示目标委员会的规模，λ表示安全参数。 此时，一个诚实的节点将从其本地链中移除尾随的，不稳定的λ区块，并呼吁第一个大小的矿工阻止BFT委员会[3].

[3]如果同一矿工开采多个块，该矿工可以简单地充当BFT协议中的多个虚拟节点 - 因此，该协议也适用于n <λ（我们的最终方案混合共识也是如此）。

粗略地说，这样的协议在静态腐败模型下可以证明是安全的，这是由于以下原因。

- 由于snailchain的一致性，所有诚实的节点在同一个BFT委员会上达成一致。 我们强调重要的是去掉尾随的λ不稳定块，否则就是诚实的
  节点对于谁应该是BFT委员会会有不同的意见（例如，由于snailchain中可能存在叉） - 在这种情况下，我们无法保证协议安全[4].
- 由于螺旋链的链条质量属性，具有适当的整体参数，我们可以确保2/3以上的委员会成员诚实，足以保证
  许可的BFT协议的安全性。
- 由于螺旋链的链条增长性质，BFT委员会形成的时间不会太长。

最后，委员会成员签署所交易的任何交易以及其序列号。 对于任何未被选为委员会成员的节点，它可以简单地计数委员会成员的签名数量以决定其自己的输出日志。 由于超过三分之二的委员会成员诚实，至少有一位诚实的委员会成员，如果至少有一位委员会成员担保同一交易及其序号（即其在输出内的相对位置
日志）。 不难看出，假设所采用的BFT协议是响应式的，这种简单协议的响应性就成立了。[3]

一个有趣的悖论。 尽管这个提议显然简单，但这个方案仍然令人激动。 一个令人感兴趣的明显悖论是：由于我们不再依赖snailchain来达成交易，为什么不在委员会选举后停止运行snailchain？ 虽然这个提议最初看起来很诱人，但很快就意识到它并不安全。 特别是，对于任何在有界多项式时间后停止执行工作证明的协议，攻击者总是可以创建一个与真实执行一样分布的模拟执行，这样一个延迟产生的节点无法区分真正的执行从模拟的一个。 我们在Section中将这个下界正式化 9,并表明任何安全的无许可协商一致都必须无限次地调用工作证明（即使对于静态安全和同步模型）。

2.2.2 处理轻度适应性腐败

从静态到轻度适应性腐败。 上述有静态委员会的方案对于适应性对手来说是不安全的，因为对手一旦当选就可以简单地腐化委员会。 不幸的是，任何从n个节点向λ大小委员会选择的方案都必须容易受到这种自适应攻击。

但实际上，节点的腐败通常不是即时的，因为需要一段时间才能感染一个干净的主机。 因此，我们定义了一个稍微放松和现实的腐败模型，从此以后称为τ腐败腐败。 在这个模型中，粗略地说，对手可以向节点发出“目标损坏”指令；然而，接收“目标损坏”的节点在τ时间步后才会变坏。

我们在这个τ-agile腐败模型下展示了一个积极的结果。 我们的主要想法是依靠轮换委员会。 当一个诚实节点的链长度达到R·csize +λ时，第R个委员会通过首先删除尾随的λ个区块来选出，然后从这个删减的链中选出

[4]请注意，相比之下，并行和独立的工作Byzcoin [37] 没有明确地做出这种观察，因此他们的协议和安全保证出现了不足之处。

最后的csize块作为委员会的矿工。 这个想法是，如果一个对手成为一个委员会成员，一旦他挖掘出一个区块（这将允许他被纳入一个委员会），那就太迟了。 在适当的τ的情况下，当节点实际上变得腐败时，委员会的任期将会结束，下一个委员会已经接管了！ 我们把它作为一个令人兴奋的开放性问题是否完全适应性安全是否可能用于响应式无许可协议。

技术挑战。 虽然高层次的想法可能看起来很简单，但事实证明，定义和处理轻度适应性腐败引入了各种技术细节 – 具体而言，我们需要以保证并发构成的方式正确处理委员会切换；而且，对手可以追溯腐败旧委员会的可能性也带来了新的挑战。

日常操作。 今后为方便起见，我们会说每个委员会服务一天，并输出每日日志。 我们的混合共识协议本质上是输出这些日志的连接。 我们现在描述委员会成员和非成员的日常运作。

- *委*员会成员。 在每一天R，第R委员会将运行一个BFT实例。 委员会成员将继续运行BFT协议来提交事务，直到它收到一个
  “停止”指令，在此时调用特殊的停止过程。 因此，委员会成员将逐渐输出已承诺的交易。 提交的事务将填充节点的日志记录R。

  每当一个诚实的委员会成员向其日志R添加一个新的事务tx时，它将签署元组（R，f，tx），其中R表示当前日期，而f表示当日tx的序列号。 然后诚实的委员会成员将签名的元组闲聊给网络。
- *委*员会非会员。 非会员从网络上听到已签署的交易。 每当
  一个非成员听到一个元组（R，f，tx）已经被多于[1]通信部分R签名，[3]
  成员，他添加了tx到它的日志R:

  如果日志$_R$[f]未被填充: 日志$_R$[f]: = tx

  请注意，非会员委员会可以将它的日志写入日志R[f]，因为可能会收到乱序信息。 但是，只有在所有前面的事务都已提交之后，才能处理事务。 之后，当我们定义每个节点的输出LOG时，我们强制总是按顺序将LOG写入LOG中 – 如果委员会非成员将LOG的最长连续前缀R输出到LOG，则可以实现此操作。

委员会切换。 每当一个节点进入一个表示R + 1的新的日子时，就需要进行委员会切换。 正确地实现委员会的切换而不会在两者之间留下不活动的空白是有点棘手的。 我们提出一种方法，在每天开始时，新委员会开始新的BFT实例，同时老委员会启动现有实例的停止过程。 这会引入一些细微之处，因为在瞬态窗口中，BFT协议的两个（或更多）实例可能正在同时执行 – 在这些情况下，节点需要正确线性化多个实例的输出；我们还需要确保多个BFT实例的并发组合。

下面我们首先描述前一个BFT实例的停止过程；然后我们描述节点在可能同时执行BFT实例的情况下如何输出线性化日志。

在下面，让R是前一天。

- 上一届委员会成员。 如果一个节点是第r个委员会的成员，表示为$comm_R$，则它向前一个BFT输入一个特殊的带符号停止事务 - 一个节点可以运行多个BFT
虚拟节点，在这种情况下，一个签名停止事务被输入到每个BFT虚拟节点。 当BFT的日志收集足够多的由不同的委员会成员公钥签署的这些停止交易时，日志会被最终确定，并且所有后来的交易都会被忽略。 此时此刻，
我们说以前的BFT已经终止。 当前一个BFT终止时，通信成员$_M$将签署该元组（R，| $\log_R$|）并将签名的元组闲聊到网络。 这允许通讯的非成员$_M$确定日志$_R$何时结束。

  此外，一个诚实的委员会成员签名（R，散列（日志$_R$））散列是碰撞抵抗，并提出签名元组到底层snailchain-我们此后将此行为称为盖章。 正如我们稍后解释的那样，及时盖印可以追溯到可以追溯腐败老委员会的对手。

  此时，诚实通讯$_M$成员输出“完成”。

- 非上一届委员会成员。 如果该节点不是BFT的委员会成员
实例为天R，它等待通信$_M$成员的$\frac{1}{3}$分数以上担保元组
（R，f）。 发生这种情况时，它知道f是$\log_R$的最终序列号。 因此，在输出“完成”之前，它只是等待所有日志$_R$[：f]被填充。

每当一个节点（非通信成员$_M$的成员）输出“完成”时，其日志$_R$就被认为是最终的。 我们注意到，诚实通信$_{R+1}$成员一旦感知到R + 1日的开始并且不等待他们的日志$_R$是最终的，就会在第R + 1日开始新的BFT实例。 这确保所有通信$_{R+1}$成员在短时间内彼此启动新的BFT实例（而等待日志$_R$为最终会在下一个BFT实例的启动时间产生额外漂移）。

输出一个线性化的日志。 节点需要将他们的日志记录收集到最终日志中，记录表示为LOG - 并且该最终日志必须满足部分中定义的属性 3.2. 尤其是，这个最终的日志LOG以递增顺序输出交易，因为在所有前面的交易已经累计之前，可能无法处理交易。 正如我们指出的那样，委员会非成员可能会写入其日志记录$_R$，表示无序。 此外，当第R日的BFT实例开始时，先前的BFT实例可能没有完全完成，因此$\log_R$会
必须等待日志$_{R-1}$是最终的。

因此，要按顺序输出最终的日志LOG，我们只需定义LOG以包含：

- 最大连续日志记录日志$_1$，日志$_2$，. . .日志$_{r-1}$所有这些都必须是最终的。
- 每日日志日志中最长的连续前缀^{TR}。

链上加盖：防止追溯性腐败。 考虑到足够长的时间，对手最终可能腐败足够多的BFT委员会成员，此时，对手可以通过任何其选择的信息伪造BFT委员会成员的签名。 因此，长期以来，BFT委员会成员的签名是不值得的。 在这种追溯性攻击的可能性下，产卵晚的节点不能依靠计算委员会成员的签名来判定太古老的日志。

为了应对这一挑战，我们依靠链上冲压技术。 当诚实的BFT委员会成员终止他们的BFT实例时，他们将签署每日日志的散列并且提议该元组作为基础snailchain的交易。 我们在适当的情况下证明

参数选择，足够多的诚实的BFT委员会成员的日志散列及时（特别是在节点实际上变得腐败之前）在snailchain上盖章。 通过这种方式，生成较晚的节点可以从基础的snailchain中恢复过去每日日志的正确散列（而不是计算古代委员会成员的签名，然后可能会腐败）。 我们强调，链上加盖仅适用于后期加入的节点恢复历史日志。 在线交易确认只需要对当前委员会的签名进行计数，而不需要等待链上加盖。

2.2.3 连锁品质和弹性

如果采矿是随机抽选，那么这将是理想的选择，在每个街区，大自然都会随机抽取赢家。 如果确实如此，那么我们可以实现完美的链质量，即如果节点的α分数损坏，则大致为1-α链质量。 不幸的是，之前的一些作品 [28,30,47]有
表明Nakamoto的共识不能被认为是完美的彩票由于自私的采矿攻击。 当诚实的节点挖掘一个块时，他们立即宣布块，但是损坏的节点不需要遵循这个规则。 粗暴地说，当一个自私的采矿攻击发生时，腐败的节点会挖掘一个
B区*离开目前最长的一条连锁线，他们扣留了公共区块B*，并继续在自己的私人分支上开采。 如果在某个时刻，诚实的节点碰巧发现了一个新的B区链，此时攻击者立即释放区块B*，并与网络冲击攻击，块B*将比B更快地到达其他节点。以这种方式，
敌手已经成功地抹去了诚实节点的工作 - 事实上每次都是腐败的
节点挖掘一个块，他们有机会擦除一个诚实的节点块。 因此，Nakamoto的共识需要大约3/4的整体诚实才能达到2/3链质量（并且需要2/3链质量才能确保2/3的BFT委员会是诚实的） - 事实上，这也是之前的作品之所以如此（Decker et al。 [23]和拜茨币 [37]）宣称实现最佳弹性的做法不正确。

如果不是Nakamoto的共识，我们会采用Fruitchain来避免链条质量损失带来的弹性损失 [48]作为底层的区块链协议。 如Pass和Shi所示 [48],果实链可以抵御这种自私的采矿攻击，因此达到几乎完美的链条质量，即在典型的参数化下，任何α<1/2的损坏都能达到大致（1-α）链质量。因此，关于Fruitchain的混合共识需要
只有2/3的整体诚实hashpower（大约）来实现安全。

2.2.4 模块化协议组合和形式化推理

为了帮助正式推理和演示，我们的协议通过模块化组合方法进行描述。

每日离线共识。 我们首先构造一个名为DailyBFT的中间抽象，它描述了委员会成员和非成员分别在每天的日志上达成一致。 我们的混合共识协议将为每天的DailyBFT [R]分配一个实例，其中R是日期编号以及DailyBFT实例的唯一会话标识符。 然后混合共识连接这些DailyBFT实例输出的每日日志。

在DailyBFT实例中，每个选出的委员会成员都会产生一个或多个BFT虚拟节点，具体取决于委员会中包含了多少公钥。 如果一个节点没有被选举为委员会，它将计算委员会成员的签名来决定其每日日志。

变量	含义
tx	交易
f	每个BFT实例内的事务的序列号
LOG	每个节点输出的完全有序的日志，LOG总是按顺序填充
日志记录	一个BFT实例的日志，在日志中称为每日记录事务编号f到f^t
$[f: f^t]$	日志[1: f]
日志 $[: f]$	安全参数
λ	对手的一小部分hashpower网络的
α	最大实际延迟
δ	网络延迟（通常是松散的）委员会大小的先验上限，我们的
Δ	协议设置csize: =λ
csize	th: = lcsize / 3l，一个阈值
下	(R) : = (R-1) csize + 1, upper (R) : =
(R) ⊢ (R)	底层snailchain协议中的节点本地链
链[: -λ]	除了节点本地链的最后一个λ块之外
MinersOf (chain	在链中挖掘每个块的公钥[s: t]。 几个公钥可能属于同一个节点。
	签名消息msg，其验证密钥是pk
$\{MSG\}_{PK}$ – $T_{风}$	底层BFT方案的活性参数

表2：符号

我们正式确认并证明了DailyBFT的安全属性：在阅读Section中提供的详细形式时，需要注意以下几点 5.2.

- 虽然较低级别的BFT构件仅为委员会成员声明其安全属性（即一致性和活跃性），但在DailyBFT中，这些安全属性已扩展到
  非委员会成员也是如此。

- 较低级别的BFT构件假设所有委员会成员都是在BFT实例启动之前产生的。 然而，在DailyBFT中，这些安全属性需要扩展到
  委员会非会员后来可能会产生（但不会太晚）。

- 另一方面，DailyBFT并不保证加入太晚的节点的安全性（即一致性和活跃性），因为委员会成员未来可能会腐败得很厉害，在
  委员会成员可以签署任意元组，因此，加入节点后期不能依靠计数签名来决定他们的日志。 我们把它推迟到混合共识来应对这种攻击，通过检查每天在snailchain上加盖的日志哈希来加入节点来恢复古老的日志。

- DailyBFT提供了一个keygen抽象：在每个keygen查询中，DailyBFT生成并输出一个新的矿工公钥pk – 混合共识协议将pk并入块
  被开采。 之后，DailyBFT将接收来自已选择一组pks作为委员会成员的环境的输入。 这是处理委员会密钥的敌对选择性开放的地方。 因此，DailyBFT的安全性证明利用了BFT协议的强大安全性来论证BFT协议在DailyBFT内作为子协议运行时，

将尊重所述的安全特性，包括一致性和活跃性 – 否则可以构建一个减少签名安全性的减少。

- 最后，与BFT相比，DailyBFT还实施了满足两个属性，及时终止和终止协议的终止程序。 及时终止
  说诚实的节点收到停止指令后，BFT协议很快终止。 终止协议规定，所有诚实节点在终止时输出相同的最终日志。
  终止是通过让诚实的BFT虚拟节点向底层BFT输入一个特殊的带符号停止事务来实现的。 当1 | comm | / 3l停止由不同的委员会成员密钥签名的交易已经累积在日志中时，所有后来的交易都将被忽略，并且日志为
  最终确定。

杂交共识。 我们现在描述我们的最终产品，即混合共识协议。 杂交共识消耗了DailyBFT的多个实例，其中轮流委员会就每日日志达成一致。 混合共识主要做以下几点：

- 它使用snailchain作为全局时钟有效地管理DailyBFT实例的产卵和终止，从而在诚实节点之间提供弱同步；
- 回想一下，每个DailyBFT实例并不能确保产生太晚的节点的安全性，因为委员会成员将来可能腐败得很厉害，他们可以
  签署任意元组。 因此，混合共识引入了链上加盖机制，以便将安全保证延伸到即使晚点产生的节点。 具体来说，委员会成员在他们的BFT实例终止时将其签名的每日日志哈希标记到snailchain上。 迟到产生的节点将依靠这个链上邮票来识别和恢复过去的日常日志（而不是从委员会成员的连锁签名中算出）。

2.2.5 主要定理

下面我们陈述我们的主要定理：通过在混合共识中使用Fruitchain作为潜在的snailchain，我们获得具有最佳弹性的第一个响应式无权限共识协议。 主要定理的证明将在后面的文章中介绍。

令α表示敌对分数，令ρ表示单个节点在每个时间步骤中挖掘块的概率，令λ表示混合共识的安全参数，并且令η为混合共识期望的额外参数（例如委员会大小将被选择为 λ^{η}），让 κ 表示底层Fruitchain协议的安全参数，我们有以下
定理：

定理3（关于Fruitchain的混合共识）。 对于任意（任意小）常数E＞0，让 $\alpha=1-E$，对于每个n，δ，存在足够小的 $\rho:=\Theta(\lambda)$，合适的 $\kappa=\Theta(\lambda)$ 常数η＞0，使得具有参数（ρ，κ）的Fruitchain上的HybridConsensus$^{\lambda,\eta}$对于任何合适 τ 的任何ppt（n，α，δ，τ）– 有效（A，Z）是安全的，使得Γ TF348）（n，α，δ，τ）= 1[5] 实现

$$T_{\text{暖身}}:=1.5\lambda\ (1+\frac{1}{\eta})\ /\ (1-5\eta)\ n\rho,\qquad T_{\text{确认}}:=O(\lambda\delta)$$

[5]可接受的对手的正式定义将在正式章节中介绍。

请注意，在上述定理中，即使在受到攻击时，$T_{确认}$参数也针对最差情况的交易确认时间。 在乐观的情况下，混合共识达到的交易确认时间为$O(\delta)$。 此外，尽管该定理是根据网络的实际延迟δ来表述的，但实际上我们必须预先确定上限估计（表示为Δ）来参数化难题难度级别ρ。 只要Δ确实是δ的上界，则安全性由上述定理保证，并且该方案实现响应，即，交易确认时间不取决于上限Δ，而是取决于实际网络延迟δ。 如前所述，如果我们选择较宽松的估计值Δ（即Δ的值较大），那么该方案将用更困难的难题进行参数化 - 一方面，这允许我们容忍更高比例的损坏节点；另一方面，敏捷性参数τ以及协议的预热时间将相应增加。

备注1（关于响应性的说明）。 看起来反直观的是，既然我们的方案是响应式的，为什么我们仍然需要关于Δ的先验知识，即网络延迟的上限？ 特别是，我们可以简单地选择参数Δ为无穷大吗？仔细检查后，我们的敏捷参数τ和协议的预热时间$T_{暖身}$都将取决于Δ。 如果我们选择一个更大的Δ，潜在的snailchain会采用更难的谜题，因此容忍更多的腐败节点；但另一方面，我们正在交易敏捷性和协议的热身时间。

2.3其他相关工作

我们现在回顾其他相关工作。 比较与最密切相关的工作，包括Decker et al。 [23]和拜茨币[37]如前所述。

扩大分散的共识。 比特币和分散的，无许可证的加密货币的可扩展性是一个非常明显的问题，并且导致了激烈的讨论，并且在社区中激烈的辩论 [22,53]. 加密货币社区已经提出了各种增量补丁来缓解近期的可扩展性压力，包括调整块大小和其他 [8,26,31,32,54].

Eyal等人 提出BitcoinNG [27], 在每个时代使用一个缓慢的snailchain协议来选举一个单独的领导者，领导者负责在其任命期间合并和线性化交易。 实际上，BitcoinNG可以被看作是流水线块传输，通过分解传输块传输 - 有效地减少Nakamoto共识中最坏时延Δ的上界。 BitcoinNG仍需要节点等待底层snailchain中的$\Theta(\lambda)$块（在BitcoinNG中称为密钥块）才能稳定，以便在λ是安全参数的情况下确认事务。 相比之下，混合共识是敏感的，并且乐观情况下的交易确认时间仅为$O(\delta)$，其中δ是网络的实际延迟，而不是先验已知的上限Δ。 BitcoinNG并没有对他们的协议给予正式的处理，但可以想象，他们的协议可以被证明实现了无许可的共识抽象。

侧链 [9]是解决比特币的可扩展性问题的另一个值得注意的努力。 Side-chain的想法是支持主要比特币区块链的共识协议，并且侧链中的货币与比特币挂钩。 侧链协议缺乏正式的保证，协议描述和实现仍然有些不完整。

其他各种方法 [11,38,49–52] 最近被提出以不同的信任假设在“分散”环境的不同变体中达成共识。

允许达成共识。 过去30年来，社区广泛调查了许可模式中的共识协议 [14,15,21,25,29,35,39–41]. 这些作品通常考虑三种不同的模型：1）同步模型 [24] 协议在循环中进行，并且一轮中发送的消息保证在下一轮开始时到达接收方； 2）部分同步模型 [25] 网络有一个有界的延迟参数，但协议不知道这个延迟；和3）异步模型 [14,15] 网络延迟可能无限增长。

我们的网络模型类似于部分同步的标准概念 [25] 但不一样。 虽然我们允许协议知道网络延迟的先验上限，但我们的目标是实现响应。 我们强调任何等待网络延迟并将延迟视为同步轮的协议本身无法响应。 如果协议确实等待网络延迟作为同步轮，这确实会转化为同步模型，但是，我们的响应性要求使得在我们的网络模型中协议的设计不是微不足道的。

早期关于许可共识的着作也考虑了重组。 例如，垂直Paxos [40] 和BFT-SMART [12] 允许节点以动态方式重新配置。 这些作品考虑了群体重构的一个相关但有些不同的目的。 调查这些技术是否可以适应我们的设置来执行委员会成员的切换会很有趣。 然而，我们指出，早期的组重构技术并不能证明在选择性开放攻击下的安全性（事实上，大多数这些作品并未采用密码学上合理的推理框架）。 如果我们要适应这些技术，则需要一种新的密码合理的治疗方法。

分布式系统和密码学。 共识和分布式系统与密码学密切相关，如多方计算（MPC）。 一方面，多方计算（MPC）实质上依赖广播或分布式共识原语来实现一致性并且可能活跃，通常称为MPC环境中的保证输出。 另一方面，分布式共识协议通常使用密码来确保安全。 例如，经过认证的拜占庭式模型 [24] 利用数字签名，密码学家将这种设置假设称为公共密钥基础设施 [16, 17, 20].

另*一方面，未经认证的拜占庭模型* [41] 在分布式系统中实际上被密码员称为认证信道模型 [16, 17, 20]. 当协议采用计算安全的密码学原语时，我们隐含地假设网络的延迟必须在安全参数中被多项式约束（但可以是异步情况下的无限多项式），因为我们无法保证指数级长的协议的安全性。 当分布式一致性协议使用计算安全基元时，最好的做法是依靠计算减少来证明协议的安全性 – 已经很好理解，将密码系统建模成最自然的黑盒而没有仔细审查可能会导致错误并有缺陷 [5–7, 10, 13, 18, 33, 34, 43, 44].

我们的论文展示了我们采用协议组合框架的这种方法 [16, 17, 20] 由密码学家开发推理分布式系统协议
– 我们表明这样做是必要的，特别是通过处理选择性开放攻击。 如果我们没有采用正式的，密码合理的推理框架，这些问题很容易被忽视。

3 问题定义

功能极其微不足道。 在一些安全参数$\lambda \in \mathbb{N}$方面，本文中的所有安全失败将被表示为（指数）强弱函数。我们说
如果存在一些常数$c_0 > 0$，c_1，函数$\mathrm{negl}(\bullet)$可以忽略不计
$\lambda \in \mathbb{N}$，$\mathrm{negl}(\lambda) \leq \exp(-(c_0\lambda + c_1))$。 在本文的其余部分，我们只是简单地使用可忽略的术语，但它的所有用途可以被强烈忽略的自动替换。

3.1 正式模型

执行模式。 我们假设以下执行模式：

- *交互式图灵机。* 我们假设一个标准的交互式图灵机（ITM）模型 [16, 17, 20] 通常在密码学文献中被采用（但是增加了作为例证的工作证明）
  后来变得平坦）。 有一个潜在的全球时钟会随着时间而增加； 每个时钟滴答被称为原子时间步骤。

  节点可以在每个原子时间步骤执行无限多项式计算量，以及多项式地发送和接收许多消息。虽然本文没有明确指出，但节点接收来自环境Z的输入并将其输出发送到环境。

- *验证的工作。* 我们假设有一对随机预言（H, H.ver）。 在每个原子时间步中，每个节点最多可以创建一个H oracle查询，但是是一个无限制的查询
  （多项式）数量的H.ver查询。 如果存在多个区块链协议实例，我们假设每个协议实例都有自己独立的随机预言器。 环境不能直接查询随机预言器，但可以通过对手的帮助查询随机预言器。

- *腐败。* 在任何时间点，环境Z都可以以任意方式与腐败节点进行通信。 这也意味着环境可以看到腐败的内部状态
  节点。 腐败节点可以任意偏离规定的协议，即展现拜占庭故障。 所有的腐败节点都由表示为A的概率多项式时间对手控制，并且对手可以看到腐败节点的内部状态。
  对于诚实的节点，环境不能观察它们的内部状态，但可以观察通过协议定义输出到环境的任何信息诚实节点。 关于腐败模式的细节将在后面介绍。

- *网络交付。* 敌手负责在节点之间传递消息。 我们假设对手有能力推迟或重新排序消息，可能会受到影响
  如下所述的某些限制。

τ-严重腐败。 在标准的自适应腐败模型中，只要环境希望破坏节点，腐败就会立即发生。 我们的协议在稍微宽松的自适应腐败模型下被证明是安全的，我们称之为τ-agile腐败。 粗略地说，τ-agile腐败表示，环境实际上损坏一个节点需要一段时间。 更正式地说，根据以下程序，允许环境破坏并产生新的节点：

- *延迟腐败。* 我们假设环境可以自适应地破坏一个节点，但有以下限制。 为了腐蚀节点i，环境必须发出“目标损坏”

指示节点i在某个时间点表示为t。 节点i不会立即变坏，而是保持诚实，直到t +τ，并且在t +τ时刻变得腐败 – 此时，腐败节点i与环境任意通信并且可以随意与协议偏离。

- *杀死*一个腐败的节点。 一旦节点实际上变得腐败，环境就可以发出“杀死”指令来终止节点。被杀的节点不再活着。 环境不能杀人
直接的诚实节点，而不会首先破坏它们。

- *产生*新的节点。 环境也被允许产生新的节点，无论是诚实的还是腐败的节点。 在时间$t_{卵}$产生的节点在时间$t_{卵}$被认为是有效的。 产卵
一个腐败的节点相当于增加瞬间发生的攻击者的散列能力。 如果产生了一个诚实的节点，如果它希望稍后损坏该节点，则环境必须遵循延迟腐败程序。 一个诚实的，新产生的节点开始运行主协议。

我们说一个节点（已经生成并且没有被杀死）可以有三个互斥的状态：

1. *完*整：没有收到“目标损坏”指令的诚实节点。
2. *腐*败前：一个诚实的节点，已经收到了“目标腐败”指令，但尚未腐败。
3. *腐*败：这个节点可能会被腐化，或者说是诚实的，但却收到了“目标腐败”指令，实际上已经腐败。

完整节点和预腐败节点都被认为是诚实的。

从此以后，每当我们说“一个诚实的节点我在时间t执行某些动作”，我们就意味着节点i在时间t是诚实的。 例如，如果我们说一个诚实的节点在时间t向环境Z输出一条消息，我们隐含地意味着节点在时间t是诚实的（但
可能会在以后变得腐败）。 或者，如果我们说一个诚实的节点执行一个动作，它
意味着该节点在执行动作时是诚实的，尽管它在未来某个时候可能会损坏。

完全自适应的腐败和静态腐败。 我们注意到0-敏捷腐败相当于完全自适应的腐败模型，其中环境Z可以立即破坏节点。 在完全自适应的腐败模型下，如果它是诚实的，那么节点是完整的。

我们还在无权限模型中定义了静态腐败 – 静态腐败是一种较弱的腐败模型，仅用于证明我们的下限 – 注意，假设较弱的腐败模型会产生更强的下限。 我们假设在静态腐败模型中，环境可以随时产生诚实或腐败的节点。 但是，一旦产生了一个诚实的节点，环境就不会在以后损坏它。

八卦网络模型。 我们假设诚实节点发送的所有消息都分布在八卦网络上，并最终可以被所有其他诚实节点听到。 对于在时间T加入的节点，它将在时间T之后接收由诚实节点发送的所有消息。一个诚实的节点不需要知道网络中其他节点的身份就可以向所有其他节点传递消息。

攻击者不能通过诚实的节点丢弃或修改消息，但可以重新排序或延迟受到某些限制的消息。敌手可以选择性地将消息传递给子集，但不是所有的诚实节点。

我们假设消息发送者的身份是未知的。 消息可以被签名，但是一个诚实的节点并不知道公钥与节点的物理身份之间的对应关系。

我们定义了以下类型的八卦网络，它们对对手延迟和重新排序信息的能力施加了不同的限制：

- *同*步模型。 在同步模型中，在时间t由诚实节点讲话的消息
  在时间步骤t + 1保证到达所有诚实节点，可能不按顺序。此外，
  历史消息会立即传递给新产生的节点。

- δ-部分同步模型。 在一个δ-部分同步模型中，在时间t由一个诚实的节点讲话的消息被保证到达所有诚实的节点，可能无序，按时间
  t +δ。 而且，历史消息会立即传递给新产生的节点。

  更正式地说，假设一个诚实的节点在时间步骤t闲聊消息。 ≤ t^*，那么如果一个节点i在时间步$t^*+\delta$是诚实的，那么它肯定收到了这个消息。

请注意，在实践中，诚实的节点可以实现历史抄本检索服务 - 这样，当产生共识实例时，节点可以获得整个历史抄本的副本。 不难看出，如果任何一个诚实的节点保持诚实，并且至少能够活3个小时，那么不会有历史成绩单丢失。

正如后面的部分所述 3.2,我们允许我们的协议知道可能松散的上限
Δ对网络延迟的影响，因为底层的snailchain必须知道这样一个参数化挖掘难度的上限。 但是，我们要求协议具有响应性，即它的实际性能只能取决于网络的实际δ值，而不是松散的上限
Δ.

随机协议执行和概率空间。 设Π为某协议，令A，Z为概率多项式时间（或简称ppt）算法，令λ∈N。令exec［Π］(A，Z，λ)为表示共享视图的随机变量所有节点（即他们的所有输入，随机硬币和...）
收到的消息，包括来自随机预言的消息）。

让属性成为一个函数，它把一个固定的视图作为输入，并输出0或者1.在整篇论文中，每当我们说概率为p的exec［Π］(A，Z，λ)时，我们正式表示

$$\Pr[\text{视图} \leftarrow^{\$} \text{exec}[\Pi](A, Z, \lambda) : \text{属性}(\text{视图}) = 1] = p$$

其中概率被用于A，Z，所有诚实节点的所有随机硬币以及所有随机的神谕。

符合执行。 我们经常对敌手A和Z施加约束来证明协议的安全属性。 因此，我们根据对（A，Z）的约束条件来定义我们认为合规执行的内容。

定义1（（n，δ，τ） - 有效（A，Z））。 如果A和Z是概率多项式时间算法，那么对于（n，δ，τ）是（n，δ，τ） - 有效的wrt协议Π，对于每个λ∈N，下列属性以概率1保持为exec［Π］(A，Z，λ)：

1. 在任何时间点，活动节点的数量[6] 是n；

2. *在*最多δ个时间步骤中延迟来自诚实节点的消息；

3. 一旦一个诚实的节点接收到来自环境的输入"目标腐败"，它至少需要τ
节点变坏之前的时间。

定*义2（（n，α，δ，τ）- 有效的（A，Z））。 我们说对（A，Z）是（n，α，δ，τ）- 有效的wrt协议Π，使得（A，Z）是（n，δ，τ）1,而且，对于任何λ∈N，在任何支持exec [Π]（A，Z，λ）的视图中，都认为在任何时候，不超过活动节点的α分数或者处于腐败或者预先确定的状态，腐败状态。*

在整篇论文中，尽管没有明确指出，包括n，α，δ和τ在内的所有参数都是安全参数λ中的函数。此外，为了符号简单，在本文中，我们没有明确定义事务输入的有效性规则。 然而，扩展我们的定义以纳入交易有效性规则并不难，例如Garay等人。 [30] 和Pass等人。 [47].

我们的协议在选择包括满足特定约束的n，α，δ和τ的参数时可能是安全的。 因此，我们定义了Γ可容许性的概念，其中Γ是对参数选择施加约束的函数。

定*义3（Γ可接受）。 Γ是参数n，α，δ，τ中的二元函数。 我们说那个如果存在n，δ＞0且α，τ≥0，那么ppt对（A，Z）是Γ可容许的。*

- Γ（n，α，δ，τ）= 1；
- （A，Z）为（n，α，δ，τ）-valid由定义 2；

3.2 问题定义：Permissionless Consensus

我们的HybridConsensus协议在无权限模型中实现了状态机复制抽象 - 从此我们将此抽象称为无权限共识[7]。 在无许可协议的协议中，节点维护一段时间的LOG，这是一个事务列表；并且进一步保证了一致性和活力。 我们无权的共识抽象是由Garay等人采用的"公共账本"抽象的变体。 [30] 和Pass等人。 [47]. 差异是不重要的。 特别是，我们基本上将事务mempool并入我们的抽象，这样保持事务mempool不会留给调用者。 相比之下，Garay等人的公共账本抽象 [30] 和Pass等人。 [47] 将其留给呼叫者以维护一个事务mempool。

更正式地说，无许可的共识满足以下抽象。

输入和输出。 环境Z可以在每个时间步骤向每个诚实节点输入一组事务TX。 在每个时间步中，一个诚实的节点向环境Z输出一个完全有序的LOG事务（可能是空的）。

[6]原则上，放宽这一要求并不难，并且允许节点的数量变化到一个恒定的因子，但是底层snailchain的链增长参数需要相应地调整。

[7]在经典的分布式系统文献中，术语共识也用于表示单次同意
协议。 在本文中，共识意味着状态机复制，其中节点想要就随时间增长而线性排序的日志达成一致。

安全性定义。 令ppt算法 (A, Z) 为 $(n, \alpha, \delta, \tau)$ - 对于无权限的共识协议Π有效。 令$T_{暖身}$, $T_{确认}$, $T_{引导}$为 λ , n, α , δ 和 δ 中的多项式函数 Δ。 我们说，如果存在一个可忽略的函数negl，那么对于每个$\lambda \in N$，一个无权限的共识协议Π是具有参数 $(T_{暖身}, T_{确认}, T_{引导})$ 的安全wrt (A, Z) 用1 - negl (λ) 概率，以下属性适用于exec $[\Pi](A, Z, \lambda)$:

- 一致性：一致性包括以下属性：
  - *通*用前缀。 假设一个诚实节点i在t时刻输出LOG给Z，而一个诚实节点j（相同或不同）在时间t'输出LOG'给Z，它认为$LOG \prec LOG'$或$LOG' \prec LOG$。 这里的关系$\prec$表示“是前缀”。 按照惯例，我们假设 *任*何x的$x \preceq x$和$x \prec x$。
  - *自*洽。 假设节点i在t和$t' \geqslant t$时刻诚实，并输出LOG和 LOG'分别在时间t和t'，它认为$LOG \prec LOG'$。
- *活*性：假设环境Z在$t \geq T$时输入TX到一个诚实的节点$_{暖身}$。 假设在t时刻$_{前}$产生了一些节点i，并且在$t' \geqslant \max(t_{前} + T_{引导}, t + T_{确认})$之前保持诚实。 令LOG为节点i在时间$t'$的输出，它认为任何$tx \in TXs$都包含在LOG中。

  直观地说，活跃度表示交易输入到一个诚实的节点被包含在$T_{确认}$时间内的LOG中。 此外，$T_{暖身}$被称为协议的预热时间。

请注意，上述定义是关于特定 (A, Z) 对的。 但是，我们的主要定理稍后将声明任何ppt (A, Z) 的HybridConsensus协议的安全性，只要它们遵守某些约束。

备注2. 对于我们的混合共识协议，$T_{引导} = 0$，即新产生的节点立即自举。 因此，我们经常省略编写$T_{引导} = 0$的术语而没有模棱两可的风险。 然而，我们的问题定义允许多项式$T_{引导}$，因为这将允许我们证明更强的下界。

响应性。 如果活性参数$T_{确认}$仅取决于网络的实际 δ ，而不是用于参数化协议的松散上限 Δ ，我们认为一个无权限的一致性协议是响应。

4 建筑模块

4.1 基础区块链协议snailchain

我们的主要方案是效率自举，我们从一个潜在的低速区块链引导，表示为snailchain，以获得一个快速交易确认和高吞吐量的无权限共识协议。

我们假设潜在的慢共识协议表示为snailchain（例如，比特币的Nakamoto共识 [46]）实现了“区块链”抽象，这可以被认为是一个特殊情况下每个无任务的共识协议 3.2.

抽象。 我们假设snailchain协议提供以下输入/输出抽象。

*输*入。 在每个时间步骤中，环境Z输入到每个诚实节点（recs，pk），其中recs表示一组记录，并且pk表示公钥。

*输*出。 在每个时间步骤中，诚实节点向环境输出以下内容：

$$\text{链}:=\{(\text{recs}_i, \text{pk}_i)\}_i$$

有用的概念。 我们定义了以后会有用的以下概念。

*本*地链。 在每个时间步中，一个诚实的节点向环境输出一些链，为了简单起见，我们在这个时间步中将此链称为诚实节点的本地链。

*完*整和诚实的块。 给定链表示在某个时间t一个诚实的节点的本地链，我们可以定义链中的每个块对链的前缀而言是否完整（或诚实地对待）。 一个块链[j]：=（recs，pk）据说与前缀链[：j']完好无损（或诚实对待）
其中j'<j如果在某时刻$t' \leq t$存在一些完整的节点（或诚实的响应），那么
1）节点i在时间t'将输出链'转换为Z，使得在时间t + 1处链[：j']→链'和2）Z输入（recs，pk）到节点i。
非正式地，对于一个诚实的聚会链表示链，块B：=链[j]是
完整的（或诚实的）对于前缀链[：j']其中j'<j，如果早些时候有一些诚实的节点在其本地链包含前缀链时接收到块B作为输入[j']。

安全性定义。 与之前的作品类似 [27,30,36,47],我们为snailchain协议定义以下属性。 在以下所有中，概率是根据所有诚实节点，环境Z以及执行中的对手A所消耗的随机性来定义的。

假设（A，Z）是（n，α，δ，τ） - 无效的srt链。 设W_C，W_Q，W_G为多项式函数，λ中的tions。 令Q，G，G为λ，n，α，δ和Δ中的多项式函数。 我们说那个蜗牛协议满足W_C一致性，（W_Q，Q）链质量和（W_G，G，G^t）链增长至（A，Z）可忽略的函数negl使得对于任何$\lambda \in N$，除了negl（λ）失效概率外，以下属性适用于exec [snailchain]（A，Z，λ）：

- 一致性。 对于任何在t时刻诚实的节点i以及在时间$t' \geq t$时刻诚实的任何j（相同或不同），让链表示在t时刻我输出到Z的节点，并让链'表示在t时刻节点j输出到Z的哪个节点'，它认为

$$\text{链}[:-W_C]\text{链}'$$

- *链*质量。 让链表示在任何时间t一个诚实的节点输出到Z的东西。 那么对于任何$\lambda_1 \geq W_Q(\lambda)$，如果|链| $> \lambda_1$，它认为对于任何i<|链| - λ_1，链[i：i + λ_1]中的至少1Q$\lambda_1$1个块的数目是完整的链[：i-1]。 换句话说，在任何时候，在一个真实节点的输出链中任何$\lambda_1 \geq W_Q(\lambda)$连续块的窗口中，至少Q个块的块与窗口的前缀完好无损。

- *连*锁增长。 在每个时间步骤中，以下属性都成立：
  1. 一致的长度。 假设一个诚实节点在时间t输出链。 它认为任何诚实节点都必须至少输出一个链长 在任何$t' \geq t+\delta$。
  2. *连*锁增长。 假设一个诚实的节点i在时间t输出链，在时间$t' \geq t$时一个诚实的节点j（相同或不同）输出链'，假设G●（$t'-t$）≥W TF661）（λ），它认为
  $G\cdot(t'-t) \leq |\text{链}'| - |\text{chain}| \leq G^t \bullet (t'-t)$。

因此，直观地说，连锁增长说1）诚实的节点具有大致相同的链长，2）诚实节点的链不能增长得太慢。

为了方便起见，我们现在定义一个名为liveness的派生属性。 如果我是一个诚实的节点，让 $(recs^t,)$ 表示在t时刻环境Z向诚实节点i输入的内容。 让rec记录一下。 我们说如果 $rec \in recs^t$，则Z在时间t建议记录节点i。

我们说snailchain协议满足活性wrt (A, Z) 和活性参数$T_{蜗牛}$，如果存在一个可忽略的函数negl，使得对于任何 $\lambda \in N$，具有1 - negl (λ) 概率， exec [snailchain] (A, Z, λ)：

- *活跃度。* 让rec记录一下。 如果对于每个诚实节点i，对于每个$t' = t, t + 1, \ldots$，环境Z建议rec，除非rec已经包含在节点i的输出中[8] $i \geq t + T_{蜗牛}$，如果一个诚实的节点输出链^t，则rec必须包含在链中$^t[: -\lambda]$。

直观地说，活跃度只是说如果环境Z继续输入相同的记录
记录$T_{蜗牛}$时间的所有诚实节点，然后rec将包含在所有诚实节点中
本地链最多$T_{蜗牛}$时间。

引理1（生存为派生财产）。 对于任何ppt算法A，Z，任何Q〉0，G≥G〉0，任何W_c，W_a，W_c使得W_c (λ) + W_a如果snailchain满足W_c一致性，（W_a，Q）链质量和（W_c，G）G（G） （ λ ），λ（λ ）TF698）） - 链增长wrt (A, Z)，则snailchain满足活性参数$T_{蜗牛}$= (W_c+ W_q+ λ) / G的活性wrt (A, Z)。

请注意，为了方便以后的应用，我们定义了一个略有修改的活性版本，与Garay等人比较。[30], Pass等人 [47], 和Fruitchain [48]. 直观地看到，我们的活性概念是帕斯等人所采用的暗示。[47] 和Fruitchain [48].

底层的snailchain是一个无响应的无许可的共识。 不难看出，我们在章节中定义了我们的基础snailchain抽象 4.1 可以被看作是一个“无许可共识”协议的特例实例。 特别是，每个节点的LOG都可以
链中的记录的有序列表[: -λ]。 由于我们需要在典型参数下将预期块间隔设置为Θ（Δ），所以这种无权限的一致性协议是无响应的，其中Δ是
网络延迟的先验上限。 因此$T_{确认}$: =Θ（λΔ）。

4.1.1 中本作为潜在的snailchain

Garay等人 [30] 证明Nakamoto的共识 [46] 假设完全同步的模型满足上述属性的变体，即消息即时传递并且不能被敌手推迟。 Pass等人 [47] 加强这些性质并证明Nakamoto共识在δ的适当条件下在δ-部分同步网络中满足它们。

下面我们重申Pass等人的主要定理。 [47] 为底层snailchain。 设α和β分别表示α+β= 1的腐败节点和诚实节点的比例，设ρ表示单个节点在一个时间步中挖掘一个有效块的概率。 ρ与采矿难度参数密切相关。

- 假设p: = 1- (1 - ρ) $^{\beta N}$表示某个诚实节点在单个时间步中挖掘块的成功概率。

[8] 实际上，我们可以在Nakamoto协议中执行以下优化：诚实算法可以抑制记录rec，如果它已经包含在它试图扩展的最长链中。

- 令q：=αnρ表示对手可以在单个时间步中挖掘的期望块数的上限。
- 令γ：=$\frac{p}{1+\Delta p}$可以认为是由网络延迟 δ 打折的p的一个版本。

定义4（snailchain$\Gamma^{蜗牛}$的容许参数）。 我们定义 $\Gamma^{蜗牛}_{\rho}(n, \alpha, \delta, \tau)$ = 1 *以下是：*

- *n*> 0，δ> 0，τ≥0都是λ中的多项式函数； α> 0是常数；
- 存在一个常数η> 0，使得p（1-（2δ+ 2）p）≥（1 +η）q。

定理4（Nakamoto作为潜在的snailchain [47]）。 *对于任何常数*η_0，η_1，η_2，η，$\rho > 0$，*令*$Q = 1-(1+\eta_0)^{q}$，*令*$G = \gamma/(1+\eta_1)$，*设*$G^{t} = (1+\eta_2)n\rho$，*Nakamoto共有协议 [46,47]（以下简称snailchain）参数化为采矿难度参数*ρ*满足*ηλ*-一致性，（*ηλ，*Q*）链质量和（*ηλ，*G，*G^{t}*）链增长的任意ppt* $\Gamma^{蜗牛}$ *- 允许重链snailchain。*

典型的参数化。 通常在实践中，我们将设置难题的难度参数ρ：=Θ（Δ^{-1}），其中Δ是网络延迟 δ 的先验已知上限。 在这种典型的参数化下，我们需要大约3/4的整体诚实来确保大约2/3链的质量。

推论1（Nakamoto作为潜在的snailchain [47]）。 *对于任何（任意小）常量* $E > 0$，*设*$\alpha = 1 - E/4$，*那么对于每个n，*δ *存在足够小的*$\rho_0 := \Theta(\delta^{n})$，*任何常数*$\eta > 0$，$\eta^{t} > 0$，*Nakamoto的协议与采矿难度参数*$\rho < \rho_0$*满足*$\eta^{t}\lambda$*-一致性，（*$\eta^{t}\lambda$，*Q*）链质量和（*$\eta^{t}\lambda$，*G，*$G^{t}$*）链增长对任意*$\Gamma^{蜗牛}_{\rho}$*（*A，*Z*）在*哪里*

$$Q > \frac{2}{3}, \quad G = \frac{3}{4}n\rho, \quad G^{t} = (1+\eta)n\rho$$

或者更简单地（非正式地）提出，对于每个$\alpha = 1 - E/4$对于任意小的常数E> 0，存在适当的参数化Nakamoto一致性协议，其实现$Q > 2/3$链质量。

4.1.2 果实链作为潜在的蜗牛链

使用Nakamoto作为潜在的snailchain的问题是链条质量损失。 由于自私的采矿攻击，Nakamoto要求3/4 + E整体诚实，以达到2/3链质量，这是选举委员会2/3诚实所需要的。 由于混合共识采用模块化方法，因此我们可以使用替代品Fruitchain [48]协议，它实现（几乎）与Nakamoto相同的正式抽象。

在高层次上，Fruitchain协议在底层运行了一个Nakamoto共识；然而，矿工们在寻找块时同时开采水果。 水果包含交易，底层Nakamoto区块链中的块包含成果（但不包括交易）。 在Fruitchain协议中，水果被视为新块，并被视为区块链抽象的一部分，但潜在的Nakamoto可被视为协议的内部细节，不需要暴露于外部。 我们将假定Fruitchain协议将以下参数作为输入参数（请参阅Fruitchain纸张以了解详细信息 [48]）：

- 挖掘难度参数ρ和ρ_f，分别用于挖掘块和果实。 此后我们将假设 $\rho_f := \rho$ 是硬编码的（尽管任何常数$c \geq 1$的 $\rho_f := c\rho$ 也应该起作用），因此我们不再明确提及 ρ_f。
- 回顾参数κ，即区块链悬挂水果的距离；
- 新近度参数R，即一个水果被悬挂在底层区块链中的（R·κ）-recent区块时被认为是新鲜的。 今后我们将简单地假设R := 17是硬编码的
  （尽管任何其他大于1的常数也应该起作用），因此我们今后不再明确提及新近度参数[9].

定理5（果链作为潜在的snailchain [48]）。 *(1-α) nρ, $G^{\prime} = (1+5\eta)n\rho$, $Q = (1-5\eta)(1-\alpha)$, Fruitchain协议 [48]用 (ρ, κ=) 参数化满足λ--一致性 (λ/η, Q) - 果实质量, (λ/η, G, G) - 果实生长*

ppt (A, Z) 表示$\Gamma^{\text{蜗牛}}_{\rho}$ - 可用于snailchain。

推论2（Fruitchain作为潜在的snailchain [48]）。 *对于任何（任意小）常量 $\varepsilon > 0$, 令$\alpha = 1 - \varepsilon$, 存在合适的$\kappa = \Theta(\lambda)$和常数$0 < \eta < 1$（与ε有关）；另外 $n, \delta > 0$时，存在足够小的ρ：$= \Theta(\cdot)$，使得Fruitchain与参数（ρ, κ）满足λ一致性，（·, Q）链质量和（·, G, $G^{\prime}$）链增长 η η 任何$\Gamma^{\text{蜗牛}}_{\rho}$ - 可接受 (A, Z) 其中*

$$Q > \frac{2}{3}, \quad G = \frac{2}{3}n\rho, \quad G^{\prime} = (1+5\eta)\, n\rho$$

或者更简单地（非正式地）提出，对于每个$\alpha = 1 - \varepsilon$，其中$\varepsilon > 0$是任意小的不变，有一个适当的参数化Fruitchain协议，可以实现$Q > \frac{2}{3}$链质量。

4.2 强有力的安全许可Byzantine容错

我们将依靠经过许可的共识协议。 众所周知，如何在部分同步网络中构建拜占庭容错（BFT）协议[21, 25, 42]；此外，这些协议实现了响应。

由于与选择性开放式攻击相关的技术细节，我们需要为我们的BFT构建块定义一个更强的安全概念，而不是最自然的基于属性的概念。 我们考虑使黑箱使用签名算法的BFT协议。 设$\Sigma := (\text{Gen}, \text{Sign}, \text{Verify})$表示签名方案。 我们使用符号$\text{BFT}^{\Sigma}$表示协议BFT通过签名方案Σ进行参数化。 此外，我们要求BFT只使用$\Sigma.\text{Gen}$和$\Sigma.\text{Sign}$函数的黑匣子使用 - 在我们下面的表述中，BFT节点查询将提供$\Sigma.\text{Gen}$和$\Sigma.\text{Sign}$谚语的环境。 在形式上，我们假设通过签名方案$\Sigma := (\text{Gen}, \text{Sign}, \text{Verify})$参数化的$\text{BFT}^{\Sigma}$协议实现以下抽象。 Z

输入。 允许环境向诚实节点发送以下输入。 所有其他输入都被忽略。

- 环境Z可以将开始（pk_i, comm）一次发送给诚实的节点i。

[9]我们根据Fruitchain论文中的定理3.1选取R = 17 [48]. 然而，请注意，任何常数R> 1都可证明更紧密的界限。这可以通过在Fruitchain中使用更紧密的水果新鲜引理来完成 [48].

- 如果已经输入了开始命令，则环境Z可以在每个时间步骤将一组事务TX输入到诚实的节点。
- 签名（msg）查询的答案。

输出。 诚实的节点随着时间的推移输出以下项目到Z.

- 如果输入了开始命令，一个诚实的节点将在每个时间步中向Z输出一个完全有序的日志记录。
- 如果输入了开始命令，则一个诚实的节点可以输出到Z表格的查询 sign（msg）其中msg∈{0,1}*表示一个消息。

符合执行。 我们考虑在一个部分同步的网络中执行一个BFT协议，其中有一些静态损坏，详见下文。 Z环境和对手 *A*也必须满足一定的限制。 设$T_{邮票}$为 λ，n，Q中的多项式有界函数， 和δ。 一对概率多项式时间算法（A，Z）被认为是（n，Q，δ，τ，$T_{邮票}$） - 当以下条件成立时有效的BFT：

- （A，Z）是（n，δ，τ） - 根据定义有效的BFT 1.
- *有*点静态腐败。 必须在$T_{开始}$之前声明所有的“产卵”和“目标损坏”指令，其中$T_{开始}$表示启动命令首先通过Z输入到诚实节点的时间。
- *委*员会协议。 如果诚实节点i在时间t接收到来自Z的输入开始（pk_i，comm），并且诚实节点j在时间t'接收输入开始（pk_j，$comm^t$），则认为通信= $comm^t$。 此外，如果i = j，则pk_i / = pk_j。
- *关*闭开始。 假设$T_{开始}$是诚实节点接收输入开始（，）的最早时间。 那么，对于我在t时刻$_{开始}$+ δ 诚实的任何节点，我必须接受时间$T_{开始}$+ δ 的输入开始（，）_。 诚实时，每个节点最多接收一次启动。
- *弹*性。 必须在输入到$T_{邮票}$之前保持诚实的节点的开始命令中指定至少1Q●| comm | 1 pk_i∈comm的数量。
- *签*名oracle的正确性。 对于任何开始（pk_i，）命令输入到诚实节点，pk_i必须在签名方案Σ的有效公钥范围内。

  在诚实节点i的任何符号（msg）查询中，Z立即返回σ，使得Σ.Verify（pk_i，msg，σ）= 1。

安全性定义。 设$T_{邮票}$，$T_{蒲福风级}$为 λ，n，Q和 δ 中的多项式有界函数。 假设（A，Z）是（n，Q，δ，τ，$T_{邮票}$） - 有效的BFT。 那么，对于任何观点的支持 exec［BFT］（A，Z，λ），如果以下属性成立，我们说安全^{Tbft}（视图）= 1。

- 一致性。 一致性包含以下属性：
  - *通用前缀。* 如果一个诚实的节点i在任何时间t 〈$T_{邮票}$和诚实的节点输出日志 *j*（相同或不同）在任何时间t'<$T_{邮票}$输出记录^t，它认为或者记录日志^{t}或 日志^t< 日志。

- *自洽*。 假设一个诚实的节点i分别在时间t和t'输出日志和日志^{t}，使得$t < t' < T_{邮票}$，它必须保存$\log \prec \log^{t}$。

- *活跃度*。 如果Z在时间$T_{开始} \leqslant t < T_{邮票} - T_{蒲福风级}$处将TX输入到诚实节点，则任何在时间$t' = t + T_{蒲福风级}$处诚实的节点将输出日志在时间$t'$，使$TXs \approx \log$。 $T_{蒲福风级}$被称为活性参数。

定义5（*非常安全的BFT协议*）。 *设$T_{蒲福风级}$为λ，n，Q和δ中的正多项式。 我们说协议BFT对于（1 - Q） - 活性参数的腐败具有很强的安全性*
$T_{蒲福风级}$iff对于任何n，$\delta > 0$，$\tau \geqslant 0$，任何正多项式$T_{邮票}$，对于任何ppt A和任何多项式g，存在ppt对手B和多项式g^{t}，因此对于任何ppt Z，
（A，Z）为（n，Q，δ，τ，$T_{邮票}$） - 有效的BFT，对于任意$\lambda \in N$，

$$\Pr[\text{view} \leftarrow \text{exec}[\text{BFT}](A, Z, \lambda) : \text{安全}^{Tbft}(\text{视图}) / = 1] \geqslant g(\lambda)$$

$$\Rightarrow \Pr[\text{view} \leftarrow \text{exec}[\text{BFT}](B, Z, \lambda) : \text{伪造}(\text{view}) = 1] \geq g^{t}(\lambda)$$

在伪造（视图）= 1的情况下，如果在视图中，

- 在某一时刻，敌手向环境Z输出伪造对（i，msg，σ），使得节点i在时间t是诚实的;
- 到时间t，环境Z具有输入开始（pk_i，）到节点i;-
- $\Sigma.\text{Verify}(pk_i, msg, \sigma) = 1$; 和
- 到时间t，节点i尚未向Z提交查询符号（msg），其中答案为σ。

定义微妙：BFT的腐败模型。 我们的BFT构建块必须在“有点静态”的腐败模式下安全。 我们现在详细阐述相关的定义细微之处。 首先，必须在$T_{开始}$之前发布任何“产卵”或“目标腐败”指令，即当第一次启动输入到一个诚实的节点时 - 在这个意义上，安全概念似乎是“有点静态的”。 另一方面，由于以下原因，我们的安全概念比标准的“静态”安全概念更强大：对于在$T_{开始}$之前处于预先状态的节点而言，它们有可能在过程中损坏BFT协议和$T_{邮票}$之前。 重要的是，对于任何尚未损坏的节点，我们的所有安全属性（包括一致性和活性属性）必须在$T_{邮票}$之前保留。 相比之下，静态概念不需要将安全保证扩展到预先破坏的节点。 事实上，我们不难发现，我们的“稍微静态”的安全概念严格地比标准的“静态”安全概念更强大，并且构建一个可能表现出这种分离的（可能是人为的）BFT协议并不困难。

定义的微妙之处：BFT的强大安全性。 我们认为，由于与敌对选择性开放攻击相关的技术细节，我们需要为BFT子协议定义上述更强的安全概念。 下面我们将这个概念与最自然的基于财产的安全概念进行比较。

首先，尽管协议的大多数其他安全定义遵循最自然的基于属性的定义风格，但BFT构件块的上述安全性概念更强。 特别是，如果BFT满足上述强大的安全性概念，那么一个真实的实例化，其中诚实节点现在生成它们自己的签名密钥对并实现它们自己的签名神谕，将满足
最自然的基于财产的概念。 当然，为了使描述完整，在这种情况下有效的环境Z将等待听到每个诚实的BFT节点输出公钥pk_i，并且

然后输入start（comm）给所有诚实节点，其中comm包含足够多的诚实节点的公钥。

当诚实节点实现自己的密钥生成和签名神谕时，我们可以将诚实节点的密钥生成和签名oracle实现分组到环境Z*中。 然后，
这个特定的Z*绝不会公开真实的节点各自的秘密签名密钥。 因此，如果有的话
存在一些对手ppt A，使得exec［BFT］（A，Z*，λ）以不可忽略的概率不符合任何这些性质，则我们可以构造对手B，使得在交互期间exec［BFT］（ A，Z*，λ），B有效地打破了签名方案的安全性。

其次，我们指出，基于自然属性的定义比较薄弱，不足以达到我们的目的。 特别是在后来的HybridConsensus协议中，用于BFT的环境Z可以选择性地打开一组公钥以包含在BFT协议的启动命令中。 对于
例如，人们可以很容易想象一个有点人为的BFT协议，在最自然的基于属性的定义下（如本文中所有其他定义一样），它将是安全的，但会容易受到选择性开放式攻击的影响：想象一下，诚实节点公开他们的秘密签名如果对所选公钥的某些谓词满足 - 这个谓词可以很容易地被选择，使得它对于一个诚实生成的公开密钥集合只有negl（λ）概率满足，而不受对抗选择性开放的影响，但对压倒性的财产感到满意在敌对选择性开放下（例如，如果所有公钥以1结束）。

幸运的是，不难看出许多已知许可的BFT协议实例满足这种强大的安全性概念，例如PBFT [21]与数字签名。

关于签署预言的说明。 我们注意到，或者，可以将签署预言划分为采用GUC方法的全局签名功能[17]. 特别是，GUC是必要的，因为相同的签名方案由多个协议，内部BFT协议和我们的外部DailyBFT协议共享。 如果我们采用GUC方法，那么我们的黑箱减少安全概念也可能更简单，因为我们不需要处理具有签名密钥的环境。 另一方面，使用GUC可能会在符号方面引入其他复杂性。这两种方法基本上是通过重新划分算法边界来实现的。

定理6（卡斯特罗和利斯科夫 [21],在附录中简要描述 A)。 *存在BFT协议*
这对（1-Q）< 1 3 *活性参数T*藩福风级*：= O（nδ）。*

为了达到上述目的，我们可以修改PBFT的指数超时策略，使节点在每n次视图更改时加倍超时。 为了完整性，在附录中 A,我们简要描述PBFT协议，并且我们将读者引用到Castro和Liskov [21] 了解更多细节和优化。 请注意，稍后当我们将BFT用作混合共识中的子协议时，BFT节点n的数量将由csize：=λ替代。

5 正式计划：Nakamoto的混合共识

5.1 符号公约

正式框架的选择。 我们使用广受欢迎的Universal Composition [16,17,20]用于形式化和模块化组成协议的框架。 为了展示我们的建筑，我们将采用模块化方法。 对于每个（子）协议，我们正式描述其抽象 - 不是通过定义理想的功能，而是使用基于属性的方法。 然后我们将展示如何编写这些子协议以最终构建我们的HybridConsensus协议。

会话标识符约定。 对于任何协议prot，如果我们编写prot [sid]，那么sid（或方括号中的任何变量）表示协议实例的会话标识符。 如果我们只在没有方括号的情况下编写prot，那么这意味着我们只关心协议的一个特定会话（尽管更高级别的协议可以调用多个会话），因此我们不明确指示会话标识符。

5.2 日常离线共识协议

对于模块化协议组合，我们定义一个称为DailyBFT的日常离线共识协议的中间抽象。 在DailyBFT中，委员会成员运行偏离BFT的实例来决定每日日志，而非成员计算委员会成员的签名。

DailyBFT概述。 DailyBFT中间抽象的定义通过以下方式扩展了BFT:

- *将*安全性扩展到委员会非成员和后期产卵节点。 在定义层面上，DailyBFT定义扩展了BFT的定义，以包含委员会非成员。 在
  特别是在下面的DailyBFT的正式定义中，所有的安全属性不仅必须由委员会成员来满足，也应该由委员会非成员来满足。 此外，尽管BFT定义假设所有节点都是在$T_{开始}$之前生成的，但DailyBFT的定义允许节点稍后产生。 因此，这里我们的安全性定义包括一致性和活跃性适用于足够早产生的任何节点（委员会成员或非成员一致），即在截止日期$T_{邮票}$之前。 这些安全保证不会延伸到产生得太晚的节点，因为委员会成员在将来可能变得腐败，此时他们可以签署任意元组。 正是由于这个原因，我们的混合共识协议（以DailyBFT作为构建块）需要明确处理后期产卵，以便将安全保证延伸到产晚的节点。

- 终止。 DailyBFT明确规定终止程序必须满足两个要求，即终止协议和及时终止。 具体而言，允许环境Z向节点发送停止指令。 及时终止需要BFT实例
  诚实的节点收到输入停止后很快终止。 终止协议要求所有诚实的节点在终止时达成相同的最终日志。

- 签署每日日志哈希。 在DailyBFT中，委员会成员输出签署的每日日志哈希值，稍后将由混合共识协议消耗。 这些签署的每日日志哈希满足
  *完*整性和不可伪造性。 完整性表示，诚实的委员会成员输出正确签名的日志日志。 不可伪造性说，除了正确的散列之外，环境/攻击者不能伪造任何其他值的签名。

形式上，假设一个DailyBFT $[TF]^{D}$协议，其中R是会话标识符（也称为日期），并且由分发D参数化，提供以下抽象。

输入。 在每个时间步骤中，环境Z可以多次提供以下类型的输入：1）keygen； 2）start（comm）其中$comm = \{pk_i\}_{i\in[n]}$； 3）TXs；和4）停止。

输出。 诚实节点向Z输出以下类型的消息:

- 在输入keygen上，诚实节点输出$pk \leftarrow D$。

- 在每个时间步t中，诚实节点输出到环境notdone（log），直到最后一步t^*，它输出done（log^*，recs），其中recs是一组签名元组为最后的每日日志的散列。 输出完成后（log^*，recs），诚实节点停止输出 ∅
在未来的时间步骤。

术语。 假设在支持exec [DailyBFT]（A，Z，λ）的特定视图中，环境Z向所有诚实的节点输入一个唯一的开始（comm）命令 - 稍后我们的遵从性规则将要求这种情况，那么comm: = $\{pk_i\}_i$被称为选举委员会。
我们说，如果以下情况成立，节点i在时间t是诚实的委员会成员：

- 在第一次启动（comm）命令输入到任何诚实节点之前，节点i输出到Z a pk
包含在comm中。

- 节点我保持诚实，直到时间t（但可能会在后来腐败）。
从今以后，如果我们说“一个诚实的委员会成员我执行某种行动或者是接受者的行为”
在某种意义上t时刻的某种行为“，我们含蓄地指节点i在t时刻是一个诚实的委员会成员，即它在t时刻保持诚实，但可能在晚些时候腐败。

$T_{开始}$表示诚实委员会成员收到输入信息的最早时间。
$T_{停止}$表示诚实的委员会成员收到输入停止的最早时间。
我们说一个诚实的节点输出日志作为一个简写，意味着它输出完成（日志）或非日志（日志）。 当一个诚实的节点在某个时间输出完成（日志）时，我们说该日志是节点i的最终每日日志。

符合执行。 （A，Z）不仅是（n，δ，τ），我们说一对（A，Z）是（n，Q，δ，τ，$T_{邮票}$，$T_{蒲福风级}$）- 通过定义有效的关于DailyBFT i,但以下情况也成立：

- 委员会协议。 如果诚实节点i在时间t接收到来自Z的输入开始（comm），并且诚实节点j在时间t'从Z接收输入开始（$comm'$），则认为comm = $comm'$。
- 关闭开始和停止。 假设$T_{开始}$是诚实节点接收输入开始（）的最早时间。 那么，对于任何-在时间$T_{开始}+\delta$处诚实的节点，我必须接收到时间$T_{开始}+\delta$的输入开始（）。
类似地，让$T_{停止}$是一个诚实节点接收输入停止的最早时间。 那么，对于任何在时刻$T_{停止}+\delta$我都诚实的节点，我必须接受时间$T_{停止}+\delta$的输入停止。 对于任何在t时刻接收到停止的诚实节点，它必须在$t'<t$时刻开始接收。
- 弹性。 至少必须输出第1Q·| comm | 1 pk_i∈comm的数量，并且早于第1个
通过保持诚实的节点直到$T_{邮票}$，开始任何诚实节点的命令输入。
- 早点停下来。 $T_{停止}+ T_{蒲福风级}+\delta \leqslant T_{邮票}$，其中$T_{停止}$是最早的诚实委员会成员接受投入停止的时间。
- 暂时的静态腐败。 对于任何pk∈comm，如果pk由在$T_{邮票}$之前变得腐败的节点输出，那么“目标损坏”指令必须在$T_{开始}$之前发布。

安全性定义。 如果对于任何n> 0，δ> 0，任何τ≥0，任何$T_{邮票}$> 0，DailyBFT协议被认为对于具有活性参数$T_{蒲福风级}$的（1-Q）（A，Z）即（n，Q，δ，τ，$T_{邮票}$，$T_{蒲福风级}$）- 有效的DailyBFT，存在一个可忽略的函数negl，使得对于每个λ∈N，除了negl（λ）概率，以下属性适用于exec [DailyBFT]（A，Z，λ）：

- 及时终止。 时间终止包括以下内容：

- 任何在时间$T_{停止}+T_{蒲福风级}$中诚实的委员会成员必须按时间$T_{停止}+T_{蒲福风级}$完成（日志）输出。
- 对于任何在$t \geq T_{停止}+T_{蒲福风级}+\delta$时刻坦诚的节点，它必须在t时刻完成输出（log）。

在这两种情况下，当一个诚实的节点输出完成（日志）时，我们将日志称为节点的最终每日日志。

请注意，由于（A，Z）是（n，Q，δ，τ，$T_{邮票}$，$T_{蒲福风级}$）- 对于DailyBFT有效，及时终止意味着以下内容：在$T_{邮票}$之前产生并保留诚实直到$T_{邮票}$必须拥有输出完成（，）$T_{邮票}$。 换句话说，如果任何节点在$T_{邮票}$之前产生并输出（）），诚实时，完成（，）必须输出不迟于$T_{邮票}$。

- 一致性。 一致性包括以下内容：
  - *自洽*。 对于在$T_{邮票}$之前产生并且在时刻t'诚实的任何节点i，假设节点i在时间$t \leqslant t'$输出对数并在时间t输出对数log^t，那个$log \prec log^t$。
  - *终止协议*。 对于在$T_{邮票}$之前生成的任何节点i以及也在$T_{邮票}$之前生成的任何节点j，假设节点i输出完成（log^t），并且节点j输出完成（log^t）在它们变坏之前，它认为$log = log^t$。
  - *通用前缀*。 对于在$T_{邮票}$之前产生的任何节点i，j，假设我是诚实的时间t并在时间t输出日志，并且j在t时刻是诚实的^t，并且在时间t输出日志^t，它认为要么是日志要么是日志^t？日志。

请注意，终止协议和自一致性可能暗示了通用前缀，但请记住，常见前缀必须另外适用于在发生损坏前从未有机会输出done（，）的节点。

- *活跃度*。 假设Z在时间$T_{开始} \leqslant t < T_{停止}-T_{蒲福风级}$中输入TX给诚实委员会成员。 那么，对于在时间$t_{卵} \leqslant T_{邮票}$中生成的任何诚实节点i，如果我在时间$t' \geqslant t+T_{蒲福风级}+\delta$时诚实，那么节点i必须有一些输出日志时间$t^* \leqslant t'$，使$TXs \approx log$。
- *完整性*。 让comm是包含在开始命令输入到诚实节点的唯一集合。 对于每个在某个时刻t由节点输出的$pk \in comm$，并且如果节点i在时间t输出完成（log，recs），则认为有效记录$\{R, hash(log)\}_{PK}-1 \in recs$其中有效性是通过使用pk进行正确的签名验证来定义的。
- *不可伪造性*。 令$t \leq T_{邮票}$，令$pk \in comm$由时刻诚实的节点i输出*吨*。 然后，如果到时间t，攻击者A向环境Z输出有效元组$\{R, h\}_{PK}-1$其中R是当前的DailyBFT实例的会话标识符，那么它必须拥有该节点输出完成（日志）由t和h =散列（日志）。

施工。 我们在图中展示了来自BFT的DailyBFT协议的构造 1.以下是DailyBFT的操作的非正式说明：

- *BFT虚拟节点和委员会的选择性开放*。 DailyBFT节点通过键盘查询将新鲜的公钥输出到其环境中。 然后，当它收到一个开始（comm）命令时，如果comm包含一个或多个自己的公钥，则该节点将被选为委员会成员。 在这种情况下，节点将为属于自己的通信中的每个公钥分配一个BFT虚拟节点。 在这里，委员会由环境通过start（comm）命令选择性地打开，稍后我们的证明需要利用BFT的强大安全性。

Subprotocol DailyBFT [R]

输入init：f：= 0，log：=∅，mykeys：=∅

输入keygen：（pk，sk）$\leftarrow^{\$}\Sigma$.Gen（$1^{\lambda}$），将pk添加到mykeys，输出pk

在输入停止时：对于每个BFT^{PK}虚拟节点分叉：输入TXs：={$\{stop\}_{PK}$-1}到BFT^{PK}

在输入开始（通信）：如果mykeys∩comm / =∅：isMember = true，否则isMember = false

~~案例isMember = true~~

对于每个$PK_{\in}$我的钥匙comm：分叉一个BFT虚拟节点和BFT.start（pk，comm）。
//此后此BFT虚拟节点被表示为BFT^{PK}

每次某些BFT^{PK}虚拟节点输出符号（msg）时：return $\{msg\}_{PK}$-1
//为BFT虚拟节点实现一个签名oracle

在输入TX上：输入TX到分叉的每个BFT虚拟节点设BFT_0表示

第一个这样的BFT虚拟节点分叉

让完整（日志）=真如果日志包含由th：=正确签名的停止 $1^{|comm|}$ / /
3不同的pks在comm

每次接收到开始并且没有完成输出时，步骤t：从BFT_0接收输出日志*
如果完成（日志*），则记录*：=最短的日志前缀*，使得完成（日志*）
对于每个tx∈log* - 不是停止事务的日志：
令f：= f + 1，对于每个pk∈mykeys∩com：gossip {R，f，tx}$_{PK}$-1 log：= log*
如果完成（日志）：调用Finalize；否则输出notdone（log）

最终确定：
记录：=∅，从日志中删除所有停止事务
对于每个pk∈mykeys∩comm：令x：= {R，| log |}$_{PK}$-1，recs：=recs∪{x}，八卦x
输出完成（记录，记录），并在未来的时间步骤停止输出。

案例isMember = false

~~在接收{R，f}$_{PK}$-1或{R，f，tx}$_{PK}$-1：向历史添加消息并检查以下内容：~~

on collect（r，f，tx）和签名st r = R和th：= $1^{|comm|}$ / 3不同的pks
在comm中正确地标记了元组：
如果log [f]没有被设置，让log [f]：= tx

on collect（r，f）和签名st r = R和th：= $1^{|comm|}$ / 3不同的pks
在comm中正确地标记了元组：
等待log [：f]全部填充

输出完成（日志，∅），并在未来的时间步骤停止输出。

每个时间步骤直到完成输出：

让log^t：= log的最长连续前缀，输出notdone（log^t）

图1：日常离线共识协议。由于每个签名密钥都被重新用于内部BFT协议和外部DailyBFT协议，因此我们假设签名算法为内部BFT实例标记每个消息的前缀为“0”，每个消息为外部的DailyBFT，前缀为“1“来避免命名空间冲突。

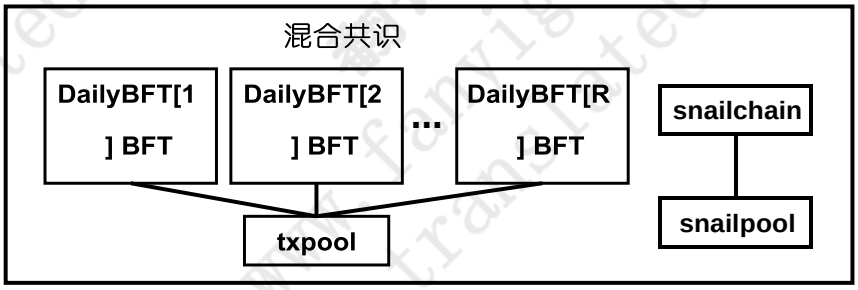

图2：混合共识方案的模块化组成。

子协议mempool

初始化：TXs：=ø
在接收TX_1时：TXs：=TXs⋃Ts_1
在输入提议（TXs_1）上：TXs：=TXs⋃Ts_1，闲聊TXs_1
在输入查询（确认）：返回TXs \确认

图3：mempool子协议记录事务，并在查询后提出一个集合
的未完成交易。 为简单起见，这里没有记载的明显的实际优化是mempool可以清除LOG中已经确认的交易。

- 会员和非会员基本操作。 委员会成员根据BFT协议填写每日日志，而非委员会成员则对委员会的签名进行统计
  成员来填充他们的日志。

- 终止。 节点执行终止程序如下：只要诚实的委员会成员收到停止指令，它就会向每个BFT虚拟节点输入一个特殊的签名停止事务。 只要内部BFT实例输出一个包含停止传输的日志，由至少1个| comm | / 3l不同的委员会公钥签署的行动，日志被确定并输出。 在第一个1 | comm | / 3l停止交易之后的所有交易（具有不同的委员会公钥）将被忽略。

- 签署每日日志哈希。 当委员会成员输出完成时，他们还输出最终每日日志的签名摘要 - 稍后，我们的HybridConsensus协议会将此摘要标记到
  蜗牛。

定理7（来自BFT的DailyBFT）。 假设DailyBFT采用的签名方案Σ是安全，而且这个散列是一个随机预言。 假设BFT对于Q^3具有活性参数T_{BFTT}的（1-Q）腐败是安全的。那么，DailyBFT对于（1 - Q）的腐败是安全的
活性参数$T_{蒲福风级}$：= $T_{BFTT}+\delta$。

这个定理的证明推迟到Section 8.

5.3 混合共识协议

我们现在描述我们的最终产品，即混合共识协议。 杂交共识消耗了DailyBFT的多个实例，其中轮流委员会就每日日志达成一致。 混合共识主要做以下几点：

- 它使用snailchain作为全局时钟有效地管理DailyBFT实例的产卵和终止，从而在诚实节点之间提供弱同步；

Protocol HybridConsensus$^{\lambda}$，通过λ进行参数化

在init：R：= 0，LOG_s：=∅，LOG：=∅，csize：=λ。

Mempools：

- 分叉mempool的一个实例表示snailpool，它存储snailchain的待处理记录。
- 分叉另一个存储待处理事务的mempool实例，表示为txpool。
- 在输入TX上：txpool.propose（TXs）。

Snailchain：在每个时间步中，分叉一个snailchain实例：

- 让链表示当前的本地链
- 让pk：= DailyBFT［R + 1］.keygen其中R表示当前日期
- 让recs：= snailpool.query（ExtractRecs（chain［：-λ］）），输入（recs，pk）给snailchain

预处理：

等到| chain | ≥csize +λ

L：=在历史记录中查找最大的（R，$\log_R$）元组的有序列表，使得R递增（R，hash（$\log_R$））是链上有效的wrt链LOG：= $\log_1$||$\log_2$|| . . . ||日志$_{/L/}$

日常离线共识：

令R：= | L |，派生一个DailyBFT［R + 1］的实例：

Loop：

等到| chain | ≥上（R + 1）+λ，令R：= R + 1

假设$comm_R$：= MinersOf（chain［lower（R）：upper（R）］）其中MinersOf将每个块解析为（$recs_i$，pk_I）并返回包含所有pk_Is

如果存在实例DailyBFT［R-1］，则DailyBFT［R-1］.stop分叉DailyBFT［R + 1］

DailyBFT［R］。开始（$COMM_R$）

每个时间步骤：让TXs：= txpool.query（LOG），将TX输入到DailyBFT［R］

输出：在每个时间步：让R表示当天。 如果DailyBFT［r］在此时间或更早的时间步骤中完成输出，让isdone（r）= true。

- 如果在此时间步中完成DailyBFT［R-1］输出（日志$_{-1}$，录制）：　　LOG_s：= LOG_s|| $\log_{-1}$，snailpool.propose（recs）
- 假设$\log_{-1}$和log分别为该时间步中DailyBFT［R - 1］和DailyBFT［R］的输出日志（或者如果没有输出，则为∅）
- 如果isdone（R-1）：LOG：= LOG_s|| log；否则LOG：= LOG_s|| $\log_{-1}$。 输出LOG

链上有效：如果以下条件成立，则元组（R，h）是链上有效的链条

- 至少对于：1csize / 31不同的pk∈MinersOf（chain［lower（R），upper（R，h}$_{PK}$-1是链中［：-λ］中的第一个出现，其中pk签名了表单的一个元组（R，）.

图4：主HybridConsensus协议。 一个新产生的，诚实的节点开始运行这个协议。 我们假设历史是所有发送和接收的历史成绩单的集合。 我们假设到子协议实例的消息路由是隐含的：每当任何子策略［sid］实例被分支时，历史［subprot［sid］］和与子策略［sid］有关的协议消息将自动路由到子策略［sid］实例。

- 回想一下，每个DailyBFT实例都不能确保产生太晚的节点的安全性，因为委员会成员在将来可能变得腐败得很厉害，他们可以签名
  任意元组。 因此，混合共识引入了链上加盖机制，以便将安全保证延伸到即使晚点产生的节点。

数字 4 是HybridConsensus协议的算法描述。 数字 2 说明了混合共识协议的模块组成。 具体而言，混合共识协议在内部运行以下子协议实例：两个mempool实例分别表示为snailpool和txpool，一个snailchain实例和多个DailyBFT实例。 我们现在更具体地解释这些子协议实例。

交易mempools。 HybridConsensus协议维护mempools协议的两个实例（参见图 3),分别表示为txpool和snailpool。 txpool是一个mempool，用于维护未完成的交易，并通过DailyBFT实例进行确认，并且该snailpool用于维护每日日志摘要在snailchain上加盖。 mempool协议非常简单：只要环境输入新的事务，它就会在网络上闲聊交易。 每当它听到来自网络的交易时，都会将它们保存在mempool中。

snailchain。 HybridConsensus协议内部分叉了一个snailchain实例。 首先，snailchain用于达成委员会的协议，后者将运行离线BFT共识。 该委员会被选为csize的矿工：=λ连续的块。 潜在的snailchain链条质量属性确保足够多的这些矿工诚实足够长。 其次，这个snailchain实例不是用于提交交易，而是用于标记每日日志摘要，以便该协议可以抵御追溯者在将来腐蚀委员会成员的追溯性腐败。

DailyBFT实例。 HybridConsensus协议分派了DailyBFT协议的多个实例，我们使用索引R来表示每个实例的会话标识符。 R也被称为日数，因此每个DailyBFT [R]实例输出“日志”。 在每个DailyBFT实例中，当选的委员会成员依赖基础的BFT协议进行交易并输出每日日志，而委员会非成员计算来自委员会成员的签名来填充他们的日志。

运营。 每个节点都保留一个历史记录，表示历史记录 - 我们假设这是为了形式化的简单性，并且可以在实践中对其进行优化。 新产生的节点立即获得历史记录（实际上，这可以通过具有诚实的节点提供历史检索服务来实例化）。
当一个新节点出现时，它会按如下方式填充它的LOG：

- *匹*配链上有效的元组。 新产生的节点首先标识形式（R，h）的所有链上有效元组，其中R是日数，h是每日日志的散列值。 然后，
  节点将搜索历史记录并确定与h一致的适当日志日志。 该节点使用这些每日日志填充LOG。 这种链上匹配过程为新产生的节点有效地提供了一种安全的机制来追踪和填充其输出LOG的旧条目。
- *通*过每日链接的共识。 一旦这个追赶过程完成，节点将从此依靠DailyBFT实例来进一步填充其输出LOG的剩余条目。 在
  每个DailyBFT实例，节点都可以充当委员会非委员的委员会成员。

要做到这一点，一个节点会监视snailchain实例的输出链。 一旦链条
长度超过csize·R +λ，则第R日开始，此时节点输入停止到前一个DailyBFT [R-1]实例（如果存在），并且输入start（MinersOf（chain [lower（R） : upper（R）]））到DailyBFT [R]实例。 在此期间通常会有一段时间的重叠
DailyBFT [R-1]和DailyBFT [R]实例同时运行并输出其各自的日志。 当节点将日常日志吸收到最终输出LOG中时，
他们确保LOG始终是连续的，不会留下间隙。 由于DailyBFT的及时终止财产，旧的DailyBFT [R - 1]将很快终止，此时新的DailyBFT [R]实例将完全接管。

5.4 定理陈述

定义6（混合共有Γ^{HC}_{ρ}的容许参数）。 令$T_{蒲福风级}$：= 0（m δ）为具有m个节点的BFT的活性参数。 如果以下成立，我们定义Γ^{HC}_{ρ}（n， α， δ， τ）= 1。

- n> 0，δ> 0，τ≥0都是λ中的多项式函数； α> 0是常数；
- 存在一个常数η> 0，使得p（1-（2δ+ 2）p）≥（1 +η）q。 （这是底层snailchain安全所需的。）
- 存在一个常数η_0> 0，使得Q：= 1-（1 +η_0）$^{-\frac{q}{\gamma}}$ > $\frac{2}{3}$。 （这是我们需要的获得超过2/3链条质量的蜗牛。）
- 存在一个常数η_1> 0，使得G^t：=（1 +η_1）n ρ < $\frac{\lambda}{T_{蒲福风级}}$ 对于任何λ∈N（这是需要这样的链条不增长太快，以确保活力。）
- 对于一些适当的大常数c，存在一个常数η_2> 0，使得τ >4 λ（1 +η_2）/ γ +c λ δ。 （这是需要的，以便对手的灵活性受到足够的限制。）

在上面，参数p，q，γ是在n，α，δ中的函数，如在Section中定义的 4.1.

定理8（混合共识的主要定理）。 假设散列H：{0,1}*→{0,1}$^{\lambda}$是独立的随机过程，且签名方案Σ是安全的。 那么，对于任何常数η> 0，与Nakamoto作为snailchain和采矿难度参数ρ实例化的混合共识
（$T_{暖身}$， $T_{确认}$）与任何ppt Γ^{HC}_{ρ} - 可允许的（A，Z）是安全的，其中

$$T_{暖身}：=2\lambda（1+\eta）/\gamma，T_{确认}：=0（\lambda\delta）$$

请注意，上面的$T_{确认}$参数是最坏的情况，在乐观情况下，混合共识达到的交易确认时间为0（δ）。

上述定理的证明将在Section中介绍 8.

典型的参数化。 通常在实践中，如果我们将难题的难度参数ρ：=Θ（^{1}）设置得足够小，其中Δ是网络延迟上可能存在的松散上界，其先验已知。 在这种参数化下，如果整体腐败率α大约为3/4，那么我们确保大约2/3的链质量。

推论3（关于Nakamoto的混合共识的典型参数：定理的重述??）。 假设对于任意常数E> 0，$\alpha=\frac{1}{2}E$。 那么对于每一个n，δ，存在足够小的ρ_0：=Θ（$\frac{1}{\delta n}$），使得以Nakamoto为基础的混合共识

o对*任意τ的任何ppt（n，α，δ，τ）－有效（A，Z）是安全的，使得*Γ^{HC}*（n，α，δ，τ）= 1，实现*

$$\rho$$

$$T_{暖身}: =8\lambda/3n\rho, \quad T_{确认}: = O(\lambda\delta)$$

5.5 实际考虑和可能的优化

上面描述的方案为了简单起见进行了优化，并且有助于形式分析，但不适用于实际性能。 有很多可能的优化。 例如，与其让委员会成员逐一签署每笔交易，他们可以一次签署一批交易。 在打印过程中，我们可以依靠阈值签名方案并将每日日志散列的单个签名印在snailchain上，而不是让每个节点向snailchain发送单独的签名。 修剪旧成绩单的存储也很容易。 我们将实际优化和实施留给未来的工作。

6 延伸：果实链作为底层的蜗牛链

Pass和Shi最近提出了一个名为Fruitchain的新区块链协议 [48]. 充分对于任意小的常数E> 0，小难题难度参数p_{nb}: =Θ（1）和$_3$α: =1-E，果实链在连续块的任何足够大的窗口上实现Q>^{2}链质量。 构建混合共识和Fruitchains以获得响应性的无许可协商一致性协议并不难，该协议能够抵御1/3-E整体腐败的任意小常量*E* > 0.

果实链提供了与Nakamoto共识相同的正式抽象，但具有不同的参数。 今后，我们将使用术语“果实质量”来表示果实链的质量，“果实生长”则意味着果实链的增长。 在章节中陈述的典型参数下4.1.2 和推论 2，为了获得η-最优链条质量，Fruitchain要求果实质量窗口合理地大，即Θ（λ/η）。同样，水果生长也需要时间窗口相当长。 鉴于此，当我们采用Fruitchain作为底层snailchain时，我们需要对Section中描述的协议进行如下修改 5:

$$\lambda, \eta$$

- 让HybridConsensus 用参数λ和η进行参数化。
- 重新定义csize: =$\overline{\eta}^{\lambda}$，lower（R）：$_{\eta}$=$^{(R-1)\lambda}$+1，upper（R）: =$\eta^{\bullet R\cdot\lambda}$。 换句话说，委员会大小被设定为$_{\eta}^{\lambda}$，协议等待链长| ≥上（R）+λ开始*第*R天。

定义7（通过果蝇$\Gamma^{hcfruit}$进行杂交共有的允许参数）。 设$T_{蒲福风级}$: =

$$\rho$$

0（mδ）是具有m个节点的BFT的活性参数。 我们定义$\Gamma^{hcfruit}$*（n，α，δ，τ）= 1以下是：*

- *n*> 0，δ> 0，τ≥0都是λ中的多项式函数； α> 0是常数；
- 存在一个常数η^{t}> 0，使得p（1-（2δ+ 2）p）≥（1 +η^{t}）q。 （这是底层snailchain安全所需的。）
- *Q*: =（1-5n）（1-α）>$_3^2$。 （这是需要的，以便我们可以获得超过2/3链条质量的蜗牛。）
- *G**: =（1 +5η）nβ<$T_{蒲福风级}$ 对于任何λ∈N（这是需要的，以使链不增长

$$\lambda$$

太快以至于不能保证活力。）

- *对于一些适当的大常数c，τ>3λ（1 + $\frac{1}{\eta}$）/（（1-5n）（1- α）n ρ）+c λ δ。* （这是需要的，以便对手的灵活性受到足够的限制。）

在上面，参数p，q，γ是在n，α，δ中的函数，如在Section中定义的 4.1.

定理9（关于果实链的混合共识：定理的重述 3)。 *对于任意（任意小）常数E> 0，令α=1-E，并且对于每个n， δ存在足够小的ρ=: =Θ（1），合适的κ=Θ（λ）和常数η> 0，使得HybridConsensus超过了Fruitchain*

参数（ρ，κ）对任何合适的τ都是安全的，任何ppt（n，α，δ，τ）- 有效（A，Z）

$\Gamma^{\text{fruit}}_{\rho,\eta}(n, \alpha, \delta, \tau)$

$$T_{\text{暖身}}: =1.5\lambda\ (1+\frac{1}{\eta})\ /\ (1-5\eta)\ n\rho, \qquad T_{\text{确认}}: = 0\ (\lambda\ \delta)$$

再次，上面的$T_{\text{确认}}$= 0（λ δ）是最差的交易确认时间（即使在受到攻击时）。 乐观的交易确认时间是0（δ），即与安全参数λ无关。

7 证明路线图

在提交详细的证明之前，我们首先描述一个高层次的路线图以帮助理解。 为了简单起见，我们使用Nakamoto的混合共识作为我们描述的一个例子，因为除了不同的参数外，对于Fruitchain的混合共识的证明是相同的。

DailyBFT的HybridConsensus。 我们的证明将按照以下步骤进行。

1. 首先，主要在引理 2,引理 3,和事实 4,我们证明当作为HybridConsensus的子协议执行时，DailyBFT的所有实例都具有绝对可能的有效环境。 一旦我们证明了这一点，我们就可以在以后的证明中依靠DailyBFT的安全属性。

2. 接下来，我们证明了一对引理$T_{\text{邮票}}$（R）成为每一天R的“最后期限”。所有诚实的委员会成员的行为将在时间$T_{\text{邮票}}$（R）下完成并生效。

   引理 4 粗略地说，R日的所有诚实委员会成员都会在时间$T_{\text{邮票}}$（R）上盖上正确签署的每日散列给snailchain。 事实 5 粗略地说，产生于$T_{\text{邮票}}$（R）之后的节点不会创建DailyBFT（R）实例，但将依赖链上加盖的每日日志哈希来决定第R日的日志。

3. 已经建立$T_{\text{邮票}}$（R）作为R日的最后期限，然后我们使用以下策略证明一致性（引理 5 和定理 10):

   - 对于实际创建DailyBFT（R）实例的节点，我们知道它们必须在$T_{\text{邮票}}$（R）之前产生。 因此，我们依靠DailyBFT（R）的属性来证明这种一致性
     节点（委员或非会员）。
   - 对于没有生成DailyBFT（R）实例的节点，我们表明如果他们通过检查snailchain上的内容来恢复日常日志，他们也会满足一致性。
     直觉上，委员会成员总是在他们变得腐败之前在snailchain上正确签署日志记录。 因此，即使它们稍后变得腐败，并且此后可以将任意事物印记到snailchain上，但由于诚实节点仅识别委员会成员公钥的第一个加盖的日志散列值，所以它将会太晚。

4. 最后，我们证明了混合共识的生动性（定理 11）大致如下。 非正式地说，假设在R天（大致说来），环境将tx输入到一个诚实的节点。 有两种情况：1）在R天足够早地提出tx，使得DailyBFT [R]实例的活性特性适用；和2）由于tx在非常接近于一天结束时关闭，所以提议tx太迟而不能被纳入R日志。 在这种情况下，tx将被回滚到R + 1日。直观地说，由于DailyBFT [R]实例将很快终止，并且由DailyBFT [R + 1]的活性属性，在tx之前也不会花太长时间并入日R + 1的日志。

   但是，以上仅适用于实际生成DailyBFT [R]（或DailyBFT [R + 1]）实例的节点。 对于任何加入太迟且没有生成DailyBFT [R]（或DailyBFT [R + 1]）的节点，它将在处理历史记录后产生一天R的日志。 现在，通过一致性属性，我们知道无论日后记录晚节点输出，它都会包含tx。

来自BFT的DailyBFT。 这个证明的最技术部分涉及证明以下内容。 请注意，当BFT作为DailyBFT内部的子协议运行时，BFT感知的环境部分由DailyBFT协议指定。 回想一下，BFT的环境需要为BFT实现一个签名oracle。 当作为一个子协议在DailyBFT内部运行时，签署oracle是通过DailyBFT协议实现的。 根据诚实的DailyBFT协议的定义，诚实的节点永远不会公开他们的签名密钥。 对于这样的环境（BFT），如果安全
在Section中指定的属性 4.2 可以被ppt攻击者A破坏，我们可以构造一个减少签名安全性的Re。

上述基本上允许我们证明，当BFT在DailyBFT内部作为子协议实例运行时，BFT的环境很好，因此BFT的所有声明安全属性将保持不变，除非忽略概率。 此证明的其余部分此后依赖于BFT的这些属性来作出论证。

8 详细的证明

下面我们介绍我们对Nakamoto的混合共识的证明，除了改变参数之外，对于Fruitchain的混合共识的证明是相同的。

8.1 术语和简单事实

子协议的环境。 在执行与A，Z的协议HybridConsensus协议时，让 (A，Z) [subprot [sid]]成为子协议实例子协议[sid]与之对接的对手/环境对。 (A，Z) [subprot [sid]]由 (A，Z) 和HybridConsensus outer的部分定义为subprot [sid]。 我们还使用符号Z [subprot [sid]]来表示子协议实例subprot [sid]与之对接的环境。 回想一下图 2 在部分 5.3 说明了模块化
我们的HybridConsensus协议的组成。

下面的事实说，如果DailyBFT的环境输入一些start (comm)，其中comm包含节点i的输出，那么在时间t≥T时TF是真实的$_{开始}$，那么节点i必须在T$_{开始}$之前具有输出pk。 换句话说，DailyBFT的环境无法预测诚实节点输出的未来pk对。 这个简单的事实在整个过程中都很方便，因为每当我们说一些pk∈comm由一个诚实的节点i输出时，这个诚实的节点隐含地是一个委员会成员。 回想一下，根据定义，对于节点i被认为是委员会成员，它必须在T之前输出一些pk∈comm$_{开始}$。

事实1（公钥的不可预测性）。 假设签名方案是安全的，它必须保证对任何ppt (A, Z) 和任何λ∈N，以下属性适用于exec [DailyBFT] (A, Z, λ)，
与1 - negl (λ) 概率：
如果Z输入到任何诚实的节点start (comm)，并且让我是在t≥$T_{开始}$时刻诚实的节点并且此外输出pk∈com，那么它认为节点i是诚实的委员会成员wrt DailyBFT在t的时刻，即上述pk∈comm必须在Tstart之前由i输出。

证明。 简单来说，安全签名方案必须具有高熵公钥。

事实2.对于一个安全的签名方案，多项式中许多诚实生成的公钥不会与1 - negl (λ) 概率相冲突。

因此，今后我们简单地假设诚实节点生成的签名公钥不会相互冲突。

8.2 混合共识证明

值得注意的事件。 给定引理 2，我们知道对于每个多项式有界的R∈N， (A, Z) [DailyBFT [R]]尊重委员会的一致性。 因此，对于协议实例DailyBFT [R]，诚实的委员会成员的概念是明确的。 为了方便，我们明确表示
为重要事件定义以下时间。

- $T_{开始}$ (R) ：诚实委员会成员投入其最早时间开始进入其DailyBFT [R]实例的最早时间；
- $T_{停止}$ (R) ：诚实委员会成员投入停止其最新的DailyBFT [R]实例的最早时间。
- $T_{邮票}$ (R) ：= $T_{开始}$ (R) + λ / G + $T_{蒲福风级}$+δ+ $T_{蜗牛}$。

引理2 (DailyBFT的环境满足委员会协议，关闭开始和停止，以及临时静态腐败)。 令 (A, Z) 为任意n∈N，α＞ 0，δ＞ 0和τ＞3λ／ G + $T_{蒲福风级}$+ δ + T (n, α, δ, τ) 的有效wrt HybridConsensus TF1495)，使得snailchain满足一致性，Q链质量和 (G, G^t) - 链增长wrt (A, Z) [snailchain]。 那么，对于任何λ∈N，下面的属性
以1 - negl (λ) 概率执行exec [HybridConsensus] (A, Z, λ)：
对于任何R = poly (λ) ∈N， (A, Z) [DailyBFT [R]]满足委员会协议，关闭启动和停止以及临时静态腐败。

证明。 从混合共识的定义中直接得出委员会协议
并从snailchain的一致性属性。
从HybridConsensus的定义直接开始和停止
并从snailchain的一致长度属性（这是链增长的一部分）。
从$T_{邮票}$ (R) 的定义和潜在的τ -agile腐败模型中可以看出，暂时的静态腐败是直截了当的。

事实3 (有限的一天的长度)。 假设 (A, Z) 对于任意n∈N，α＞ 0，δ＞ 0且τ≥0使得混合一致性 (n, α, δ, τ) 有效，使得snailchain满足一致性，Q链质量和 (G, G^t) - 链增长wrt (A, Z) [snailchain]。 那么，对于任何λ∈N，下面对于具有1 - negl (λ) 概率的exec [HybridConsensus] (A, Z, λ)

$$2\lambda / G^{t} \leqslant T_{\text{开始}}(1) \leqslant 2\lambda / G,$$

$$(R) = T_{\text{开始}}(R+1) \leqslant T_{\text{开始}}(R) = poly\ poly\ poly\ poly + \lambda / G$$

证明。 对于满足上述要求的 (A, Z) 对，对于任意λ∈N，exec [HybridConsensus] (A, Z, λ) 的下列表达式具有1 - negl (λ) 概率。

$T_{\text{停止}}(R) = T_{\text{开始}}(R+1)$ 从定义直接出现的事实
诚实的HybridConsensus协议。 此外，通过对诚实HybridConsensus的定义
协议，当其本地链长度达到csizeR +λ时，诚实节点向DailyBFT [R-1]发送停止。 通过 (G, G^{t}) 链增长，得出$T_{\text{开始}}(R) + \lambda / G^{t} \leqslant T_{\text{停止}}(R) = T_{\text{开始}}(R+1) \leqslant T_{\text{开始}}(R) + csize / G = T_{\text{开始}}(R) + \lambda / G$。 同样，通过 (G, G^{t}) 链增长，认为$2\lambda / G^{t} \leqslant T_{\text{开始}}(1) \leqslant 2\lambda / G$。

今后，我们将假设exec [HybridConsensus] (A, Z, λ) 将声明在事实中声明的不良事件 1，引理 2，和事实 3. 如果发生这种不良事件，则执行中止。
由于所有这些不良事件都是以negl (λ) 概率出现的，因此断言不良事件的新执行在计算上难以区别于满足事实中指定条件的 (A, Z) 对 1，引理 2，和事实 3.

特别是，由于我们假设委员会达成了一致意见，今后我们将用任何具体的观点来支持执行委员 [HybridConsensus] (A, Z, λ)，使用符号$comm_{R}$表示全球议定的第R天委员会）。

引*理3 (足够多的通讯R成员保持诚实，直到$T_{\text{邮票}}(R)$)。 令Q，G，G^{t}为λ，n，α，δ中的多项式有界函数。 对于任何$T_{\text{蒲福风级}} > 0$和任何常数η > 0，令$T_{\text{蜗牛}} := (1+\eta)\lambda / G$，令 (A, Z) 为 (n, α, δ, τ) 对于任何n∈N，α > 0，δ > 0和τ > 3λ / G + $T_{\text{蒲福风级}}$+ δ + $T_{\text{蜗牛}}$，对于任意n∈N的混合共识使得snailchain满足一致性，Q-链质量和 (G, G^{t}) - 链增长 (A, Z) [snailchain]。 那么，对于任何λ∈N，下面的属性都成立*
对于具有1-negl (λ) 概率的exec [HybridConsensus] (A, Z, λ)

对于任何R = poly (λ) ∈N，$comm_{R}$的至少Q部分由保持诚实的节点输出
直到

$$T_{\text{邮票}}(R) := T_{\text{开始}}(R) + \lambda / G + T_{\text{蒲福风级}} + \delta + T_{\text{蜗牛}},$$

证明。 对于满足上述要求的 (A, Z) 对，对于任意λ∈N，exec [HybridConsensus] (A, Z, λ) 的下列表达式具有1 - negl (λ) 概率。

让链在任何时候都是一个诚实的节点的本地链 ≥上 (R)。 通过Q链
质量并且因为csize =λ，所以链[下 (R)，上 (R)]的至少Q-部分是完整的块wrt
链[: lower (R) - 1]。 这意味着对于至少Q个指数i∈[lower (R)，upper (R)]的部分，存在节点j，该节点j在某个较早的时间t是完整的，从而它输出到Z [snailchain]链t在时间t-1包含前缀链[: lower (R) - 1]，Z [snailchain]在时间t为节点j提供输入链[i]: = (recs, pk)。 根据HybridConsensus协议的定义，pk $comm_{R}$
并且pk必须是时间t处的某个DailyBFT [R]节点j的实例的输出。 现在由于 (G, G^{t}) 链增长，至多 (csize +λ) / G =2λ / G时间t和T之间$_{\text{开始}}(R)$。 最后，由于
r> (3λ/ G + $T_{\text{蒲福风级}}$+ δ + $T_{\text{蜗牛}}$) - 敏捷性，我们有节点j保持诚实直到$T_{\text{邮票}}(R)$。 因此，至少Q的部分$comm_{R}$由$T_{\text{邮票}}(R)$保持诚实的节点输出。

接下来，我们证明$T_{\text{邮票}}(R)$被选择得足够远以便为所有通信R成员的DailyBFT实例提供足够的时间来输出彼此一致的最终每日日志（可能为空）。

事实4 (DailyBFT的环境是有效的。)。 设Q, G, G为λ, n, α, δ中的多项式有界函数。 对于任何$T_{蒲福风级}$> 0和任何常数η> 0, 令$T_{蜗牛}$: = (1 +η) λ / G, 令 (A, Z) 为 (n, α, δ, τ) - 有效对于任意n∈N, α> 0, δ> 0和τ> 3λ / G + $T_{蒲福风级}$+δ+ $T_{蜗牛}$的任意n个混合共识, 使得snailchain满足一致性, Q链质量和 (G, G') - 链增长 (A, Z) [snailchain]。

那么, 对于任何λ∈N, 具有1-negl (λ) 概率, 以下适用于exec [HybridConsensus] (A, Z, λ):

(A, Z) [DailyBFT [R]]是 (n, Q, δ, τ, $T_{邮票}$ (R) , $T_{蒲福风级}$) - 有效的DailyBFT。

证明。 通过组合Lemmas直接简单 2,3 和事实 3.

引理4 (及时加盖)。 令G, G', Q为λ, n, α, δ中的多项式有界函数。假设DailyBFT对于具有活性参数$T_{蒲福风级}$的 (1-Q) 腐败是安全的。 对于任何常数η> 0, 令$T_{蜗牛}$: = (1 +η) λ / G。 设 (A, Z) 为 (n, α, δ, τ) - 对于一些n∈N, α> 0, δ> 0和τ> 3λ / G + $T_{蒲福风级}$+δ+ T_TF1604), 使得除了wrt (A, Z) [snailchain]外, snailchain满足一致性, Q链质量和 (G, G') 链增长。 然后, 对于任何λ∈N, 具有1 - negl (λ) 概率, 对于exec [HybridConsensus] (A, Z, λ) 和对于任何R = poly (λ) ∈N, 以下属性成立。

假设任何诚实的节点在t≥$T_{邮票}$ (R) 时输出一个链。 对于任何pk∈com_R由一些在$T_{邮票}$ (R) 处诚实的节点i输出, 让h: = hash (log_R) 其中log_R表示在节点i的DailyBFT [R]实例的最终日志输出中, 然后, 形式{R, h}$_{PK}$-1 (其中有效性由签名验证定义) 的有效记录包含在链[: -λ]中, 并且对于不同的h' / = h, 不存在形式{R, h'}$_{PK}$-1的任何其他有效记录。

证明。 对于满足上述要求的 (A, Z) 对, 对于任意λ∈N, exec [HybridConsensus] (A, Z, λ) 的下列表达式具有1 - negl (λ) 概率。

由于事实 4, (A, Z) [DailyBFT [R]]是 (n, α, δ, τ, Q, $T_{邮票}$ (R) , $T_{蒲福风级}$) - 有效的DailyBFT。 由事实 1,对于在时间$T_{邮票}$ (R) 处诚实的节点i输出的这样的pk∈com_R, 节点i是对于DailyBFT [R]的诚实委员会成员。 因此, 通过及时终止DailyBFT, 节点我的DailyBFT [R]实例将通过时间$T_{停止}$ (R) + $T_{蒲福风级}$输出完成一些元组 (日志, recs)。 根据诚实HybridConsensus算法的定义, 节点i将通过$T_{停止}$ (R) + $T_{蒲福风级}$调用snailpool.propose (recs); 根据mempool协议的定义, 每个诚实节点的snailpool.TX将包含$T_{停止}$ (R) + $T_{蒲福风级}$+δ的recs。 因此, 从时间$T_{停止}$ (R) + $T_{蒲福风级}$+δ开始, 在每个时间步骤中, Z [snailchain]将输入到snailchain的recs中, 用于每个本地链[: -λ]尚未包含rec的诚实节点。 根据snailchain的活性性质, 在$T_{停止}$ (T) + $T_{蒲福风级}$+δ+ $T_{蜗牛}$之后的任何时刻, recs都会出现在每个诚实节点的本地链[: -λ]中。 完整性属性的DailyBFT协议{R, h}$_{PK}$-1∈recs, 其中h: = hash (log)。 最后, 由于事实 3, $T_{停止}$ (R) - $T_{开始}$ (R) <λ / G。 因此, 认为时间$T_{邮票}$ (R) = $T_{开始}$ (R) +λ / G + $T_{蒲福风级}$+δ+ $T_{蜗牛}$, 每个诚实节点的本地链[: -λ]将包含{R, hash (log) }$_{PK}$-1, 其中log是节点i的DailyBFT [R]实例的最终每日日志输出。

我们现在证明这样一个元组rec = {R, h}$_{PK}$-1是第一个出现在链[: -λ]中的类型, 即没有形式rec'的其他有效记录: = {R, h'}$_{PK}$-1其中h' 在rec rec in之前链。 我们通过矛盾来证明。 假设链中有一个元组rec'在rec [: -λ]之前。 由于rec必须在时间$T_{邮票}$出现在chain [: λ]中, 由于snailchain的一致性, 所以必须rec'。 然而, 通过对诚实的DailyBFT算法的定义, 一个诚实的节点不应该输出具有不同最终日志的两个不同的完成 (,) 消息。 因此, 如果rec'出现在时间$T_{邮票}$的诚实节点链中[: -λ], 这显然违反了DailyBFT的不可伪造性属性。

事实5 (没有DailyBFT的晚期产卵)。 令G, G', Q为λ, n, α, δ中的多项式有界函数。假设DailyBFT对于具有活性参数$T_{蒲福风级}$的 (1-Q) 腐败是安全的。 对于任何

常数η＞0，令$T_{蜗牛}$：＝（1＋η）λ／G。　令（A，Z）为（n，α，δ，τ）－对于某些n∈N，α＞0，δ＞0和τ＞3λ／G＋$T_{蒲福风级}$+δ＋T_{TFI674}）使得snailchain满足一致性，Q链质量和（G，G^t）链增长（A，Z）[snailchain]。　然后，对于任何λ∈N，具有1－negl（λ）概率，对于exec [HybridConsensus]（A，Z，λ）和对于任何R＝poly（λ）∈N，以下属性成立。

如果一个诚实的节点按时间$T_{邮票}$（R）产生，则它只会生成一个DailyBFT [R]实例。

证明。 HybridConsensus算法和引理定义简单明了 4. 二

引理5（追溯一致性）。　令$Q>^2$，设G，G^t为λ，n，α，δ中的多项式有界函数。　假设DailyBFT对存在参数$T_{蒲福风级}$的（1-Q）腐败是安全的。　对于任何常数η＞0，假设$T_{蜗牛}$：＝（1＋η）λ／G，设（n，α，δ，τ）为有效的wrt HybridConsensus对于某个n∈N，α＞0，δ＞0和τ＞3λ／G＋$T_{蒲福风级}$+δ＋$T_{蜗牛}$使得snailchain满足一致性，Q链质量和（G，G^t）链增长A，Z）[snailchain]。　然后，对于任何λ∈N，具有1-negl（λ）概率，对于exec [HybridConsensus]（A，Z，λ）以及对于任何R＝poly（λ）∈N，以下属性成立。

让链表随时表示一个诚实节点的输出。　假设（R，h）是一个链上有效元组链，它认为存在一个在$T_{邮票}$（R）处诚实的节点i输出的pk∈com_R，并且h＝hash（log_R）其中log_R是节点i的（唯一）最终每日日志输出
DailyBFT [R]实例。

证明。 对于满足上述要求的（A，Z）对，对于任意λ∈N，exec [HybridConsensus]（A，Z，λ）的下列表达式具有1－negl（λ）概率。

对于（R，h）是一个有效的链上元组wrt链，至少1csize／3l个pk∈com_R：
1）正确签名的元组{R，h}$_{PK}$-1必须出现在链[：－λ]中；和2）这个元组是链上第一个出现{R，}$_{PK}$-1形式的有效元组。

由事实 4,在时间$T_{邮票}$（R）处诚实的节点输出至少Q部分的通信。
假设$Q>^2$，并且由于鸽子洞原理，对于至少一个由某个公钥pk签署的这种签名，它必须保证它由节点i输出（在$T_{开始}$之前（R）），直到$T_{邮票}$（R）保持诚实。 由引理 4,这个签名必须证明（R，hash（log）），其中log表示节点i输出的（唯一）最终日志。

定理10（混合共识的一致性）。　假设散列H：{0,1}→{0,1}^{λ}是独立的随机过程，且签名方案Σ是安全的。　令Q＞，令G，G为λ，n，α，δ中的多项式有界函数。　假设DailyBFT对（1-Q）是安全的 －
腐败活性参数$T_{蒲福风级}$。　对于任何常数η＞0，令$T_{蜗牛}$：＝（1＋η）λ／G。　让（A，Z）be（n，α，δ，τ）－对于某些n有效的wrt HybridConsensus　N，α＞0，δ＞0和τ＞3λ／G＋$T_{蒲福风级}$+δ＋$T_{蜗牛}$，使得snailchain满足一致性，Q链质量和（G，G^t）链增长（A，Z）[snailchain]。　然后，对于任意λ∈N，用1-negl（λ）概率，exec [HybridConsensus]（A，Z，λ）满足在Section 3.2.*

证明。 自我一致性从HybridConsensus的定义中微不足道。　下面我们重点说明证明通用前缀。
对于满足上述要求的（A，Z）对，
对于任何λ∈N，exec [HybridConsensus]（A，Z，λ）具有1－negl（λ）概率。

根据诚实的HybridConsensus算法的定义，我们可以解析一个诚实的节点的LOG
对于一些R∈N，

$$LOG := log_1 || log_2 || \ . \ . \ . \ || 日志_{R-1} || 日志_R$$

对于其中r∈[R]的每个日志$_r$，它可以是以下情况之一：

1. *每*日离群共识的最终日志。 日志$_r$是DailyBFT [r]实例的最终日志输出。 由事实 5,它认为这个节点在$T_{邮票}$ (r) 之前产生。

2. *非*最终日志的日常离线共识。 $\log_r$包含在DailyBFT [r]实例的输出中，输出采用notdone ($\log_r$) 形式 - 在这种情况下，通过定义诚实的HybridConsensus算法，必须是这种情况该日志$_r$是LOG中包含的最后一个日志日志。 此外，由事实 5,它认为这个节点在$T_{邮票}$ (r) 之前产生。

3. *匹*配链上有效元组。 对于诚实节点的本地链[: -λ]，存在链式有效元组 (r, h)，使得散列 ($\log_r$) = h。

首先请注意，如果一个诚实节点i的DailyBFT [r]实例输出日志$_r$由于1，而一个诚实节点j的DailyBFT [r]实例 (相同或不同)，则输出日志^{TR}为1，保存该日志$_r$=日志^{TR}。 DailyBFT的一致性属性很简单。

其次，请注意，如果一个诚实节点i的DailyBFT [r]实例输出日志$_r$归因于1，并且诚实节点j的DailyBFT [r]实例 (相同或不同)，由于2输出日志^{TR}，则它必须保存该日志^{TR}< $\log_r$。 这紧接着来自DailyBFT的一致性属性。

接下来，如果一个诚实的节点j (相同或不同) 输出由于3而产生的日志^{TR}，那么必须存在一个诚实的节点i，输出由于1而产生的日志$_r$，使得$\log_r$= $\log^{TR}$。 这直接来自引理 5,以及哈希oracle的碰撞概率可以忽略的事实。

证明的其余部分遵循一种简单的方式，通过观察由于事实而导致日长在λ中多项式有界 3;并且对于所有r = poly (λ) ∈N，至少至少有一个 pk∈com$_r$在$T_{邮票}$ (r) 处是诚实的，并且将在时间$T_{邮票}$ (r) 下完成输出 (，)。

定*理11 (混合共识的活力)。 假设散列H: {0,1}*→{0,1}$^\lambda$是独立的随机过程，且签名方案Σ是安全的。 让Q〉，让G是一个函数 在λ，n，α，δ中。 假设DailyBFT对存在参数$T_{蒲福风级}$的 (1-Q) 腐败是安全的。 对于任何常数η > 0，令$T_{蜗牛}$: = (1 +η) λ / G_0。 令 (A，Z) 为 (n，α，δ，τ) - 对于某个n∈N，α> 0，δ> 0和τ>3λ / G + $T_{蒲福风级}$+δ+ $T_{蜗牛}$，使得对于某些G<，snailchain满足一致性，Q链质量和 (G，G') 链增长wrt (A，Z) [snailchain]。 然后，对于任意λ∈N，用1 - negl (λ) 概率，exec [HybridConsensus] (A，Z，λ) sat-*

根据部分定义的活泼性 3.2 参数$T_{暖身}$: =2λ / G，$T_{确认}$: = 2$T_{蒲福风级}$+2δ。

证明。 对于满足上述要求的 (A，Z) 对，对于任意λ∈N，exec [HybridConsensus] (A，Z，λ) 的下列表达式具有1 - negl (λ) 概率。

假设Z在t≥$T_{暖身}$=2λ / G时输入TX到某个诚实节点。 根据定义 txpool协议，在t +δ时刻，所有诚实的节点都有TX ⊆ txpool.TXs。 假设一个诚实的节点i在某个时间t是诚实的^t ≥ t + $T_{确认}$= t + 2$T_{蒲福风级}$+2δ，我们表明节点i的输出时间t的日志$'$必须包含所有的TX。

由事实 3,和G'>^{t}，$T_{开始}$ (1) ⩽2λ / G = $T_{暖身}$的事实。 此外，对于任何R∈N，$T_{开始}$ (R) <$T_{停止}$ (R) -$T_{蒲福风级}$-δ<$T_{开始}$ (R + 1)。 设R是t⩽$T_{停止}$ (R) -$T_{蒲福风级}$ - δ的最小整数。 现在，以下两种情况之一必须是真实的:

- 情况1: t≥$T_{开始}$ (R) = $T_{停止}$ (R-1)。

  在这种情况下，根据HybridConsensus和Fact的定义 4,$_{停止}$ (R) - π_{bft}.中存在一个诚实通信成员，其Z [DailyBFT [R]]将包括TXs \ LOG输入到DailyBFT [R])

- 情况2: $T_{停止}$ (R-1) -$T_{蒲福风级}$-δ<t <$T_{停止}$ (R-1) = $T_{开始}$ (R) TF1832)

在这种情况下，根据HybridConsensus和Fact的定义 4，存在一个诚实通信$_R$成员，其Z [DailyBFT [R]]将通过$T_{开始}$ (R) $<t + T_{蒲福风级}+\delta$ 在其对DailyBFT [R]的输入中包括TXs \ LOG。

因此，无论哪种情况，都存在一位诚实的委员$_R$成员 Z [DailyBFT [R]] 将在时间$t^* < t + T_{蒲福风级}+\delta$ 中包括TX LOG到其对DailyBFT [R]的输入，使得$T_{开始}$ $t^* < T_{停止}$ (R) $\leq T_{蒲福风级}$。 此后让LOG*表示该委员会成员的输出LOG 在时间t^*。

- 我们首先证明，如果节点我曾经分叉过一个DailyBFT实例，并且在t时刻诚实$^{t'} \geq t + 2T_{蒲福风级}+2\delta$，那么节点i在t时刻的输出LOG将包含所有的TX。

  由事实 5，如果节点i分叉了一个DailyBFT [R]实例，则其$t_{卵} \leq T_{邮票}$ (R)。 由DailyBFT协议的活跃性质和事实 4，如果节点i在时间$t' \geq t+ 2T_{蒲福风级}+2\delta$ 时是诚实的，那么通过t'节点i的DailyBFT [R]实例必须具有输出完成 ($\log_R$)，使得TXs \ LOG $^* \subseteq \log_R$。

  现在，对于任何$1 \leq r \leq R-1$，根据诚实的HybridConsensus协议的定义，节点i 没有分叉DailyBFT [r]实例和日志$_r$在预处理期间计算 (参见图 4)；要么 节点我做了一个DailyBFT [r]实例。 在后一种情况下，由于事实，我们知道节点i由$T_{邮票}$ (r) 分叉 5. 由事实 4 以及DailyBFT协议的及时终止属性，如果节点i在时间$t't + 2T_{蒲福风级}+2\delta$ 处诚实，节点i的DailyBFT [r]实例将具有输出完成 到时间t'。

  通过对诚实的HybridConsensus协议的定义，如果节点i在时间$t' > t+ 2T_{蒲福风级}+2\delta$ 是诚实的，则其输出LOG将包含时间t'处的所有TXs \ LOG* 。

- 现在，考虑节点时我没有分叉DailyBFT [R]实例的情况。 在这种情况下，节点i 必须通过匹配链上有效元组来计算日志$_R$，并且在时间$t_{卵}$中输出$\log_R$。

  基于这个事实 4，必须存在至少一个在时间$T_{邮票}$ (R) 处诚实的节点v输出的至少一个$pk \in com_R$，并且由于DailyBFT的及时终止属性，节点v的DailyBFT [R] 实例会在老实的时候输出日志$_R$。 通过DailyBFT的活性属性，节点v输出的最终日志记录$_R$必须包含所有TX \ LOG*。

  由于定理 10，节点i输出的日志$_R$必须与节点 v 的输出相同 DailyBFT [R]实例，因此也必须包含所有TX \ LOG*。

  最后，由于HybridConsensus的一致性属性，对于任何时候都是诚实的节点 $t' > t+ 2T_{蒲福风级}+2\delta > t^*$令LOG为时间t的输出$^{t'}$，则它保持$LOG^* \times LOG$或$LOG \times LOG^*$。 但是，由于TXs \ $LOG^* \subseteq LOG$，它必定是$LOG^* \prec LOG$的情况。 因此我们得出结论$TXs \subseteq LOG$。

8.3 日常离线共识证明

在一些观点中，假设 (A, Z) 是协议DailyBFT的对手/环境对，我们使用符号 (A, Z) [BFT]来表示与BFT子协议实例接口的对手/环境对。 (A, Z) [BFT]由 (A, Z) 以及诚实的DailyBFT协议共同定义。 请注意，在节点i损坏后，Z [BFT]向A [BFT]显示运行在节点i上的所有BFT实例的所有公共/秘密密钥。

引理6 *(在DailyBFT环境中的BFT安全性)。 如果BFT对于 (1-Q) 腐败和活性参数$T_{蒲福风级}$的定义有很强的安全性 5，并且进一步BFT用安全签名方案Σ来实例化，那么对于任何n, δ, $T_{邮票} > 0$, $\tau \geq 0$, 对于任何ppt (A, Z)*

(n, Q, δ, τ, $T_{邮票}$, $T_{蒲福风级}$) - 对于有效的DailyBFT, 存在可忽略的函数negl, 使得对于任何 λ ∈N,

$$\Pr[\text{视图} \leftarrow \text{exec}[\text{DailyBFT}](A, Z, \lambda) : \text{secure}^{Tbft}(\text{view}) = 1] \geqslant 1 - \text{negl}(\lambda)$$

其中安全^{Tbft} (视图) 如在章节中定义 4.2 - 但用"诚实的BFT虚拟节点"取代任何"诚实的节点"。

*证*明。 我们构建一个ppt减少Re如下。

- 减少Re与签名挑战者Σ*相互作用，并获得表示为pk*的签名公钥。 今后我们将pk*称为挑战公钥。
- 减少Re将模拟所有运行DailyBFT的诚实节点。 当一个诚实节点的BFT虚拟节点要求keygen时，Re翻转一个随机硬币，并以概率1 / n返回挑战公钥pk*；否则，减少Re使用诚实算法生成签名密钥对。 因此，除了挑战公钥pk*嵌入的坐标之外，减少Re知道所有秘密签名密钥。 每当一个诚实的人
  节点的BFT虚拟节点发出挑战公钥pk*的符号查询，Re简单地转发
  查询签名挑战者。
- 约简Re与环境Z相互作用。只要Z向真实节点输入任何东西，Re就会简单地将消息转发到其头部模拟的相应诚实节点。 每当一个模拟的诚实节点需要输出任何东西给Z时，减少Re只是转发消息。 同样，减少Re也转发Z和A之间的任何消息。
  现在，如果一个诚实的节点在某个时刻变得腐败，那么Re需要返回节点的私有状态 。 只要腐败节点不是嵌入了pk*的节点，就会减少
  Re可以成功模拟。 如果损坏的节点碰巧是嵌入pk*的地方，那么
  减少只是中止。 很明显，减少不会中止的可能性是
  不可忽略的。

很明显，在减少Re不中止的事件的条件下，减少与 (A, Z) 的接口被相同地分配为DailyBFT协议的实际执行。

不难看出，如果 (A, Z) 是 (n, Q, δ, τ, $T_{邮票}$, $T_{蒲福风级}$) - 对于DailyBFT有效，则 1 - negl (λ) 概率， (A, Z) [BFT]是 (n, Q, δ, τ, $T_{邮票}$) - 有效的BFT。 特别是，使用 Σ.Gen (1) 算法诚实生成的公钥具有negλ (λ) 碰撞概率，因为
否则签名方案很容易被破坏。

为了避免矛盾，假设引理不成立，即存在一些多项式g和ppt (A, Z)，即 (n, Q, δ, τ, $T_{邮票}$, $T_{蒲福风级}$) 无效的Wrt DailyBFT这样的

$$\Pr[\text{view} \leftarrow \text{exec}[\text{DailyBFT}](A, Z, \lambda) : \text{secure}^{Tbft}(\text{view}) / = 1] \geqslant g(\lambda)$$

安全^{Tbft} (视图) 的定义就像在Section中一样 4.2,但是关于DailyBFT内部的虚拟BFT节点。 根据强安全性的定义 (见定义 5),我们知道存在一个
ppt敌手B和多项式g'等

$$\Pr[\text{view} \leftarrow \text{exec}[\text{DailyBFT}](B, Z, \lambda) : \text{伪造}(\text{view}) = 1] \geq g^{t}(\lambda)$$

现在，考虑减少Re与 (B, Z) 交互的执行。 如前所述，只要执行不中止，从 (B, Z) 的角度来看，执行就是一个真正的执行。 因此，以不可忽略的概率，B将输出给Re一些伪造品，并且如果伪造品恰好是对于发生不可忽略的pk
概率，减少Re会发现伪造签名方案。

定理12（来自BFT的DailyBFT，定理的重述 7）。 *假设签名方案*
DailyBFT使用的Σ是安全的，并且该散列是一个随机预言。 假设BFT是安全的
针对Q>²的活性参数T_{BFTT}的抗（1-Q）腐败。 那么，DailyBFT是安全的
针对具有活性参数$T_{蒲福风级}$的（1-Q）腐败：= T_{BFTT}+δ。

*证*明。 对于任何（n，Q，δ，τ，$T_{邮票}$，$T_{蒲福风级}$）有效的任何（A，Z），任何n，δ，$T_{邮票}$> 0，w.r.t. DailyBFT，exec [DailyBFT]（A，Z，λ）具有以下属性，但可忽略的概率除外：

- *及时终止。* 由于BFT的环境遵守紧急停止，所有BFT虚拟节点都通过$T_{停止}$+ δ 接收停止。 通过BFT的活性属性和DailyBFT的定义，所有诚实的委员会成员在$T_{停止}$+ T_{BFTT}+δ= $T_{停止}$+ $T_{蒲福风级}$ TF2001）+ $T_{蒲福风级}$，停止通信中由1 | comm | / 31不同公钥签署的交易。 当这种情况发生时，按照DailyBFT诚信委员会成员八卦签名的散列的定义
  他们的日志，输出完成（，）。-由于至少Q>²
  3 通信的一部分被输出

  在$T_{邮票}$> $T_{停止}$+ $T_{蒲福风级}$之前保持诚实的委员会成员，不难看出，如果
  停止+ $T_{蒲福风级}$+ δ，它将收集足够的签名并且将完成输出（，）。
- *一致性。* - -
  - *自*洽。 根据DailyBFT的定义，自我一致性是显而易见的。
  - *终*止协议。 根据定义，DailyBFT诚实的委员会成员只需输出与内部BFT_0虚拟节点输出的最终日志相同的最终日志（取消停止事务）。 通过基础BFT的一致性属性，如果一个诚实的委员会成员我的BFT在某个时间输出日志$t_{邮票}$，并且诚实的委员会成员j的BFT输出日志^{t}在某个时间t^t<T（TF2020），它必须保存log<log^{t}或
    记录日志^t。
    如果节点i和j在$T_{邮票}$之前产生并且输出完成（，），而它们仍然是诚实的，那么完成（，）必须不晚于$T_{邮票}$输出。 通过诚实的DailyBFT的定义
    算法，它必须是在终止时输出相同的每日日志的情况。
    我们现在将终止协议证明延伸至T之前产生的委员会非成员$_{邮票}$。 这将取决于签名方案的安全性。 假设有一些
    委员会非会员k在$T_{邮票}$之前产生，并且输出完成（log*）在某个时刻是诚实的。 它一定是这种情况

    1. 由于及时终止属性，由时间$T_{邮票}$输出完成（日志*）；
    2. 日志*可以被解析为

       $$log^* = tx^{*1} || tx^{*2} || \ . \ . \ . \ tx^{**}$$

    3. 对于其中i∈[f*]的每个（R，i，tx_i），对于其中R是会话标识符的元组（R，f），节点k已经接收到至少有1个签名的有效签名TF2046）| comm | 1不同的公众
       通讯中的键。
       3

    由于至少有一部分通信是由$T_{邮票}$中诚实的委员会成员输出的，并且通过鸽舍原则，至少有一个有效签名
    每个元组（R，i，tx_i）或元组（R，f*）由节点v输出的pk∈com通过直到$T_{邮票}$保持诚实。 如前所述，诚实的委员会成员输出相同的最终每日日志，以后称为日志。 解析日志：= tx_1|| . . . || tx。
    假设一个矛盾是log*/ = log。 那么必须存在一个元组（R，i，tx_i）或（R，f*），使得tx_i/ = tx_i或f*/ = f。 不失一般性，假设tx_i/ = tx_i（另一个

很相似）。 然后，节点k将通过节点v输出的pk∈com输出的 (R, i, tx_i) 接收至少一个有效签名，该节点v保持诚实直到$T_{邮票}$。 如果存在ppt (A, Z) 使得log*/ = log具有不可忽略的概率，那么我们可以建立一个打破签名方案的减少。 基本上这种减少模拟所有诚实的用户并与之交互 (A, Z) 对。 只要Z输入keygen，它就可以从签名的挑战者中选择嵌入一个pk。 要做到这一点，每次keygen被查询，减少使得一个

随机猜测并决定是否嵌入签名挑战者的PK。 这种猜测对于1 / poly (λ) 概率是正确的 - 如果猜测后来证明是错误的，那么简化会简单地中止。 每当嵌入式PK需要签名时，缩减就会查询签名挑战者。 对于所有其他密钥对，缩减知道相应的秘密

密钥，并且当节点变得腐败时可以将它公开给 (A, Z)。 最后，当 (A, Z) 在这样一个元组{R, i, tx^*_i}$_{PK}$-1上输出一个有效签名时，缩减输出这个伪造。 请注意，减少应该永远不必对签名挑战者进行这样的查询

tx_i/ = tx_i。

- *通用前缀。* 假设委员会成员i和j在时间t和t分别是诚实的^t，并且我在时间t输出日志并且j在时间t输出日志^t。 由于及时的终止属性，必须是t <$T_{邮票}$和t<$T_{邮票}$的情况。 现在，由于

基础BFT的一致性属性和DailyBFT的定义，不难发现log-log^{t}或log^t-log。

我们现在将论点扩大到T之前产生的委员会非成员$_{邮票}$。 这里

我们依靠签名方案的安全性。 假设k是T之前产生的任何委员会非成员$_{邮票}$。 我们认为，如果节点k的输出完成 (log^t)，那么log^{t}必须与诚实的委员会成员的最终日志表示一致 - 正如我们前面所述，所有诚实

委员会成员必须输出相同的最终日志；根据定义，至少有一名委员会成员在T之前保持诚实$_{邮票}$。

如果节点k在诚实时输出notdone (log*)，那么由于及时终止属性和诚实节点在输出done (，) 后不再输出的事实，它必须保证notdone (log (TF2095)) 在$T_{邮票}$之前输出。 现在解析日志*: = log_1|| . . . || log_m，并解析日志: = log_1|| . . . |日志。 那么它必须保证f≥m和log_i= log_i - 后者可以使用与终止协议完全相同的参数类型显示。

- *活跃度。* 根据DailyBFT协议的定义，任何在t时刻向诚实委员会成员输入的TX都会在相同的时间步t输入到内部BFT虚拟节点。 此外，内在

BFT虚拟节点与外部的DailyBFT具有相同的$T_{开始}$和$T_{停止}$，因为诚实的委员会成员只是将启动和停止命令传递给内部的BFT虚拟节点。

根据BFT协议的活性，如果$T_{开始}$≤t<$T_{停止}$-$T_{蒲福风级}$，对于时间t + T的任何委员会成员$_{蒲福风级}$，其内部BFT虚拟节点将输出一个日志包括时间t + T的所有TX$_{蒲福风级}$。 由于t <$T_{停止}$ - $T_{蒲福风级}$，完整 (log) 必须返回false，否则我们可以轻松构造一个破坏签名安全性的对手。

现在考虑一下所有委员会成员在时间t + $T_{蒲福风级}$诚实，每个委员会成员考虑它在时间t + $T_{蒲福风级}$输出的最长日志。 现在取所有这些日志的交集并将其表示为日志^t。 我们认为TXs⊆log^t，并明确完成 (log^t) =

假。 现在，根据DailyBFT协议的定义，对于每个i∈[| log^t|]，对于每个pk∈com由一些诚实的T节点$_{邮票}$输出，它必须保证每个节点在时间上诚实

t + $T_{蒲福风级}$+ δ必须已经接收到有效签名的元组{R, i, log^t[i]}$_{PK}$-1，其中R是DailyBFT的会话标识符。 在$T_{邮票}$> t + $T_{蒲福风级}$+ δ之前，节点保持诚实的通信输出中必须至少有1Q | comm | 1个pks。 由于Q>$\frac{2}{3}$，任何节点在时间上都是诚实的

$t + T_{蒲福风级} + \delta$ 将输出一个对数*st | log*| ≥时间t + $T_{蒲福风级} + \delta$。 通过DailyBFT的一致性，log*[i] = log^t[i]，因此tx∈log*。

- 完整性。 DailyBFT的定义和签名方案的正确性显而易见。
- 不可伪造性。 如果存在ppt (A, Z) 使得不可伪造性可以以不可忽视的概率被破坏，那么我们可以建立一个减少破坏签名方案。 基本上这种减少模拟所有诚实用户并与 (A, Z) 对交互。只要Z输入keygen，它就可以从签名的挑战者中选择嵌入一个pk。 要做到这一点，每一个当keygen被查询时，缩减会进行随机猜测并决定是否嵌入签名挑战者的pk。 这种猜测对于1 / poly (λ) 概率是正确的 - 如果猜测后来证明是错误的，那么简化会简单地中止。 每当嵌入式PK需要签名时，缩减就会查询签名挑战者。 对于所有其他密钥对该减少知道相应的秘密密钥，并且当节点变得腐败时可以将其披露给 (A, Z) 。 此后我们假设嵌入的pk属于节点i。 最后，当 (A, Z) 在这样一个元组{R, h}$_{PK}$-1上输出一个有效签名时，这个缩减输出为伪造。 注意，减少不应该对签名挑战者进行这样的查询 (R, h)，因为根据定义， (A, Z) 对破坏不可伪造性，它必定是节点i没有完成输出的情况日志，) 通过伪造使得h = hash (log) 。

8.4 扩大对Fruitchain混合共识的证明

到目前为止，我们已经完成了对Nakamoto的混合共识的证明。 当我们使用Fruitchain作为潜在的snailchain时，证明几乎是一样的，除了参数必须重新调整。

考虑HybridConsensus$^{\lambda,\rho}$参数化为 η。 换句话说，委员会规模csize被选为csize: = λ / η。在证明中，我们可以插入以下修改后的参数:

- $T_{蜗牛}$: = $(2\lambda + \frac{\lambda}{\eta}) / G$;
- $T_{stamp}(R) := Tstart(R) + \frac{\lambda}{\eta G} + T_{蒲福风级} + \delta + T_{蜗牛}$;
- $\tau_{\eta} + \lambda) / G + T_{蒲福风级} + \delta + T_{蜗牛} = 3\lambda (1 + \frac{1}{\eta}) / G + T_{蒲福风级} + \delta$;
- $T_{暖身}$: = $\lambda (1 + \frac{1}{\eta}) / G$;
- $t_G > \frac{\lambda}{\eta (T_{蒲福风级} + \delta)}$;
- $G = (1-5n) (1-\alpha) n\rho$和$G^t = (1 + 5\eta) n\rho$。

有了这些新的参数，其余的证据就如同Nakamoto的混合共识一样。

9 下界

9.1 工作量证明不能停止

我们现在证明了一个下界，表明任何安全的无权限共识协议必须无限次地调用工作证明，假设没有额外的信任假设。 我们强调，这个下限并不排除依赖额外信任假设的方法，如股权证明 [11,38,49,50].

定理13（任何安全无许可的共识协议必须无限次地调用工作证明。）。 令Γ表示n，α，δ，τ中的任意二元函数，使得对于作为正多项式（在λ中），非负多项式τ和α的一些n，δ的Γ（n，α，δ，τ）= 1 > 1 /聚一些积极的
多项式多项式。 令Π是一个协议，对任何Γ可容许的ppt（A，Z）存在一个多项式函数poly，使得对于每个λ∈N，exec［Π］（A，Z，λ）满足下列属性具有1 / poly（λ）概率：

- 有界的工作证明。 $_{POW}$: = $poly_0$（n， α， δ， λ）
  对于某些多项式$poly_0$；
- 活跃度。 生活在部分定义 3.2 $_{确认}$= $poly_1$（n， α， δ， λ），$T_{暖身}$= $poly_2$（n， α， δ， λ）和$\Pi_{引导}$= poly对于一些非负多项式聚，聚和聚，TF2196）（n， α， δ， λ）。

那么，存在Γ可容许的ppt（A，Z），使得对于任何λ∈N，exec［Π］（A，Z，λ）
不满足某些多项式多项式的概率1 / poly（λ）的一致性。

直觉上，这个定理表明任何无权限的共识协议都可以抵御1 / poly（λ）的腐败部分，即使在同步网络模型中也必须无限制地调用工作证明，并且要抵抗静态腐败。

证明。 令Π是任何协议，使得对于Γ可容许的任何ppt（A^t，Z^t），存在一个多项式函数poly，使得对于每个λ∈N，具有1 / poly（λ）概率， exec［Π］（A^t，Z^t，λ）满足上面定义的有界证明和活性。 我们现在演示如何构建ppt
（A，Z）是Γ可允许的，使得exec［Π］（A，Z，λ）对于某些多项式多项式以1 / poly（λ）概率打破公共前缀性质。

我们考虑一个Γ可容许的对（A，Z），其行为如下。

交易输入。 在时间$T_{暖身}$，Z采样$tx \leftarrow^{\$} \{0,1\}^{\lambda}$，并且向真实节点输入TXs: = {tx}。 除此之外，Z不输入任何其他交易。

真正的执行。 A指示所有腐败节点在实际执行中诚实行事。

模拟执行。 从时间T_{POW}开始，A也模拟头部的想象执行。 为此，A模拟所有诚实节点和环境Z的执行。假设在该模拟执行中，模拟环境Z输入TXs: = {tx^*}其中
$tx^* \leftarrow^{\$} \{0,1\}^{\lambda}$在模拟时间$T_{暖身}$。

网络和腐败。 在真实和模拟执行中，对手A立即传递消息，即在下一个时间步骤内。 环境静态腐败α
节点的一小部分。

晚产卵节点。 假设一个新的节点i在时间t产生$_{卵}$: = max（T_{POW}+ T_{POW}/α，$T_{暖身}$+ $T_{确认}$）+1。此时，攻击者A将使所有模拟的诚实节点与节点i交互，其中所有模拟的诚实节点遵循诚实协议。
每当节点i闲聊一条消息时，消息就会在δ= 1时间内传递给真实节点中的真实节点，并且传递给模拟执行中的诚实节点。

没有一致性。 现在，具有1 / poly（λ）概率，对于真实和模拟执行，有界证明和活性都满足。条件是这些属性满足真实和模拟执行，我们认为一致性不能满足至少1/2的概率。

假设节点的α分数是损坏的，不难看出，在任何时刻$t \geq t_{\text{弱}}-1$，攻击者A都能够输出与真实执行相同分布的模拟执行，因为对手A将有足够的时间来完成所有必要的H查询。

现在，由于活跃性，真正执行中的诚实节点必须输出一个由$t_{\text{弱}}-1$表示的LOG其中$tx \in LOG$和$tx^* \notin LOG$ - 除了实际执行以来的negl（λ）概率在$t_{\text{弱}}$之前无法知道tx^*。 类似地，除了negl（λ）概率，模拟执行中的诚实节点必须通过$t_{\text{弱}}-1$和其中$tx^* \notin LOG$和$tx \notin LOG$来输出LOG。

由于活跃性，新产生的节点i必须按时间max（$t_{\text{弱}}+ T_{\text{引导}}$，$T_{\text{暖身}}+ T_{\text{确认}}$）使得$tx \in LOG$。 由于真实和模拟执行是相同分布的，因此$tx^* \in LOG$也是。 由于真实和模拟执行是相同的分布式，$tx^*$在LOG中tx之前的概率至少为1/2 - 在这种情况下一致性（对于真正的执行）不能满足。

9.2 对于响应式协议，1/3腐败严重

我们现在表明，在无权限模型中，即使协议知道网络延迟的先验上限Δ，也不存在响应协议，该协议在哈希功率方面可以容忍1/3或更多的损坏。 由于我们的混合共识协议容忍1/3 - E由于没有一个响应式协议可以容忍超过1/3的腐败，所以它（几乎）很严密。

我们的下界是对Som-polinsky证明的相关下界的直接修改 [2]，他们表明，在部分同步环境中，如果协议未知网络的延迟上限，那么没有安全的无许可协商一致协议可以容忍超过1/3的损坏。我们的下界证明（也是Sompolinsky's）在精神上也接近于Dwork，Lynch和Stockmeyer所显示的部分同步下界 [25] - 但是，他们的约束需要根据工作证明来适应无许可的设置。 特别是，Dwork等人的下界构造了一个3个节点的显式攻击，其中一个节点由对手控制，作为具有不同输入的两个单独节点，并与两个诚实节点交互以拆分它们的视图。 在工作证明设置中，难点在于对手不能同时模拟两个节点，因为这样做必须解决两次工作证明。 但是，我们使用类似于Sompolinsky的技巧 [2]，敌人仍然扮演两个角色，但随着时间的推移将工作证明分开 - 而受害者诚实的节点无法区分对手是迟到开始解决难题还是仅仅是网络延迟很大。

定理14（响应协议不能容忍1/3损坏）。 没有安全的无权限的协商一致也可以容忍1/3或更多的腐败。

证明。 假设存在协议Π，该协议Π抵御1/3的损坏并且是响应的，即其活性参数$T_{\text{确认}}= T_{\text{确认}}$（λ，n，α，δ）是网络实际延迟δ，但不是先验已知的上限延迟Δ。 这意味着在一些$T_{\text{暖身}}= poly$（λ，n，α，δ，Δ）时间之后，输入到诚实节点的事务将包含在$T_{\text{确认}}$时间内的任何诚实节点的输出LOG中，即使当1 / 3个节点崩溃。

我们现在描述一个明确的攻击，当α= 1/3时可以打破一致性。 假设有3个节点A，P_0和P_1，其中A由对手A控制，并且P_0和P_1是诚实的。 令Δ：= $2T_{\text{确认}}$（λ，n，α，$T_{\text{确认}}$（λ，n，α，1））是以λ为界的多项式。 该

对手A首先表现诚实，并在$T_{暖身}$（λ，n，α，1，Δ）时间过去之前立即传送所有消息。

此时，攻击者A开始在P_0和P_1之间延迟最大量Δ的消息，但是在P_0和A之间立即传送消息。此时，环境Z输入事务$tx \leftarrow^{\$} \{0,1\}^{\lambda}$到$P_0$并且不同交易$tx' \leftarrow^{\$} \{0,1\}^{\lambda}$到$P_1$。 此外，受损节点A停止向P发送消息；但是，它会记住并存储每条消息
从P_1收到。 与P_0交互时，腐败节点遵循诚实协议。 由于该协议即使在1/3节点发生崩溃时也是响应的，除了可忽略的概率以外，P_0的输出LOG将在某固定多项式时间$T_{确认}$（λ，n，α，1）<Δ中包含tx。

令$t^* = T_{暖身}$（λ，n，α，1，Δ）+ $T_{确认}$（λ，n，α，1）表示当t包含在P中的时间的上限$_0$的输出LOG。 在时间t^*，受损节点A停止发送消息给P_0，但开始与P_1进行交互，如下所示。 首先，A将其内部状态重置为它在T时刻的状态$_{暖身}$。
回想一下，A将收到的所有消息P_1排入缓冲区。 它现在会假装任何实时
$t \geq t^*$为假时间$t - T_{确认}$（λ，n，α，1），并且它在实时步骤t-T期间从缓冲器重放从P_1接收的所有消息$_{确认}$（λ，n，α，1）。 现在A遵循诚实的协议，并且对于发往P_1的每条消息，敌手立即传递消息。 请注意P_1不能
区分A是否开始求解$T_{确认}$（λ，n，α，1）时间迟，或者从A到P的网络链接$_1$是否具有$T_{确认}$（λ，n，α，1）实际延迟，但P_1到A链路传递消息
即刻。 由于协议即使在1/3节点崩溃时也是响应的，所以P_1将在t中输出LOG中包含$tx' / = tx_{确认}$（λ，n，α，$T_{确认}$，α，1））<Δ时间。 请注意，由于Δ很大，P_1还没有
听说P_0的交易tx呢。 因此，t时刻在P_1的输出日志中的概率
$T_{确认}$（λ，n，α，$T_{确认}$（λ，n，α，1））在λ中可以忽略不计。 显然，这打破了一致性。 二

备注。 我们注意到，为古典许可设置显示类似的下限并不困难。 具体而言，即使在采用PKI的情况下，在经典的许可设置中，任何响应式安全共识协议都不能容忍1/3或更多的损坏。 这样的下界将是Dwork等人与未知Δ（在许可设置中）部分同步的下界证明的直接推广。

致谢

我们感谢Kyle Croman，Ittay Eyal，Ari Juels，Antonio Marcedone，Andrew Miller，Emin Gun Sirer，Yonatan Sompolinsky，Dawn Song和Fan Zhang作为有益的讨论，尤其是Dominic Williams多次出色的讨论激励我们解决了这个问题。

这项工作部分支持NSF拨款CNS-1217821，CNS-1314857，CNS-1514261，CNS-1544613，CNS-1561209，CNS-1601879，CNS-1617676，AFOSR奖FA9550-15-1-0262，办公室
海军研究青年调查员计划奖，微软教职员奖学金，帕卡德奖学金，斯隆奖学金，谷歌教师研究奖和VMWare研究奖。 这项工作部分完成，部分作者访问Simons Institute for Computing Theory，由Simons Foundation和DIMACS / Simons通过NSF授权CNS-1523467进行密码学协作。

参考

[1] 与Kartik Nayak和Ling Ren的个人交流。

[2] 与Yonatan Sompolinsky的个人交流。

[3] Byzcoin：安全地缩放区块链。 http://hackingdistributed.com/2016/08/04/ byzcoin/.

[4] 解开比特币和拜占庭矿业的矿业激励机制。 http://bford.github.io/2016/10/25/ 矿业/.

[5] Mart'ınAbadi和JanJu¨rjens。 正式窃听及其计算解释。 在计算机软件的理论方面，第四届国际研讨会（TACS），第82-94页，2001年。

[6] Mart'ınAbadi和Phillip Rogaway。 协调密码学的两种观点（正式加密的计算正确性）。 J. Cryptology，20（3）：395,2007。

[7] 佩德罗Adao，Gergei Bana，乔纳森Herzog和安德烈Scedrov。 存在密钥周期时正式加密的可靠性。 在计算机安全--ESORICS 2005，第10届欧洲计算机安全研讨会，意大利米兰，2005年9月12 - 14日，会议录，第374-396页，2005年。

[8] 加文安德烈森。 增加最大块大小（bip 101）。 https://github.com/bitcoin/ BIPS /斑点/主/ BIP-0101.mediawiki，2015年10月检索。

[9] Adam Back，Matt Corallo，Luke Dashjr，Mark Friedenbach，Gregory Maxwell，Andrew Miller，Andrew Poelstra，Jorge Timon和Pieter Wuille。 通过挂钩侧链实现区块链创新。 https://blockstream.com/sidechains.pdf。

[10] Michael Backes，Birgit Pfitzmann和Michael Waidner。 普遍可组合的加密库。 IACR密码学ePrint档案，2003：15,2003。

[11] Iddo Bentov，Ariel Gabizon和Alex Mizrahi。 没有工作证明的加密货币。 CoRR，abs / 1406.5694,2014。

[12] Alysson Neves Bessani，Joao Sousa和EduardoAd'ılioPelinson Alchieri。 用BFT-SMART为群众提供状态机复制。 2014年6月23 - 26日在美国乔治亚州亚特兰大举行的第44届IEEE / IFIP可靠系统和网络年度国际会议，2014年DSN第355-362页。

[13] Florian Bohl和Dominique Unruh。 象征性的通用可组合性。 In Proceedings of the 2013 IEEE 26th Computer Security Foundations Symposium，CSF '13，pages 257-271,2013。

[14] Gabriel Bracha和Sam Toueg。 异步共识和广播协议。 J. ACM，32（4）：824-840，1985年10月。

[15] *Christian Cachin，Klaus Kursawe，Frank Petzold和Victor Shoup。 安全和高效的异步广播协议。 在Advances in Cryptology-CRYPTO 2001，21st Annual International Cryptology Conference，Santa Barbara，California，USA，2001年8月19 - 23日，Proceedings，第524-541页，2001中。*

[16] R.卡内蒂。 通用可组合安全性：密码协议的新范例。 在 FOCS，2001。

[17] Ran Canetti，Yevgeniy Dodis，Rafael Pass和Shabsi Walfish。 具有全局设置的通用可组合安全性。 在密码学理论，第61-85页。 施普林格，2007年。

[18] Ran Canetti和Jonathan Herzog。 通用组合符号安全分析。 J. Cryptology，24（1）：83-147，2011。

[19] Ran Canetti和Tal Rabin。 具有最佳弹性的快速异步拜占庭协议。 在第二十五届年度ACM计算理论研讨会论文集，STOC'93，第42-51页，1993年。

[20] Ran Canetti和Tal Rabin。 具有联合状态的通用组合。 在CRYPTO，2003年。

[21] 米格尔卡斯特罗和芭芭拉Liskov。 实用的拜占庭容错。 在OSDI，1999年。

[22] Kyle Croman，Christian Decker，Ittay Eyal，Adem Efe Gencer，Ari Juels，Ahmed Kosba，Andrew Miller，Prateek Saxena，Elaine Shi，Emin Gun Sirer，Dawn Song和Roger Wattenhofer。 关于扩展分散式区块链（一份立场文件）。 比特币研讨会，2016。

[23] Christian Decker，Jochen Seidel和Roger Wattenhofer。 比特币符合强一致性。 在第17届国际分布式计算和网络会议论文集中，ICDCN'16,2016。

[24] Danny Dolev和H. Raymond Strong。 经过验证的拜占庭协议算法。 *Siam Journal on Computing - SIAMCOMP，12（4）：656-666,1983。*

[25] 辛西娅Dwork，南希林奇和拉里Stockmeyer。 存在部分同步的共识。 J. ACM，1988年。

[26] Pieter Wuille Eric Lombrozo，Johnson Lau。 隔离见证（共识层）。 HTTPS：//github.com/CodeShark/bips/blob/segwit/bip-codeshark-jl2012-segwit.mediawiki.

[27] Ittay Eyal，Adem Efe Gencer，Emin Gun Sirer和Robbert van Renesse。 Bitcoin-NG：可扩展的区块链协议。 在NSDI，2016年。

[28] Ittay Eyal和Emin Gun Sirer。 多数是不够的：比特币挖掘很脆弱。 在FC中，2014年。

[29] Michael J. Fischer，Nancy A. Lynch和Michael S. Paterson。 一个错误的流程无法实现分布式共识。 J. ACM，32（2）：374-382,1985年4月。

[30] Juan Garay，Aggelos Kiayias和Nikos Leonardos。 比特币骨干协议：分析和应用。 Cryptology ePrint Archive，2014/7/2014报告。

[31] 杰夫Garzik。 块大小增加到2mb（bip 102）。 https://github.com/bitcoin/bips/blob/主/BIP-0102.mediawiki，2015年10月检索。

[32] 杰夫Garzik。 制定分散的经济政策。 http://gtf.org/garzik/bitcoin/ BIP100-blocksizechangeproposal.pdf.

[33] Omer Horvitz和Virgil D. Gligor。 弱密钥真实性和正式加密的计算完整性。 在CRYPTO，第530-547页，2003。

[34] Romain Janvier，Yassine Lakhnech和Laurent Mazar'e。 完成图片：在活跃的对手面前正式加密的可靠性。 在编程语言和系统中，第14届欧洲程序设计研讨会，ESOP 2005，作为软件理论和实践联合欧洲会议（ETAPS）的一部分，第172-185页，2005年。

[35] Jonathan Katz和Chiu-Yuen Koo。 关于拜占庭协议的预期不变圆协议。 J. Comput。 SYST。 (2) : 91-112, 2009年2月。

[36] Aggelos Kiayias和Giorgos Panagiotakos。 区块链协议中的速度安全性权衡。 密码学ePrint档案, 报告2015/1019, 2015。

[37] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser和Bryan Ford。 通过集体签名强化一致性, 增强比特币安全性和性能。 CoRR, abs / 1602.06997, 2016。

[38] 宰权。 TenderMint: 无采矿业务共识, 2014年8月。

[39] 莱斯利兰波特。 快速paxos。 分布式计算, 19 (2) : 79-103,2006。

[40] Leslie Lamport, Dahlia Malkhi和周立冬。 垂直paxos和主备份复制。 在第28届ACM分布式计算原理专题讨论会论文集中, PODC '09, 第312-313页, 2009年。

[41] Leslie Lamport, Robert Shostak和Marshall Pease。 拜占庭将军问题。 ACM Trans。 程序。 郎。 Syst。, 4 (3) : 382-401, 1982年7月。

[42] 让 - 菲利普马丁和洛伦佐阿尔维西。 快速的拜占庭共识。 IEEE Trans。 可靠的安全措施。 Comput。, 3 (3), 2006。

[43] Daniele Micciancio和Bogdan Warinschi。 Abadi-Rogaway语言加密表达式的完备性定理。 J. Comput。 Secur。, 12 (1) : 99-129, 2004年1月。

[44] Daniele Micciancio和Bogdan Warinschi。 在活跃的对手面前正式加密的可靠性。 密码学理论会议 (TCC), 2004。

[45] 安德鲁米勒, 余霞, 凯尔克罗曼, 伊莱恩施和黎明歌。 BFT协议的蜜獾。 密码学ePrint档案, 2016/2016年度报告2016。 http://eprint.iacr.org/.

[46] Satoshi Nakamoto。 比特币: 一个点对点电子现金系统。 2008年。

[47] Rafael Pass, Lior Seeman和Abhi Shelat。 分析异步网络中的区块链协议。 在Eurocrypt, 2017年。

[48] Rafael Pass和Elaine Shi。 Fruitchains: 一个公平的区块链。 手稿, 2016。

[49] 瑞恩·菲佛。 decrits共识算法: 分散协议, 没有工作证明。 CoRR, abs / 1411.1101,2014。

[50] QuantumMechanic。 证明权益而不是工作证明。 bitcointalk.org, 2011年7月。

[51] 大卫施瓦茨, 诺亚杨斯和亚瑟布里托。 波纹协议共识算法, 2014年9月。

[52] Isaac C. Sheff, Robbert van Renesse和Andrew C. Myers。 分布式协议和异构信任: 技术报告。 CoRR, abs / 1412.3136,2014。

[53] *Marko Vukolic。 追求可扩展的区块链结构: 工作证明与BFT复制。 在网络安全中的开放性问题--FIFI WG 11.4国际研讨会, iNetSec 2015, 瑞士苏黎世, 2015年10月29日, 订正论文选集, 2015年第112-125页。*

[54] Pieter Wuille。 技术增长后的区块大小\ (bip 103) 。 https://github.com/ 比特币/

BIPS /斑点/主/ BIP-0103.mediawiki.

附录

A 背景许可BFT

我们简要描述一个使用PBFT的许可BFT协议的实例 [21] 举个例子。 粗略地说，PBFT [21] 是拜占庭式状态机复制的部分同步协议。

下面我们非正式地描述当n = 3f + 1时的情况。 为更一般的情况n> 3f + 1修改协议并不难。在我们的描述中，我们假设交易是以称为批次的单位提出的。

正常情况下的操作。 我们首先描述PBFT协议的正常情况下的操作，其中所有消息都由发送者签名。

1. 当前视图的领导者向所有节点提出一个元组（“建议”，v，f，批次），其中v表示视图编号，f表示序列号。
2. 当一个诚实的节点听到（“建议”，v，f，批），如果它没有发送一个准备消息给（v，f），它多播（“准备”，v，f，批）。
3. 当一个诚实的节点从同一（v，f，批）元组中的2f + 1个不同节点收集（“准备”，v，f，批）时，它会多播（“commit”，v，f，batch）。 此外，诚实节点现在考虑准备（v，f，批）：= 1。
4. 当一个诚实的节点首先从同一个（v，f，批次）元组的2f + 1个不同节点收集（“commit”，v，f，batch） 或者当它首次从同一个（v，f，批次）元组的f + 1个不同节点收集（“提交”，v，f，批）时：节点认为lcommitted（v，f，batch）：= 1， （“提交”，v，f，批次）。 这里提交的是“本地承诺”的简称。

正常情况下的协议满足以下重要属性：

- *协议。* 如果两个诚实节点分别认为准备（v，f，批）：= 1并且准备（v，f，批次^t）：= 1，则batch = batch^t。
- *在一个诚实的领导下生活。* 如果领导者是诚实的，并且自从最新视图开始以来没有诚实节点超时，那么由诚实节点提交的任何批处理将在本地提交由0（1）个原子时间步骤中的所有诚实节点。
- *充分的准备证明。* 如果至少有一个诚实节点认为lcommitted（v，f，batch）：= 1，那么至少f + 1个诚实节点认为准备好（v，f，batch）：= 1。

  如果一个诚实的节点认为准备（v，f，批）：= 1，那么它可以产生2f + 1个带符号的准备消息，从而产生这种信念。 我们将这些2f + 1准备消息的集合称为准备证明。

请注意，协议属性的直接影响是，如果两个诚实的节点分别相信lcommitted（v，f，batch）：= 1且lcommitted（v，f，batch^t）：= 1，则batch = 批次^t。 但是，正常情况下的操作并不能保证，在潜在腐败的领导下，如果一个诚实的节点认为lcommitted（v，f，batch）：= 1，其他诚实的节点就必然会想到lcommitted（v，f，batch）：= 1.这因此激发了视图更改协议。

查看更改。 只有正常情况下的协议并不能保证领导者腐败时的生气。 为了保证活跃，即使领导者已经损坏，超时时也会调用视图更改协议。 为了获得O（nδ）最坏情况下的响应时间，我们可以对PBFT的原始指数退避策略进行一个小的修改：相反，超时可以使n个视图变化加倍。 在部分同步模型中，当超时退回到Θ（δ）并且领导者诚实时，随之而来的是活力。

粗略地说，如果一个诚实的节点听到f + 1对新视图v^t的有效视图改变请求，它将通过为视图v^t多播一个视图改变消息本身来回应视图改变请求。

当新视图的领导收集2f + 1有效视图更改请求时，2f + 1的集合有效查看变更请求一起形成新视图消息。 领导者然后向所有节点提出新视图消息。 当一个诚实节点接收到新视图消息时，对于新视图消息中包含的每个（v，f，批次）准备好的准备证明，节点就好像它刚收到一个（“建议”，v，f，batch）消息，因此为该元组多播了一个准备消息，并继续与正常情况下的操作一样。

由于正常情况操作的“充分准备”属性，以下属性成立：如果一个诚实节点认为lcommitted（v，f，batch）= 1，那么至少有一个有效的准备证明将会包含在任何有效的新视图消息中。 这确保了如果至少一个诚实的节点相信lcommitted（v，f，batch）= 1，则元组（v，f，批）保证转到新视图，因此，如果其他诚实节点在新视图中本地提交（v，f，批次^t），则认为batch = batch^t。

最后，只要新领导人诚实，并且没有诚实的节点在新的领域中超时观点，那么随着新视角的活跃。

我们将读者转介给PBFT论文 [21] 有关视图更改协议以及检查点优化的详细说明。 在PBFT论文中形式化证明并不困难 [21] 并将它们扩展到一个密码良好的框架。 此外，要证明PBFT协议实现了我们在部分定义的强大安全概念并不难 4.2.