TAS 链白皮书

[Version 0.1]

6 月 29 日, 2018

TAS 链团队

申明

本文仅是 TAS 设计探讨性的白皮书，描述了 TAS 平台的现有技术以及未来开发目标，用于技术社区讨论商议。截至目前，TAS 平台仍在开发中，其设计理念、共识机制、加密算法、开发代码和其他技术细节均可能会不断更新和更改。本文内容可以认为是截止书写时刻的 TAS 平台有关的最新信息，但它未必就是完整，完备的。可以预见，随着 TAS 平台内测，公测，以及更多的 DAPP 上链，项目组会根据需求情况不断调整和更新现有技术方案。

TAS 平台将会创造一个开放，合作，创新的技术社区生态，我们欢迎各类经济学家、技术专家、法律专家、社群运营者以及全球开发者共同参与 TAS 的技术讨论，共同参与和推动区块链公链技术的落地以及进步。

为什么要做 TAS

2006 年，我们考虑把数字版权和 P2P 网络通过代币激励有效结合起来（受理专利号 CN1889119A，“基于点对点文件交换系统的数字产品销售与分享方法”[14]），版权方可以便捷地把数字产品提交到 P2P 网络，并按 0.1 模式激励矿工在线。即用户下载付出 1 虾币，矿工得 0.1 虾币，版权方得 0.9 虾币。基于这个理念，我们创办了虾米音乐（www.xiami.com），中国 TOP3 的音乐分享平台，高峰时日活跃用户（DAU）700 万，基于用户在线行为的个性化推荐长期雄踞中国第一。

2008 年，中本聪划时代的提出了“一种点对点的电子现金系统”[1]，并在白皮书公布后 3 个月公开了源代码，这份代码运行至今，无论是 BTC 本身的价值还是 BTC 的基础技术框架-区块链，对人类社会发展的影响都是前所未有的。2013 年，Vitalik Buterin 创办的以太坊[2]把可编程性引入到区块链，让越来越多的人参与到区块链大生态建设中，并激励着更多的人往更广阔的区块链未知空间进行探索。

区块链发展到今天，有足够的时间让我们从 P2P 网络、去中心化、代币经济、生态建设等各个维度做深入的思考并作出尝试。相比缩短了时间距离和空间距离的互联网，以解决异构网络信任问题为己任的区块链技术有着更为广阔的应用场景。无论是金融、数字版权、电商、互金、中介，采用区块链技术都能通过减少中间环节大幅节省成本和提高公信力。如果说 1 千亿美金的比特币是数字货币的基石，目前 5 千亿美金的数字货币总市值代表了社会对区块链技术发展的期许，则商用 DAPP 一旦爆发，有理由相信这是一个超过 5 万亿美金的超级市场。

构建大规模商用 DAPP 需要在去中心化和高安全的基础上，同时也是低能耗和高性能的共识机制，以及更重要的，低成本高安全且方便使用的智能合约系统。

观察目前在运行中和仍处在研发阶段的公链，我们发现基于 POW 的比特币和以太坊很好的解决了去中心化和安全性问题，但是由单纯 POW 带来的高能耗[6]和低性能几乎代表了硬币的另一面。我们也观察了一些基于拜占庭容错框架的公链，通过深入的分析，我们认为拜占庭容错更多是解决分布式系统的一致性问题，而对区块链要解决的去中心化场景下的信任问题，并不是最合理的解法。基于拜占庭容错的公链，一般只能有 50 个节点可以参与铸块（实际可能更低），不然节点间的协同通讯复杂度会呈指数级上升而导致无法在较短的时间窗口内做出最优解。基于 DPOS 的公链有类似的问题，无论是抵抗物理 DDOS 还是议会制的等级方式，都和比特币的基本理念相去甚远。

回溯到比特币的源头，我们认为一条合理的公链是以如下假设为基石的：

1. 任何节点可以便捷的参与到网络中和离开；
2. 任何节点都有铸块的权力；
3. 对于一个较大数量的群体来说，在良好的激励体系下，好人总是多于坏人；
4. 坏人作恶的成本应大于他的收益。

另外，如果一条公链的最终目的是构建大规模的商业化应用，则他同时也应该是低能耗和高性能的。从这些角度看，我们发现目前国内外的公链都难以达到要求。基于此，我们设计了 TAS chain。

什么是 TAS

TAS 是去中心化、高安全、低能耗，可编程的区块链公链。比特币的 POW

机制已被时间和价值证明其数学上的不可破解性，在保留 POW 的同时，我们引入 POS 里最具技术领先性的 VRF 来解决去中心化状态下的低性能和高能耗问题，设计了混合 POW 和 POS 的 SPOW 共识机制。对 DSP（去中心化、安全、性能）不可能三角模型，我们认为去中心化大于安全，而安全又大于性能，即只有在保证了高优先级特征的前提下，去提升低优先级能力才有意义，而不是放弃或削弱高优先级的特征去单纯提升低优先级能力。举个例子，如果削弱去中心化和安全去提升性能，本质上仍是在解决几个云计算中心之间的一致性问题。而区块链技术最激动人心的划时代突破，是解决异构节点之间的信任问题而非任何其他。相比人类社会里黄金的挖矿，构建于一般等价物之上的商业活动是更精细化和复杂的模型。同样的道理，TAS 的白皮书不仅仅是数学证明和工程技术架构，而是对孵化商用 DAPP 落地的一整套解决方案。我们试图构建一个人人可参与的去中心化和高安全的网络，基于这个网络可以用很低的成本开发可持续发展的商业应用。同时通过一系列数学、密码学和工程技术的引入，让这个网络在去中心化和高安全的同时，也是低能耗和高性能的。

SPOW 的另一个好处，是更符合真实的异构网络拓扑。跟人类社会有穷人和富人一样，我们认为最好的平等并不是均贫富，而是设定一套穷人富人都可以参与的游戏规则，并根据投入的不同得到不同的收益。回顾传统的 POW，也许中本聪最初的设想里并没有矿机和矿池，而是人人可参与，可随着比特币价格的日益上涨，POW 的挖矿越来越成为普通节点难以参与的游戏。POS 存在同样的问题，大部分的 POS 直接抛弃了算力，只强调在线率，极端状态下手机和高性能 PC 在 POS 世界里是一样的地位。抛弃算力的同时必然引入无利害关系，所以大部分 POS 系统都需要较高的代币质押，有些甚至是以百分比的方式，极大的制

约了人人可参与的区块链核心思想。SPOW 承认存在不同的节点，并把它们分成重节点和轻节点，在铸块过程中两者协同工作，相互促进和监督，并分享收益的不同部分。通过组合 POS 和 POW，SPOW 极大的降低了代币的质押部分，并且设计了一条由设备指纹代替代币质押（设备即押金）的可持续发展路线，进一步降低代币的质押。综合来看，SPOW 结合了 POS 和 POW 两者的优点，并通过一系列互补技术的引入大幅提高了两者的短板。同时，为了保证持久的公平性，我们不排除未来会在更新版的白皮书中引入 DAG、账本数据随机读取等技术来 anti ASIC。详见“共识机制”部分。

SPOW 共识机制兼顾广度和深度，一个有海量轻节点参与的广度网络是好人多于坏人的前提。传统的区块链节点更强调军备竞赛而不是善用现有或闲置设备，TAS 在 P2P 网络上施行了一些新的技术，在提供 NAT 穿透的同时，相比开源标准的 RFC3489 提升了 50%以上的穿透率，为 SPOW 共识机制做了网络层的物理保证。详见“P2P 网络”部分。

SPOW 共识机制目标把 TPS 提升到 2000，但是应对超大规模 DAPP 仍有不足，TAS 借鉴谷歌 MapReduce 和阿里 ODPS 思想，设计了分片并行交易计算框架。考虑到带宽、IO、信任握手和交易数据合并复杂度等瓶颈，上百万甚至可无限扩展的 TPS 更像是一个概念而不是一个能落地实施的指标。TAS 计划通过分片并行交易计算框架把 TPS 提升到 2 万，我们认为这个数量级已可支撑目前人类社会 99%的商业化应用。详见“并行计算框架”部分。

如果没有 DAPP 的落地，则所有公链吹嘘的技术都只是空中楼阁。观察了目前众多公链系统的智能合约，结合我们之前在移动互联网时代 APP 的一些成功经验，我们在如何便捷的构建易用的 DAPP 方面做出了一些革新和尝试。详见

“智能合约”部分。

我们希望汇聚核心开发团队和社区、生态的力量，把 TAS 公链基础技术扎实完善，在数字版权、众筹、游戏、中介等商用领域合作和孵化 DAPP 落地，以技术驱动区块链生态的发展。综上所述，有理由相信 TAS 会是新一代技术型公链的引领者。

TAS 关键词

POS+POW，VRF，设备指纹，手机挖矿，分片并行交易

TAS 架构

图 1. TAS 架构概览图

- 网络层

  采用无连接的 RUDP 代替高通讯成本的 TCP。

  基于 linux 内核分析的 NAT 穿透技术。

  针对组协作铸块设计的双层重叠 KAD 网络。

- 核心层

  密码学基础部分。

  双线性对椭圆曲线实现。

  动态沙米尔秘密共享实现。

  VRF 随机数生成。

  开放架构的 TVM。

  组内 POW 并行铸块机制。

  组内沙米尔共享验块机制。

  时间窗混合 POW 铸块难度动态调整函数。

  设备指纹技术。

  分片交易并行计算框架。

- 功能层

  软分叉处理。

  智能合约可复用基础类库。

  动态组成员扩容。

  SPOW 共识机制。

  铸块激励机制。

  设备混合代币的质押奖惩体系。

  动态分片计算。

- 应用层

  安卓手机挖矿支持。

  自组织治理。

真随机数 API。

条件锁仓式众筹。

跨合约交互支持，合约软升级支持。

组创建、组创始化、组铸块和组销毁的生命周期管理。

两层高 TPS（共识层+并行计算层）支持。

核心技术

SPOW 共识算法

POW 共识算法是知名公链项目比特币和以太坊所采用的共识机制。比特币九年的运行用事实证明了 POW 算法的高安全性。随着矿机，矿池的出现，人们发现 POW 算法对这些算力强大的矿工而言是潜在的中心化，这也是以太坊在系统设计之初就考虑 anti ASIC 方案的原因。另外能耗问题也是 POW 算法存在的另一个缺陷，大量的无效算力浪费了大量的社会资源。为了弥补上述的这些缺陷，我们设计了 SPOW(Slotted Proof of Work)共识系统。它是 POS+POW 混合型共识机制，在结合矿工健康指数的基础上通过随机采样将矿工分组，组协作 VRF 算法产生的真随机数来确定当前 slot 的出块组和验证组。出块组在 slot 时间内以 POW 算法竞争出块，交给验证组做验证，达成最终的出块共识。SPOW 算法在保证高度去中心化，高安全的前提下，给出了尽可能提升系统性能的设计方案。

系统设定

系统参数	说明
n	组大小

t	恢复门限值，通常$t \leq n$。
p	大素数
$GF(p)$	基于p的有限素域
slotSec	固定的槽内铸块时间（秒）
epoSlots	纪元（epoch）包含的槽（slot）个数
gMax	矿工最大可以同时加入组的个数
ValidPeriod	组存续周期 （单位：epoch）
d	网络遍历时间（单位：slot）
CheckInterval	检查建组的周期（单位：slot），epoSlots 的因子。

系统通过初始配置来完成上述系统参数的设置。

对于系统的正常运行，需要对全体矿工有一定的假设要求：

1. 系统中肯定有诚实矿工和恶意矿工
2. 诚实矿工比恶意矿工多
3. 99.99%的矿工是趋利的

模型说明

节点。我们把用户设备上运行的一个客户端进程称为节点。根据设备的算力情况，用户可以将节点的属性设置为轻节点和重节点两种：

- 重节点： 计算型 PC，矿机等
- 轻节点： 普通 PC，手机，机顶盒，移动设备，嵌入式设备等

矿工。普通用户通过注册，可以加入参与分布式记账。设 M 为所有矿工的集合，

把它们分别添加标记1，2，$\cdots \in |\mathrm{M}|$。根据职能分类为：

- 出块矿工： 算力竞争，负责出块。
- 验证矿工： 以组协作方式工作，负责区块有效性验证，达成出块共识。

组。在任何给定时间，一些或所有$\mathrm{i} \in \mathrm{M}$被排列成一个或多个子集$\mathrm{G}_1, \mathrm{G}_2, \cdots \in M$，称为组。我们假设所有组具有相同的大小。它是被一个称为组大小的系统参数n所设定。同样根据矿工种类的不同，有*出块组*，*验证组*。

矿工健康指数。这个指数是多因素的函数，目前我们暂时考虑以下几个因素（权重从高到低排列）：

- 参与铸块（出块或验证）个数，
- 参与率(实际参与次数/理论参与次数)，
- 设备健康度（通过设备指纹获得），
- 曾加入过的组个数（在线时长），
- 权益证明。

未来可能会考虑更多的因素进行扩展。

新矿工在健康指数很低时，建议使用权益证明提高自己的健康指数。随着对系统的贡献增加，健康指数高到一定程度可以不用做权益抵押。因为其他因素权重占比更高，到后期有没有权益证明对 SPOW 算法中的随机选取几乎没有影响。

开放的参与机制

SPOW 共识算法采用开放型的参与机制。在这机制里，普通用户可以通过申请加入系统成为新矿工；老矿工也可以选择通过申请注销自己的矿工身份，退回普通用户。

*矿工的注册。*普通用户可以提交矿工申请合约（系统的特殊智能合约）来申请成为矿工，申请时用户需指明矿工的类型（Type：出块矿工或验证矿工）。每个矿工都有自己唯一的身份号

$$\mathrm{ID} = \mathrm{trans\_hash}(\mathrm{pk})$$

其中pk是该矿工作为普通用户的公钥，trans_hash 现在采用输出 256bit 的哈希函数。普通用户的公钥是与它的矿工 ID 一一对应的，而且在整个系统内也是唯一的。矿工申请合约将（pk，Type）写入区块链中。该合约针对矿工类型有不同的处理：

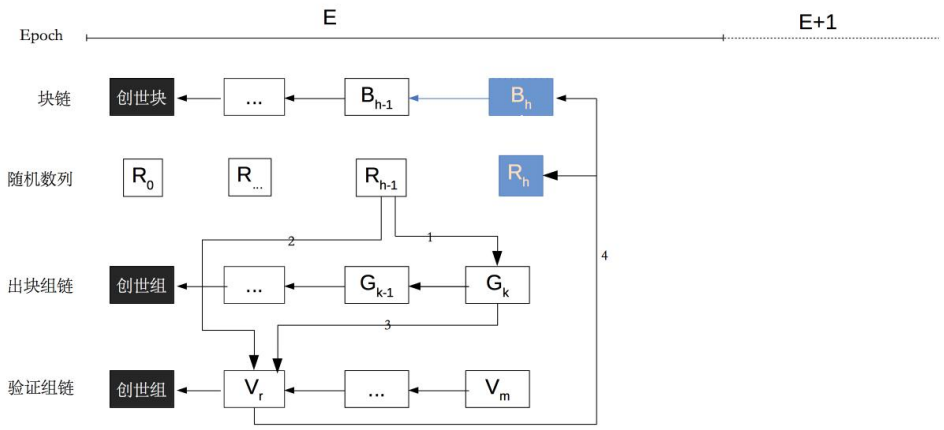
出块矿工：不需要权益抵押

验证矿工：健康指数较高，可以免保证金证明，否则，账户中若干权益抵押作为保证金证明。

原则上，我们希望重节点上的矿工担任出块矿工，轻节点上的矿工担任验证矿工。

*矿工的注销。*矿工可以提交矿工注销合约（系统的特殊智能合约）来申请退出 SPOW 系统。同样矿工注销合约会写入区块链中，若该矿工有保证金证明，判断该矿工成组状态，若未成组，则直接释放保证金给用户；否则等该矿工所在组存续周期过期时，释放保证金给用户。

铸块流程图



图示 在一个组周期中，每一轮都重复着以下操作：
1. 由上一轮随机数选出本轮的出块组
2. 由上一轮随机数选出本轮的验证组；
3. 出块组进行铸块并发给验证组验证；
4. 验证组验证签名完成后广播上链，并产生下一轮随机数；
5. 进入下一轮，重复 1-4 步直至组周期结束

SPOW 共识算法对重节点矿工，轻节点矿工随机采样分组，形成出块组链和验证组链，组协作 VRF 算法产生的真随机数来确定当前 slot 的出块组和验证组，出块组以 POW 算法竞争出候选区块，验证组以组协作方式来完成候选区块验证、并达成共识出块上链。

出于数据的不可篡改性，以及可追溯性，SPOW 系统采用双链模型，区块链和组链（出块组链和验证组链）来记录整个数据链产生过程。组链为辅，区块链的每个区块能清晰可知该区块由哪个出块组出块，哪个验证组验证，哪些矿工参与铸块。同时考虑组内矿工可能会形成组内携手作恶，所以组有存续周期，到期后会解散重新进入随机采样分组流程。

组链模型

组块的数据结构包含，但不限于以下信息：

- Type： 矿工组类型（出块组或者验证组）
- Hash： 组块哈希值
- GIS： parent 组（定义见建组检查章节）指定的信息
  - Activation： 组生效的纪元
  - Deactivation： 组失效的纪元
  - MinerArray[]： 候选矿工数组
    - pubk ： 矿工对应的用户公钥
    - ID： 矿工的身份号
- Signature： parent 组对 GIS 的签名。
- Prehash： parent 组块哈希值
- gpk： 组公钥

建组检查

系统以 CheckInterval（系统参数）为周期来做建组检查。假设当前第r轮 slot，若$r \% CheckInterval == 0$，则在铸块完成后，当前验证组查询区块链数据，构建矿工候选申请人序列（考虑链同步的问题，认为前d块之前的区块已经一致）

1. 将区块链上块高小于等于$r - d$的所有注册申请按申请顺序导出申请人序列
2. 将区块链上块高小于等于$r - d$的所有注销申请对应的矿工，从上述序列剔除
3. 对上述序列中的每个候选人，从组链里检索统计它当前已经所在的组的个数。如果统计个数大于等于 gMax，则将该候选人剔除。

上述三步操作后，得到当前的候选人序列。再将上述序列中的候选人按类型划分，得到当前的出块矿工候选人序列和验证矿工候选人序列。只要有其中一个序列满足：申请人数大于等于 2n。则由该验证组作为 parent 组，发起新组创建的提案。

新组提案

不论是出块矿工候选组提案，还是验证矿工候选组提案，其组成员选择过程一致。由于区块链上块高小于等于$\mathrm{r}-\mathrm{d}$的块已经同步，所以当前验证组内所有诚实矿工根据上述三步得到的（出块/验证）矿工候选人序列也应该一致。

以当前第 r 块中的随机数R_r作为随机种子，用 PRG 生成伪随机数列，对（出块/验证）矿工候选申请人序列进行挑选，写入 GIS.MinerArray 数组。

指定新组生效纪元。比如为当前纪元之后的第 2 个纪元

GIS.Activation = e+2

指定新组的失效纪元。由系统预先设定的参数 ValidPeriod 确定。

GIS.Deactivation = GIS.Activation + ValidPeriod

当前验证组内的诚实矿工，计算出来的组块内容应该都是一样的。所以可以由组内每个矿工分别各自出新组块，对自己的新组块签名，然后发给组内其他成员。组员在收到大于等于门限t个的相同 hash 的新块，可以用它们的组员签名恢复出组私钥对该组块进行签名。根据当前提案组的类型，会有不同操作：

- 出块矿工组：由于出块矿工主要负责工作量证明计算候选出块，所以它无需组内协作，则无需计算组公钥，直接将该组块信息广播给全网节点，通知组

链上链。

- 验证矿工组：由于验证矿工组主要负责候选块的验证，达成组内共识。作为新组的 parent 组，需要调用创世块中的分红智能合约模版，为新组创建分红合约，并公布该合约写入区块中。

验证组创建

由于组成员是去中心化网络上的对等节点，现实中某些时刻，节点因种种原因不在线不可避免，比如网络信号不好，恶意节点故意不作为等等。所以我们设计验证矿工组需要有(t,n)门限签名的能力，即有含门限t个以上的组员对消息认可签名，就能代表全组n个成员对消息的认可，并能恢复出全组对此消息的认可签名。这里我们采用了去中心化的 Shamir 秘密共享算法，来产生各个组员的组内签名私钥S_i，组内签名公钥mpk_i，以及代表组共识的组私钥对应的组公钥gpk，得到上述密钥，并对组公钥gpk达成共识，才算完成组创建。

传统的 Shamir 密钥分享技术，可以看过一个需要“中心”的过程。即它首先要有个“分发者”，“分发者”决定采用哪个秘密多项式，然后将秘密分解，发放给其他人。而区块链网络是去中心化的网络，每个节点都是平等，对等的节点，不应产生中心化的 “分发者”。另外，分发者是先于其他节点知道秘密的，如果分发者在过程中做恶，是很难预防的，因此我们采用了去中心化的密钥分享算法。其核心思想就是：让每个组员都成为“中心”（即秘密分发者），那结果就是没有物理上的“中心”。但会在组的逻辑层面上形成一个初始秘密SK（该秘密每个成员均不知晓）。每位组员通过下述步骤的前三步会形成一个组逻辑层面的分享秘密S_i。类似传统 Shamir 密钥分享算法，分享秘密集合$\{S_i\}$具有门限恢复初始秘密SK的能力，即$\{S_i\}$中任意不少于t个分享秘密均能恢复组初始秘密SK，任意少于t个分享秘密均无法得到组初始秘密SK的任何信息。

具体执行步骤如下：

1. 每个组员自行选取各自的秘密多项式$f_i(x) \in GF(p)[x]$,

$$f_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1}$$

其中多项式系数$a_{i,0}$，$a_{i,1}$，$a_{i,2}$，$\cdots, a_{i,t-1} \in GF(p)$, 均为每个组员自取的随机数。则每人的初始秘密$sk_i = f_i(0) = a_{i,0}$，以$sk_i$作为私钥，计算对应的公钥$pk_i$。

2. 每个组员计算给其他组员的分享秘密并将该分享秘密发给对应的组员。即：第 i 个组员，计算$S_{i,j} = f_i(ID_j)$，发给第 j 个组员。其中$i = 0,1,2\cdots,n;\ j = 0,1,2\cdots,n$;同时把自己的公钥$pk_i$也发给其他组员。
3. 当组员收集齐其他组员发给自己的分享秘密后, 计算所有收到的分享秘密之和$S_i = \sum_{j=1}^{n} S_{i,j} = \sum_{j=1}^{n} f_j(ID_i)$，计算$gpk = \sum_{i=1}^{n} pk_i$。

   各自将自己计算所得的组公钥gpk对全网广播。
4. 每个组员计算组内签名私钥S_i所对应的公钥mpk_i，并将组内签名公钥mpk_i告知给组内其他组员。

注意，第 2 步的组间成员间通讯，需要考虑加密通讯，防止被监听。在新组块信息 MinerArray 里，记录着组内所有组员的普通用户公钥 pubk，我们以此作为 ECDH 密钥交换做加密通讯。

经过上述步骤，每个组员获得了组内签名私钥S_i，组内签名公钥mpk_i，以及组私钥SK对应的组公钥gpk。而组层面还隐含着获得的组私钥$SK = \sum_{i=1}^{n} sk_i$，因为每个组员只知道自己的初始秘密sk_i，故没法知道SK的值。

下面对组员的组内签名私钥$\{S_i\}$具有门限恢复组私钥SK进行证明：

正确性证明：

对任意$k \geq t$,由于组员顺序不影响结果，不妨假设前k个组员。

令$F(x) = \sum_{i=1}^{n} f_i(x)$,通过$k$个组员组分享秘密的 Lagrange 插值多项式$G(x) = \sum_{i=1}^{k} S_i \prod_{j=1, j\neq i}^{k} \frac{x - ID_j}{ID_i - ID_j}$，容易知：

对$1 \leq i \leq k$,均成立

$F(ID_i) = \sum_{j=1}^{n} f_j(ID_i) = \sum_{j=1}^{n} S_{j,i} = S_i$,

$G(ID_i) = S_i$

令$H(x) = F(x) - G(x)$,容易知道$H(x)$是最高次数不超过$k-1$次的多项式，而$H(ID_i) = 0$对$1 \leq i \leq k$均成立。所以$H(x) \equiv 0$。即$F(x) = G(x)$。所以通过k个组

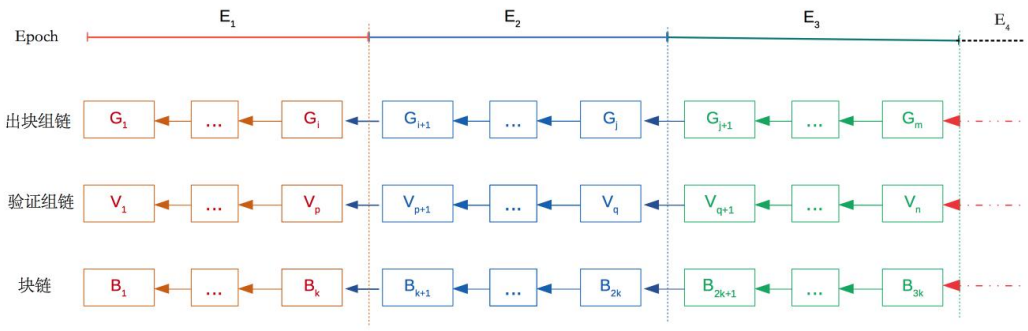
员组分享秘密的 Lagrange 插值多项式即是组的秘密多项式。计算$G(0) = F(0) = \sum_{i=1}^{n} f_i(0) = SK$

安全性证明：

由于$F(x) = \sum^{n}{}_{=1} f_i(x)$是最高次数为$t-1$次的多项式，多项式系数$t$个，而当$k < t$时，只能构建$k$个等式，由线性代数知识可知，方程个数小于未知数个数时，无法求解出未知数（解不唯一）。即无法确定多项式系数，从而无法确定$F(x)$。所以等$k < t$时，无法得到组初始秘密SK的任何信息。

验证组上组链

全网节点（包括新组成员节点）在收到相同的新组组公钥，达到门限t个以上时，才认为新组创建成功，组公钥为真。将新组块写到本地的组链上。



图示 每个 Epoch 定义为 k 块，每个 Epoch 由相应的出块组和验证组共同完成该周期的区块记账，该结束后，该周期内所有组自动解散，并生成 E+2 组周期的工作组．如上图中 E_1 结束后会生成 E_3 的所有工作组，以此类推

验证组异步创建处理

每次创建新验证组，均由 parent 组协商决定将哪些候选矿工加入新组，

并由他们自行达成组内密钥创建。而申请加入组和真正执行组创建这两种行为是异步的，由上述知道新验证组创建时，要求所有组员都分发 share piece，但在实际情况中，这个条件变得很苛刻，因为只要有一个成员不在线，就无法完成建组工作， 因而会大大降低建组成功的概率。为了解决这个问题， SPOW 共识支持新组创建的异步模式。即组创建时，允许有成员不在线，但只要在合理的时间内上线，并发现自己被指定加入新组，则自行发起上述建组步骤的第 1，2 步，将自己的 share piece 分发给其他组员，其他组员收到其 share piece 后，反馈自己的 share piece 给该组员。这样能异步收齐所有组员分发 share piece 即可新组创建成功。

验证组动态增加组员

由于验证组是轻节点设备，如普通 PC（笔记本），手机，移动设备等，流动性会比矿机，计算型 PC 大，可以预见，当验证组建组成功后的存续周期（ValidPeriod）内，节点不在线的情况会比较多。当不在线节点过多时，会直接造成(t,n)门限签名无法完成。另一方面，每个纪元（epoch）通过验证组的分红合约的记录，可以得知组内那些长期不在线的节点。所以我们考虑验证组动态增加组员算法。

算法考虑的问题是：在初始组员n人，由于活跃度的原因，组成员变为只有$r(r \geq t)$个活跃用户时，如何为组成员添加新用户。

由于用户顺序对结果不影响，不妨假设该r个活跃用户为$(ID_1, ID_2, \cdots, ID_r)$。

令$F(x) = \sum_{i=1}^{n} f_i(x)$，在验证组创建中，证明了$F(x) = \sum_{i=1}^{r} S_i \prod_{j=1, j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j}$。对于新加组员$ID_{new}$，容易知：

$S_{new} = \quad (ID_{new}) = \sum_{i=1}^{r} S_i \prod_{j=1, j\neq i}^{r} \frac{ID_{new}-ID_j}{ID_i-ID_j}$是$t-1$次多项式$F(x)$上的一点，满足当组员人数超过门限值$t$时，可以通过各自的组分享秘密，恢复出组初始秘密$SK$。由上式可知，新组员的组分享秘密由当前组活跃的$r$个组员的组分享秘密线性构成。如何不泄露自己分享秘密S_i的前提下，让新组员ID_{new}构建出它自己

的组分享秘密S_{ne} ？

具体解决流程如下：

1. 当前活跃的r个组员分别自行选取各自的秘密多项式$g_i(x) \in GF(p)[x]$，

$$g_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1},\ i = 0,1,2\cdots,r$$

其中多项式系数$a_{i,1}$，$a_{i,2}$，$\cdots, a_{i,t-1} \in GF(p)$，均为每个组员自取的随机数。且满足$a_{i,0} = g_i(0) = S_i \prod_{j=1,j\neq i}^{r} \frac{ID_{new}-ID_j}{ID_i-ID_j}$，$i = 0,1,2\cdots,r$。

2. 当前活跃的r个成员各自分别计算给其他组员的分享秘密并将该分享秘密发给对应的组员。即：第 i 个组员，计算$D_{i,j} = g_i(ID_j)$，发给第 j 个组员。其中$i = 0,1,2\cdots,r;\ j = 0,1,2\cdots,r;$
3. 当组员收集齐其他组员发给自己的数值后，计算所有收到的数值之和$D_i = \sum_{j=1}^{r} D_{i,j} = \sum_{j=1}^{r} g_j(ID_i)$，并将该数值发送给新组员。
4. 新组员计算$A(x) = \sum_{i=1}^{r} D_i \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j}$，由以前证明知$A(x) = \sum_{i=1}^{r} g_i(x)$。则$A(0) = \sum_{i=1}^{r} g_i(0) = \sum_{i=1}^{r}(D_i \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j})$,新组员的组分享秘密$S_{new} = A(0)$。

此时，活跃的r个组员和新组员构成一个新的组，但老组员各自的组分享秘密没变，组的初始秘密SK也没变，当新组人数超过门限t 时，可以通过各自的分享秘密，恢复出组初始秘密SK。

正确性证明：

因为

$$\sum_{i=1}^{r} D_i \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j} = \sum_{i=1}^{r} \sum_{h=1}^{r} D_{h,i} \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j} =$$
$$\sum_{h=1}^{r} \sum_{i=1}^{r} D_{h,i} \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j} = \sum_{h=1}^{r} g_h(x),$$

所以

$$\sum_{i=1}^{r} D_i \prod_{j=1,j\neq i}^{r} \frac{x-ID_j}{ID_i-ID_j} = \sum_{h=1}^{r} g_h(0) = S_{new} = F(ID_{new})$$

安全性证明：

由于$F(x) = \sum_{i=1}^{n} f_i(x)$是最高次数为$k-1$次的多项式，多项式系数$k$个，而当$t < k$时，只能构建$t$个等式，无法求解出多项式系数（解不唯一），从而无法确定$F(x)$。所以等$t < k$时，无法得到组初始秘密SK的任何信息。

块链模型

数据区块的数据结构包含，但不限于以下信息：

- BlockHeader： block 块头信息
  - Hash： 当前块 hash
  - Height： 当前块高
  - CurTime: 当前块铸块时间
  - PreHash： 上一块 hash
  - PreTime： 上一块铸块时间
  - Castor: 出块人 ID
  - CGroupId： 出块组 ID
  - VGroupId： 验证组 ID
  - Signature： 出块组组签名
  - Rand： 随机数
  - Difficult： POW 难度
  - Transactions[]: 交易集 hash 列表
  - Nonce

真随机数生成

结合 ECDLP 的 Shamir 秘密共享方案

我们采用的 Barreto-Naehrig 椭圆曲线$E: y^2 = x^3 + b,\ b \in GF(p)$

其中，有限素域$GF(p)$：

$$p = 36x^4 + 36x^3 + 24x^2 + 6x + 1$$
$$r = 36x^4 + 36x^3 + 18x^2 + 6x + 1$$

这里x是 63 bit 的数，p，r均为 bit 长度在 256 左右的素数。

两个有限群$G_1 = E\big(GF(p)\big)[r]$，$G_2 = E[r] \cap Ker(\pi_p - [p])$，

我们采用近年来论文中 bn 椭圆曲线上最优线性对算子[4] $e: G_1 \times G_2 \to GF(p^{12})$ 定义为：

$$e(Q,P) = (f_{6x+2,Q}(P) \cdot H)^{(p^{12}-1)/r}$$

这里$H = l_{Q_3,-Q_2}(P) \cdot l_{-Q_2+Q_3,Q_1}(P) \cdot l_{Q_1-Q_2+Q_3,[6x+2]Q}(P)$

$f_{6x+2,Q}(P)$是可以通过 Miller 算法计算的。

由双线性算子的特性：

对任意$P_1, P_2 \in G_1, Q \in G_2$，成立$e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$

对任意$P \in G_1, Q_1, Q_2 \in G_2$，成立$e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$

对任意$p \in G_1, Q \in G_2$，$a, b \in \mathbb{Z}$，成立$e([a]P, [b]Q) = e(P \cdot Q)^{ab}$

这里，记$[\cdot]$为椭圆曲线上的倍乘。如$[a]P$是椭圆曲线上点P的a倍乘积。

基于椭圆曲线的签名算法：

- m ∈ {0,1}*: 需要签名的消息（二进制表示）
- 计算$R = H(m) \in G_1$
- 计算$\sigma = [x]R$，其中x是用户签名私钥，σ即为所得签名

那么结合上述的组创建（去中心化 Shamir 秘密共享），对组员签名私钥$\{S_i\}$与组私钥SK，存在如下关系：

$$SK = \sum_{i=1}^{k} S_i \prod_{j=1, j\neq i}^{k} \frac{-ID_j}{ID_i - ID_j}, \quad k \geq t$$

则对消息$\mathrm{m} \in \{0,1\}^*$，每位组员的签名为$V_i = [S_i]R$，由双线性椭圆曲线的上述特性，可得：

$$[SK]R = \left[\sum_{i=1}^{k} S_i \prod_{j=1,j\neq i}^{k} \frac{-ID_j}{ID_i - ID_j}\right] R = \sum_{i=1}^{k} \prod_{j=1,j\neq i}^{k} \frac{-ID_j}{ID_i - ID_j} V_i$$

简而言之就是，在双线性椭圆曲线上，上述的构组方式所得组员签名私钥对消息签名后，当得到$k \geq t$条组内成员的消息签名，可以用 Lagrange 插值多项式得到组私钥SK对该消息的签名。

在 Shamir 秘密共享算法中，恢复组私钥SK，是要泄露$\{S_i\}$的。利用双线性映射 e 的性质，在不泄露$\{S_i\}$的情况下来完成组私钥SK的签名，保证了组员签名私钥可以不断复用。通过使用该技术，可以通过门限签名来实现组内共识，比拜占庭算法（BFT）效率更高。

VRF 随机数生成方法

由于组私钥SK无人知晓，所以该签名$[SK]R$具有不可选择，不可预测，不可改变，却可以通过组公钥gpk来验证签名$[SK]R$是否由该组签出的。它是一种广义的 VRF 随机数生成方法。

SPOW 系统采用的随机数生成方法，就是采用上述组协作的 VRF 方法。

记$B^i.Rand$是第i轮 slot 出的区块的 Rand 值。对于第r轮 slot，使用的随机数

$$\mathrm{R}_r = hash(B^{r-1}.Rand)$$

按选组策略可以确定第r轮 slot 的验证组，由当前验证组对$(r|R_r)$做签名，收集组内门限t个以上的签名后，恢复的组私钥签名作为当前 slot 生成随机数，写入第r轮 slot 所出的区块的$B^r.Rand$。

$$B^r.Rand = recover(sig_1(r|R_r), sig_2(r|R_r), \cdots, sig_t(r|R_r))$$

前一块的 Rand 确定当前块的工作组，当前工作组以上述公式产生当前块的 Rand，确定下一块的工作组。其中R_0是创世块中的 Rand，由系统初始设定。

若第r轮 slot 的工作组没能完成出块，则第$r+1$轮使用的随机数

$$\mathrm{R}_{r+1} = hash(hash(B^{r-1}.Rand))$$

按选组策略可以确定第$r+1$轮 slot 的工作组，对$(r+1|\mathrm{R}_{r+1})$做群组多重签名，产生当前块的 Rand。若第$r+1$轮仍未出块，当前 slot 使用的随机数，以及当前块随机数生成准则，以此类推。

选组策略

假设当前进入第r轮 slot，计算：

当前纪元$e = r/epoSlots$

随机数$\mathrm{R}_r = hash(B^{r-1}.Rand)$，

从组链中获取当前存续的出块组列表$\{\mathrm{g}CB\}$

$$\{\mathrm{g}CB^{\mathrm{i}}|\mathrm{g}CB^{\mathrm{i}}.\mathrm{GIS.Activation} \leq \mathrm{e} < \mathrm{g}CB^{\mathrm{i}}.\mathrm{GIS.Deactivation}\}$$

和验证组列表$\{\mathrm{g}VB\}$

$$\{\mathrm{g}VB^{\mathrm{j}}|\mathrm{g}VB^{\mathrm{j}}.\mathrm{GIS.Activation} \leq \mathrm{e} < \mathrm{g}VB^{\mathrm{j}}.\mathrm{GIS.Deactivation}\}$$

以该随机数R_r作为随机种子，用伪随机数生成函数 PRG 可以随机在上述存续出块组列表$\{\mathrm{g}CB\}$里确定当前出块组，验证组列表$\{\mathrm{g}VB\}$里确定当前验证组。这些选择都是其他节点可审计验证。

后期我们可能会考虑拿组的健康指数(组员健康指数和)，利用追随中本聪算法（FTS）来作为选组的依据。

出块组出块

通过计算上述的真随机数，每个节点矿工均可以计算得知当前 slot 的出块组和验证组。对某个重节点矿工，当他计算得知自己处于当前 slot 的出块组时，则启动 POW 算法，构建候选区块B，使得：

$$f(d_r) \geq SHA256\big(SHA256(B)\big)$$

其中$d_r \in [0,1]$是第r轮的难度系数，$f(d)$是难度函数。

常规的，定义候选块B的$SHA256\big(SHA256(B)\big)$的前置 0 的个数为区块的难度。

在 slot 时间段内，如果求解出满足该难度的候选区块B，则通知给当前 slot 的验证组成员，让他们做区块验证，并组内达成共识。如果 slot 时间段，没有找到满足难度条件的候选区块B，则该 slot 完成不了出块。

当前块的出块难度d_r，初始难度由创世块给出，以后是会根据前若干块的难度动态调整，保证 slot 时间的出块率。

验证组验证

验证组成员，对收到的候选区块B，进行以下操作：

- ✧ 验证区块B的出块合法性：B.CGroupId 是否是当前 slot 的合法出块组，B.Castor 是否是 B.CGroupId 的成员。
- ✧ 验证区块B是否满足难度 :计算$SHA256\big(SHA256(B)\big)$, 验证其满足难度。若该 slot 收到多个满足难度的区块B，验证组选择难度最大的，即 hash 最小的区块。若验证不通过，则无后续操作
- ✧ 对区块内容做验证，各种 Hash 和对应的状态 hash（比如 merkle tree root）

做验证，若验证通过，则用自己组内签名私钥对区块的哈希 B.Hash 做签名，并将签名发给组内其他成员。若验证不通过，则无后续操作。

验证组共识出块

验证组内成员接收到其他验证组组员的签名后，首先验证签名身份，确定该签名是由该组员签出无误。当收到大于等于门限数t个的组员签名后，可以用恢复函数恢复出当前工作组的组私钥对该 block hash 的签名：

$$g\_sig_r(block_r) = recovery(\mathrm{sig}_r{}^1(block_r), \mathrm{sig}_r{}^2(block_r), \cdots, \mathrm{sig}_r{}^t(block_r))$$

此时组内达成共识，该块可以向组外广播。

最后，以自己对该块的签名作为凭证，调用建组时上链的分红智能合约。

组外验证

组外节点收到新块的消息，首先判断该块的验证组合法性，然后从块内容中获得验证组的组 ID，然后通过组链上该组的公钥，以此来验证块上的验证组签名是否正确。若签名验证正确，本地上链成功，则将该块继续对外广播，否则停止对外广播。

通讯优化

参考比特币的 P2P 网络的传播性能：1KB 消息，在 1 秒钟内完成全网 95%的传播，而 1MB 消息需要 1.5 分钟完成全网 95%的传播。我们考虑到组成员散布在

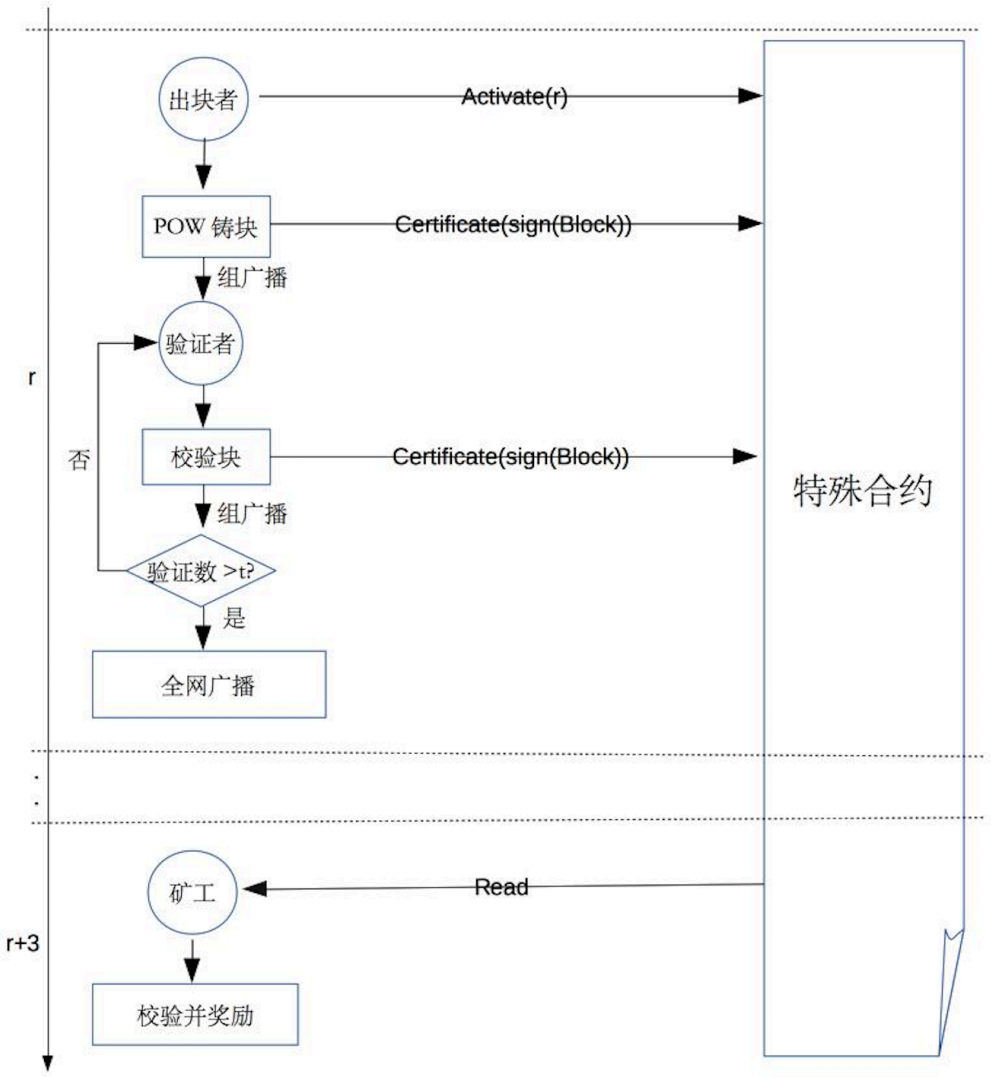
世界各地，显然出块组以 Block header 传给验证组更优。

TAS 采用验证组内 header 的 Hash 达成强一致，然后交易同步到本地后，上链时的账户状态验证来保证块的有效性。这样可以达到更高的共识效率。如果出块人是诚实矿工，这样的流程效率更高。如果出块人是恶意矿工，这区块在上链时会验证失败，由上述的组外验证保证不会被传播，这样也保证了安全。另外，如果出块人时恶意矿工，该块的验证组验证后会发起该块的分红合约，但上链验证失败使该区块上链失败。块链中分红合约与块链的不一致，可以作为作恶凭证，对作恶的出块人给出相应的惩罚。比如降低出块矿工的健康指数，保证金扣除等。

经济奖励模型

TAS 的设计必须实现对矿工的经济激励，让他们的付出有所收益，以此系统才会良性发展。 经济激励模型必须做到让所有矿工可根据自身付出的劳动获取相应的报酬，可预期，可审计，可追溯，公平公正却又不可篡改。

上述的阐述更像是一个商业的劳动报酬合约的内容，智能合约正是为此而生。自然而然的，智能合约成为实现经济激励模型的最好方法。



TAS 系统在每一轮铸块（区块/组块）开始启动一个特殊的智能合约，在本来参与铸块的每位矿工，包括算力竞赛矿工和验证矿工，在完成自己对本轮的计算工作后，携带相应工作证明的凭证 call 该合约，合约为其记录工作证明，并计算出奖励份额。在一定时间窗口后，系统自动触发该合约的奖励逻辑，将奖励打入矿工账户。

采用智能合约实现经济激励的设计，不仅可以实现安全公正的经济激励，同时为评估每个矿工节点的健康度活跃度等参数提供了重要的数据基础，以此出

发,我们可以轻易的识别出系统的积极矿工和僵尸矿工，而矿工的健康指数可以直接对矿工后续继续参与 SPOW 计算有着正向反馈的功效。

分叉处理

分叉选择：

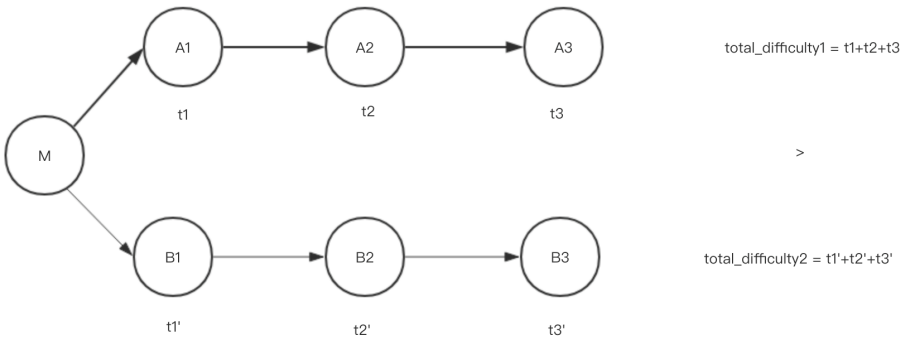
由于共识机制的特性和 P2P 网络状况的不确定性，软分叉在区块链中无法避免，在出现软分叉之后，TAS 链的各节点以总难度最高的分叉

$$totalD = \sum D_i$$

作为规范链，分叉节点会通过分叉调整简单快速地将本地链调整至规范链，从而保证链的一致性。

分叉调整：

节点遇到分叉之后，首先找到分叉点，然后比较分叉点到当前高度的总难度值做出选择。在寻找分叉点的过程中，采取局部比较的方式，每次请求一定步长的目标链片段进行二分比较，然后根据比较结果调整步长，从而快速定位到分叉点进行分叉调整。



共识分析

去中心化分析

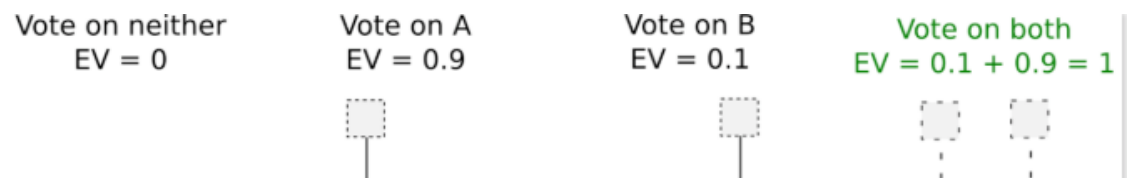
POW 算法是高度去中心化的算法，但是随着矿机，矿池出现，它们成为 POW 算法的隐性中心。算力小的节点获得铸块的概率很小。SPOW 算法，对算力设备随机分组，具体轮次由真随机数确定出块工作组。简单来说，POW 算法全网算力最优者胜出，SPOW 算法由随机指定的出块组中的算力最优者胜出。前者全局最优，后者组内最优。SPOW 算法采用分组方式，不会让全网最优者一直大概率胜出，从而让更多的参与铸块者能获得铸块收益。另外，SPOW 算法的出块组还会定期重建，随机的分组策略，多样的分组结果，让更多的参与者能获得收益，解决了算力隐性中心的问题。同时因为分组算力竞争，也大大降低了全网的能耗浪费。

SPOW 算法还对参与验证的验证组成员进行奖励。每轮验证组的选择也是由真随机数确定的，所以每个验证组都是有相同概率获得验证机会。

安全分析--攻击与防御

1. 无利害攻击(nothing at stake)

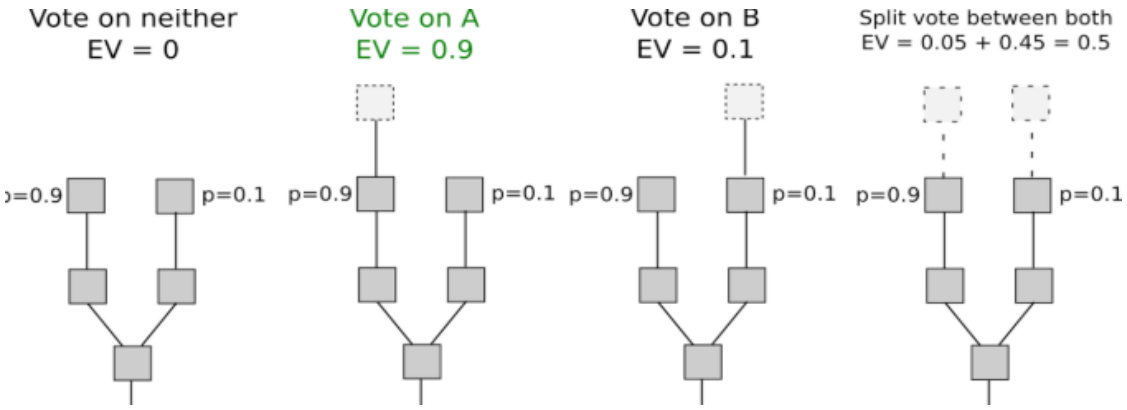
在 POS 算法中，只为创造区块提供奖励，恶意出块或者基于错误的分叉出块都没有惩罚措施，这就出现在多链竞争条件下，理智的矿工的最佳策略是在每条链上进行”挖矿”。由于他们没有花费物质上的算力，只用权益进行投票。这意味着在该机制下，不管哪条链胜出，矿工都会得到奖励，如下图:



图中, p 为该分支胜出的概率, EV 为某矿工节点的收益期望

可见，在分支情况下，矿工选择在每条链上出块的期望收益会最大化。在极端情况下，全网矿工都唯利是图，即使没有攻击者，也有可能达不成共识。

在 SPOW 算法中，选定出块组后，组内采用 POW 的方式出块。在该方式下，当出现分叉时，若同时在所有分叉中出块，需要将算力均分多分，大大降低出块



的效率，这直接影响矿工收益。如下图

本来，矿工只需要简单的计算就能知道应该在哪条链上出块，从而得到 0.9 的收益，由于算力均分，导致最后的收益只有 0.5，理智的矿工不会这么做。

下面对两个分叉的情况做个简单的数学证明:

假设：当前两个分叉A和B，各自胜出的概率为P_a,P_b 矿工对不同的分叉采取不同的算力分配，对A分叉分配X_a,B分配X_b，期望收益为Z，于是有:

$$P_a + P_b = 1$$

$$X_a + X_b = 1$$

$$Z = P_a * X_a + P_b * X_b$$

要求使得 Z 的最大化的 X_a 和 X_b;上述方程可转化为

$$Z = \left(2P_a - 1\right) * X_a - P_a + 1$$

P_a为常数，显然当$P_a > 0.5$时，X_a取最大值 1 时Z能取到最大。 这个直观的解释是: 矿工应该把所有算力分配给胜出概率最大(即难度值最大)的那条链能取到最大的收益。 所以该机制会使全网达成一致，累计难度最大的链终将胜出。

2. 长程攻击(Long-range attack)

在 POS 算法中，出块的速度没有限制。 在系统初期，矿工不多，如果这些矿工联合起来，回到成千上万个块之前的状态，从那时候开始铸块，在短时间内铸出一条更长的链，这时候用户无法辨认哪条链是主链，甚至攻击者发布的链有可能干掉主链。 当前很多 POS 实现是通过限定区块能回滚的数量减轻这个问题，但仍然治标不治本。

在 TAS 系统中，攻击者要达到长程攻击的目标，其最佳策略，也就是成功率最高的策略为:

选择回退到某个只有极少量铸块组的 epoch 的起点，因为铸块组越少需要控制的成员越少，则成本越低。 这个 epoch 几乎就是创世第一个 epoch，在该 epoch 只有一个创世组铸块。

在该 epoch 起点开始基于 POW 铸块

必须在该 epoch 结束前铸出累计难度大于主链的分叉广播出去，因为一旦 epoch 结束仍未满足此条件，则会有新的组会加入铸块, VRF 的不可选择不可预测不可篡改特性使得攻击者无法逆转铸块的轮回和历史潮流，一旦轮到其他组(诚实组)铸块，攻击者的分支会被抛弃。

我们再来分析攻击者在该 epoch 完成此条件的可能性。 SPOW 算法限定每个块出块时间的上限CT_{max}和下限CT_{min}，以及难度上限D_{max}和D_{min} 则在一个 epoch 期间(时间为T_{epoch})，攻击者无论有多少算力，最多能产生的累计难度为:

$$D_{total} = T_{epoch}/CT_{min} \cdot D_{max}$$

令$D_{max} = k * D_{min}$，假设此时主链上已有H块(即回退的高度)，最差情况下，主链的累计难度为$D_{main} = H * D_{min}$

则攻击者得逞的条件为$D_{total} > D_{main}$，即:

$$\frac{T_{epoch}}{CT_{min}} * k * D_{min} > H * D_{min}$$

简化后得:

$$\frac{T_{epoch}}{CT_{min}} * k > H$$

式子左边的参数都是 TAS 系统级参数，其计算结果是一个常量。 直观的解释是攻击者只能回退到有限的甚至是很少的几个块以前实施攻击才有可能成功，而且必须保证每次都在CT_{min}时间内铸出一个D_{max}的块，这个对算力要求非常苛刻。 另外一个方面，如果在很少的几个块以前实施这种攻击,则可视为普通的软分叉现象，系统欣然接受难度更大的分叉。

3. DDOS 攻击

DDOS 攻击主要是攻击者通过组织一批计算机,对服务节点发起大量的请求，从而使得服务节点忙于处理这些请求而耗尽资源，无法继续对外提供服务的攻击方式。 TASchain 采用分组组间随机选择出块组的方式铸块，每个铸块组出块

时间有一定的时间窗口，而组内采用 POW 的方式铸块， 假设某个时间全网的区块高度为 H，攻击者若要达到攻击的目的，必须做到以下四点:

- 时刻计算得知当前时刻的铸块组
- 通过查询 TAS 的底层网络，得到组内每个成员的网络地址
- 组织足够多的 DDOS 请求，发给组内超过 51%的成员，致使他们都瘫痪
- 持续的执行上述 3 个步骤

我们对每一点进行分析攻击者的攻击成本:

第一点是很容易做到的，只要你连接了一个正常工作的矿工节点;

第二点，由于我们每个组组成员数量很大，在成百甚至上千的数量级，在铸块的时间窗口内得出半数成员的地址需要耗费不少的时间;

第三点，使得一个矿工节点瘫痪的最有效方式是将其带宽打满，因为相对于其他资源，带宽更容易到达瓶颈。 假设运行 TAS 的矿工平均带宽为 10Mbps(这个需求在现代很容易满足)，则要打满 100 个节点的带宽，攻击者的带宽至少是 10 * 100 = 1Gbps，再加上相应的机器电力成本，这个代价已经不是普通人能够承受的。

第四点，也是最难的一点，即使攻击者在高度 H 时攻击成功，当前组无法出块，当该组的时间窗口过后，下一个组自动开始高度为 H+1 的铸块，往后类推，只要攻击者在某个时刻没成功做到第三点, TAS 会继续往前推进，所以攻击者必须持续的保持很高资源消耗去执行攻击，我们相信，这个攻击的代价没有人愿意承受。

最后, TAS 的底层网络也将实现轻量的 DDOS 防御功能，会限制同一 IP 同一时刻最多请求的数量，这会进一步加剧攻击的难度。

4. 女巫攻击(sybil attack)

如果攻击者注册大量的账号，并且使用这些账号进行挖矿，将可以提高自己的收益，尤其是在 POS 系统中，账号的注册成本非常低，而且不需要很多的算力，只要缴纳押金即可参与记账，从而有钱的人就有更大的话语权，而变得更有钱。这样破坏了系统的公平性,也容易中心化。

TAS 系统主要通过几点来预防女巫攻击:

首先，攻击者的大量账号都需要分别缴纳押金后才会被选为铸块候选者，当这些候选者陆续被加入到铸块组后，这时候攻击者的账号可以有两种选择，第一种是参与 POW 铸块以获取更高收益，这需要攻击者为每个账号配备更高的算力才能在算力竞赛中胜出，然而攻击者拥有这么高的算力获取收益后，也不是他独享，这会带动其他矿工富裕，并不有失公平。第二种方式是只参与验证，不做 POW，同样也能分享出块者的收益。 然而也只是分享部分。

其次，无论攻击者有多少账号，也无法撼动 VRF 的不可选择和不可预知的特性，也就无法改变系统的流向。

最后, TAS 的设备指纹技术多个账号在同一台设备上同时工作，这就提高了攻击者的攻击成本，攻击者必须准备更多的设备来运行这些账号。这些硬件成本与挖矿的收益不成比例时，我们相信没有人会这么做。

5. 中间人攻击

不管是区块链系统还是传统的互联网系统，所有信息都必须在网络上传播，这建立在一套已经非常成熟的 TCP/IP 协议之上。 这会引入中间人攻击的风险。

设想当主机 A 和主机 B 进行通信时，攻击者可以轻易劫持通信内容，然后可以进行信息篡改或者窃取，这是一个巨大的隐患。

区块链系统运行在点对点网络之上，节点之间明文通信，网络环境错综复杂，似乎更易受到攻击。 区块链系统存储的所有信息都是公开的，攻击者没有必要通过中间人攻击只为了获取信息。 所以对攻击者而言，只有信息篡改对他们是有意义的，比如对某一笔交易的接受地址篡改为自己受益的地址。 但是比特币系统运行了将近 10 年，经受住了全世界攻击者考验，被证明是一个安全性很高的系统，这得益于它的签名验证机制。 TAS 也有自己的签名验证机制，系统采用经过优化的椭圆曲线签名算法，消息发送者用自己的私钥 SK 对消息签名，接受者收到消息后，利用发送者的公钥 PK 进行验证。 任意消息被篡改后，都无法用原始公钥验证通过，这在密码学上得到保证。但这建立在一个前提下，发送者的私钥是绝对保密的。

6. 最后操作者攻击

假设最后操作者(即最后负责广播区块的矿工)在所有共识流程结束后，通过修改某些区块或交易信息，然后在广播出去，以达到某些目的。这里要分三种情况:

第一种情况，为了让自己获利，攻击者可以修改交易，把某些交易的目的地址设置为自己的，或者将自己某些非法交易打包进块里。 在 TAS 的共识体系中，每一个块都需要组内t(门限值)个成员签名，一旦最后操作者在组员签名后修改数据再广播，将导致接受者验证失败从而丢弃该块，攻击者就无利可言。

第二种情况，攻击者企图修改挖矿收益的分配份额。TAS 的矿工挖矿收益份额在组内共识时就被固化在签名中，这个签名所有人可以验证却无法篡改，这保证了收益份额是不可更改的。

第三种情况，攻击者篡改数据，只是为了让本轮共识最后落空，损人不利己。 由于 TAS 的组内签名验证机制是只要有成员率先收到t个成员的签名即可恢复出组签名，将块广播出去。 因此此类攻击有效的前提是，本轮共识中没有任何诚实矿工能收到t个签名。 我们简单计算一下这个概率。

我们假设每个组的诚实节点数为X，每个诚实节点收不到t个签名的概率是P ,并且每个诚实节点接受签名是独立事件，则所有诚实节点都未收到t个签名的概率为:

$$P_{fail} = P^X$$

取$X = 100$，$P = 0.9$，则

$$P_{fail} = 0.9^{100} = 0.00002656$$

即每个组只有 100 个诚实节点，每个节点有 90%的概率都收不到t个签名的情况下，攻击者得逞的概率仍然非常低。

7. 双花攻击

此类攻击者在实施攻击时有两种方式:

第一种方式，在交易集中装进自己的两笔交易，将同一笔钱转给不同的账号。在 TAS 的账户体系中，每个账户发起的交易需要指定一个交易序号作为由该账户地址发起的交易次数，每发送成功一个交易计数加 1。同时也为每个账户记录其可

用余额。 因此若攻击者的两笔双花交易采用了不同的交易序号作为交易次数，则会被认为是不同的交易，在余额充足的情况下，两笔转账成功，但是消耗的是两笔钱，这没有问题。 若余额不足，则转账失败。 若攻击者采用相同的交易序号，则其中一笔交易次数校验失败导致转账失败，双花攻击失败。在这种方式下，其他矿工能很快校验出来而拒绝执行攻击者的交易。 如果攻击者试图贿赂其他矿工，则他需要贿赂全网一半以上的矿工，这个贿赂的难度和成本非常巨大。

第二种方式，在轮到攻击者所在组铸块时，攻击者提前花费了一笔交易广播出去，同时自己也开始铸块并把该交易打进交易集中，等待该笔交易确认(TAS 的确认时间很短)后，再把自己铸的块广播出来，若攻击者的区块难度大于当前主链新增的累积难度，就会被矿工作为新的主链，如此一来，先前花费的交易就像不存在一样，但是攻击者已经得到利益。 根据前面长程攻击的分析，

假设 TAS 的交易确认时间内产生的高度为H，则主链累加的最低难度为

$$D_{accu} = H * D_{min}$$

攻击者区块的难度满足

$$D_{attack} < D_{max} = k * D_{min},$$

攻击者得逞则必须满足

$$D_{attack} > D_{accu}$$

转换为

$$k * D_{min} \geq D_{attack} > H * D_{min}$$

k和H为系统级参数，只要系统设定满足，$H > k$则攻击者永远无法得逞。 TAS 的设定正是满足了此条件。

8. 自私挖矿攻击

假设有自私的矿池 S(elfish)，和剩余的其他矿池 H(onest)，

当 S 挖到区块后，如果之前和 H 在拼人品，则立即广播，否则就私藏并开始挖下一个，此时：

1 若 H 先挖到，则 S 也广播，拼人品(网络状况)看谁最终胜出

2 若 S 靠先机先挖到，则广播所挖的两个区块

由于 S 隐藏了挖出的块，让其余的网络算力浪费资源去计算，而自己在挖一下个区块的时候就取得了先机，同时他需要监视其他的矿工节点，一经发现他们广播了该块，则需要立即出块。这在比特币系统中变得可能，有矿池拥有足够的算力，但目前比特币系统未曾出现此类攻击，更高层的经济问题决定了矿工会无私的支持比特币网络。

然而在 TAS 的 SPOW 算法中，此类问题不复存在，因为矿工无法选择自己能否成为下一个铸块节点，这是由 VRF 决定的。 算力强的矿工只会通过短时间内算出更大难度的块后立刻广播，才能确保更大的收益。 这里会有一个博弈过程，矿工需要在只要算到一个满足难度值的块就广播出去和花费更多的时间算出一个难度足够大块再广播做出权衡。前者出块的速度更快，但是由于难度小，所以竞争力小，后者出块更慢,算出更大的难度，但是要承担其他快的节点已经了更高累计难度的风险。

9. 51%攻击

在 POW 的系统中，任何具有 51%算力攻击者都可以发起 51%攻击，从而控制整个网络，让系统朝着对自己有利的方向发展。

在 TAS 系统中, VRF 站在上帝的视角，公平的对待每一位矿工。这时候的 51%攻击分两种情况:

第一种，如果 51%的节点或算力均匀的分布在所有的组中，每次 VRF 选择的组都有半数的攻击者节点，所有节点都老老实实的进行 POW 算力竞赛进行挖矿， 无论最后是攻击者胜出还是诚实者胜出，所有人都能根据一定配额得到挖矿收益，你无法做到只让自己以收益而其他人无益，这对系统反而是好事。

第二种，如果 51%的节点或算力被集中大多数组中，也就是说系统同时存在攻击者组和诚实者组。 攻击者希望一直都在自己控制的组铸块。而 TAS 在选择下一个铸块组的时候，采用了基于真随机数的 VRF 算法，该随机数与区块，交易等变量无关，具有不可选择，不可预测，不可改变却可审计等特性，使得无论攻击者拥有多少算力，都无法改变铸块的历史潮流。

P2P 网络

● TP2P（TAS P2P）介绍

我为人人，人人为我，P2P 和基于 P2P 发展起来的 DHT/KAD 等技术是去中心化自治的早期雏形，也出现了不少互联网时代的成功案例。

区块链推动了 P2P 技术的进一步发展，从单纯的分布式存储，到共享计算和带宽，以及和密码学技术的结合，都对 P2P 技术提出了更高的要求。我们观察到目前 POW 系列的公链和 POS 系列的公链使用的 P2P 技术仍是较为原始的 P2P 技术，比如不支持局域网穿透，或者需要搭建静态 IP 的专用网络或者手动开启路由设备的 UPNP 功能才能加入到 P2P 网络中。静态 IP 的网络费用高昂，而运营商提供的大部分入网设备需要破解才能开启 UPNP 功能，普通节点想要加入到区块链 P2P 网络中的门槛仍然非常高。比特币和以太坊的节点网络，越来越被矿机和矿池垄断，普通用户无论从物理网络层面和逻辑共识层面都越来越难以进入铸块节点网络。TAS 革命性的提出了 SPOW 概念，让普通 PC 甚至手机（轻节点）和矿机（重节点）都能参与到铸块共识，我们对传统的 P2P 技术做了大幅的技术升级以满足 TAS 的 SPOW 共识。重点包括：1. 使用新型的 NAT 穿透技术（专利申请中）大幅提高节点在线率。2. 针对 SPOW 以组为单位进行铸块的模式，采用基于 KAD 的组播技术提高组内组间通讯效率。3. 使用 RUDP 代替 TCP，大幅提高通讯性能。

TP2P 架构

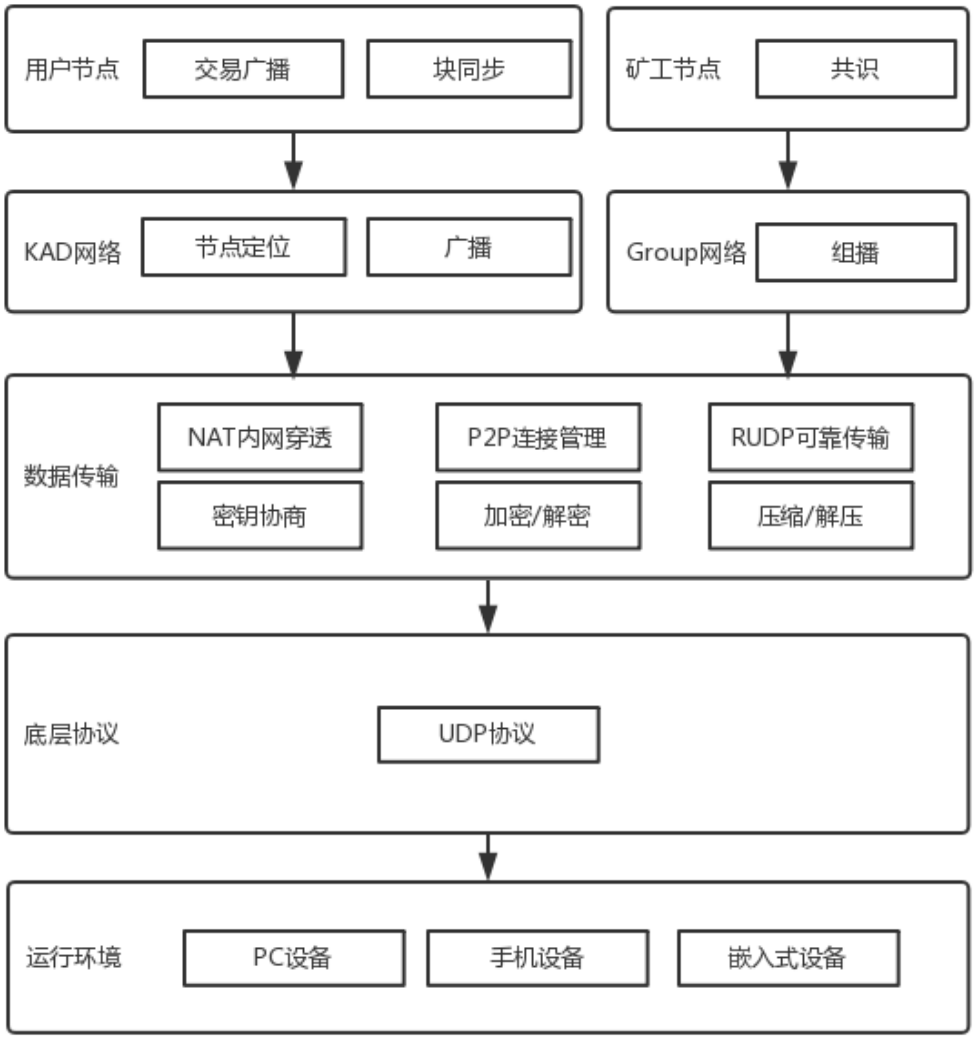


图 9. TP2P 架构图

TP2P 核心技术

- **高在线率**

P2P 技术作为区块链基石，一定程度上决定了区块链能走多远。而内网穿透又是 P2P 的关键技术之一，穿透率的高低决定了多少人可以自由的加入到节点网络参与铸块，而参与的节点越多，整个系统越安全是区块链共识的基本理念。

即我们认为在一个越大的节点网络中，越满足好人大于坏人的分布。

在内网穿透领域，主流使用标准的 STUN[10]作为解决方案，穿透率在 30%左右，效果并不理想。STUN 将 NAT 设备分为四类，全锥型、限制锥型、端口限制锥型、对称型。通过对六万个互联网应用的用户统计，全锥型占比 5%，限制锥型占比 7%，端口限制锥型占比 58%，对称型占比 30%。理论穿透率是 56%（5% x 100% + 7% x 100% + 58% x 70% + 30% x 12% = 56.20%）。同时由于 NAT 防火墙的存在，不请自来的 UDP 包到达 NAT 设备时，在连接跟踪模块产生记录，相应的一些副作用导致端口预测失败，进而导致穿透失败。STUN 并没有考虑 NAT 防火墙的影响，实际穿透率仅在 30%左右。

内网穿透的本质是预测地址转换设备的映射端口，非对称设备都可以准确预测映射端口，所以对称设备如何处理，则是决定穿透率的关键因素。

由于目前大部分路由设备都是以 Linux 内核为基础研发的路由系统，我们根据 Linux 内核协议栈的实现，从端口映射规律的角度重新定义了 NAT 设备类型，将 NAT 设备分为三类：主机端口、固定端口、对称端口。通过六万个互联网应用的用户统计，主机端口占比 75%，固定端口占比 23%，对称端口占比 2%。理论穿透率是 96%（75% x 98% + 23% x 98% + 2% x 0% = 96.04%）。

内网穿透的另一个障碍是路由器防火墙，路由器防火墙在接收到异常的数据包后会更改映射端口而导致随后的穿透失败。TAS P2P 自研了 TTL 动态调整算法，完美解决 NAT 防火墙的问题，从而使实际穿透率基本与理论穿透率相等。

观察目前已有的区块链系统，大部分不具备穿透能力，好一些的采用了主流的标准穿透技术（STUN/RFC3489），穿透率只有 30%左右，而 TAS P2P 通过多年在音乐下载、网吧多线路融合等 P2P 技术的沉淀和创新，自主研发了 95%以上高

穿透率的 NAT 技术，极大的提高了全网普通节点的在线率，为上层 SPOW 共识保证了网络物理层的高连通率。

内网穿透	STUN	TP2P
原理	RFC3489	基于 Linux 内核协议栈映射端口分配算法
穿透率	30%	95%
设备分类	全锥型/限制锥型/端口限制锥型/对称型	主机端口/固定端口/对称端口
防火墙穿透	不支持	支持

图 10. 内网穿透技术对比

- **二层组播网络**

TAS P2P 的所有节点都会加入到全局的 KAD 网络中，KAD 网络采用基于汉明距离的拓扑方法来快速定位节点，该方法的好处是在定位过程中通过对数级的收敛保证了高效性。

TAS P2P 的 KAD 网络启动后，某个节点都将和 16-32 个邻居节点建立连接，由邻居节点间的通讯完成交易同步、块同步和组同步。

同时针对 SPOW 基于组的共识，TAS P2P 建立二层子 KAD 网络定位组成员节点。每个组在完成初始化后，在组成员之间建立组播网络。每个组员启动后会和 8 个同组的组员建立连接，来保证组内成员间铸块和验证的高效通讯。相比全局的 KAD 网络，二层组播网络保证了组内消息更快速的投递，对整个网络的负载也较小。

- **采用 RUDP 代替 TCP**

在追求高在线率节点网络和大量碎片验证数据通讯的前提下，TCP 相比 RUDP 的

劣势非常明显。基于此，TAS P2P 采用开源且成熟的 RUDP 代替 TCP。

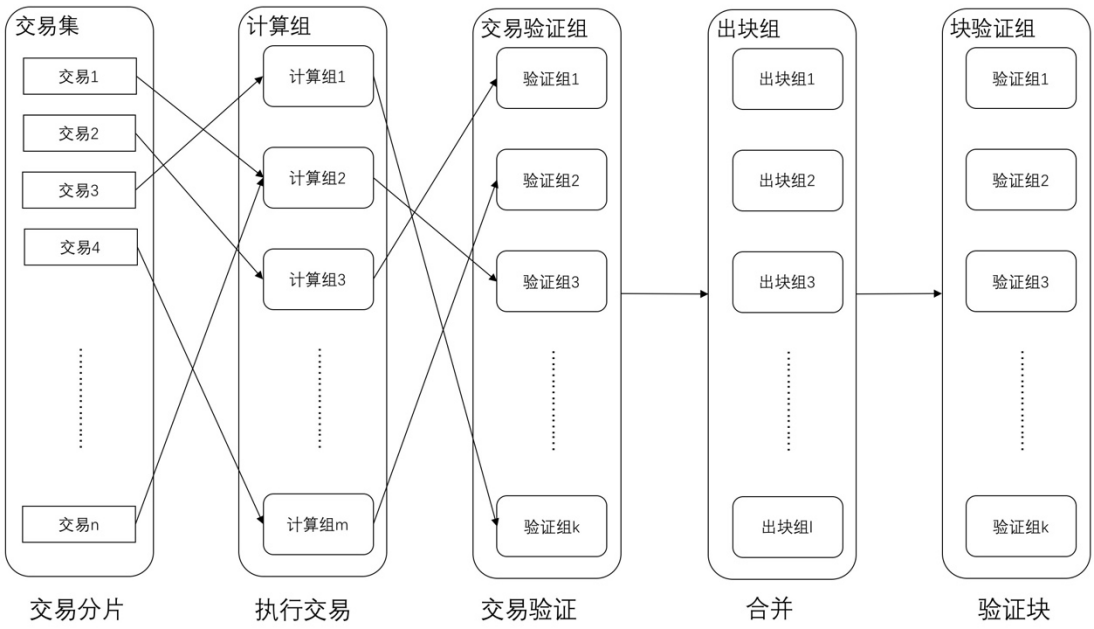
底层协议	TCP	RUDP
内网节点穿透	难	易
高质量网络传输速度	高	高
中质量网络传输速度	中	高（ARQ 快速重传）
低质量网络传输速度	低	高（FEC 冗余传输）

图 11. 通讯协议对比

分片并行执行框架

目前区块链系统在保证系统安全的前提下，性能普遍不高，所有的公链要支持大型商业级 DAPP 落地都面临着扩容的问题。扩容的核心是增加交易的吞吐量。为了更好的解决扩容问题，TAS 借鉴 Google 的 map-reduce 和阿里云批量计算的设计思想，构建出一套分片并行交易执行框架。

在我们的框架中，出块矿工由两类角色组成，分别为计算组和出块组（组概念与构建流程参见组链模型）。计算组负责执行交易，出块组负责块的最终生成。计算组与出块组都需要提交验证组，验证正确性。由于出块组需要存储全量链数据，一般由重节点组成。而计算组相对要求较低，可以由轻节点组成。系统流程如下所示



1、 交易分片

根据交易发送者的地址以及上一块的签名，将交易发送到不同的计算组。

在这里，我们并没有限制目标账户地址，比起状态通道模式具有更好的灵活性。同时相同的发送者发送的多笔交易，都将落到同一个计算组，避免了双花问题。最后在分发交易的策略中采用了 SPOW 共识的 VRF 输出随机选择计算组，通过计算组复用和不可预测性提高了系统安全性。

2、 执行交易

计算组根据当前账户状态执行交易，同时记录账户状态变化情况。计算组内采用 POW 竞争计算的方式选择哪一个节点将交易执行结果提交到交易验证组。

对于交易的执行往往需要依赖数据，也即状态，例如账户余额，合约内保存的数据等。在我们的设计中，计算组缓存了最新的状态信息，一旦接受到一个新块，就会更新本地状态。所以计算组本地即可完成交易的执行，无需跨链等复杂操作。

3、 交易验证

交易验证组通过执行交易，验证账户状态变化的准确性和一致性。当验证通过时，验证组使用门限签名方式生成组签名，提交出块组。

4、 合并

出块组合并收到的账户状态变化，生成最终的账户状态，出块，提交块验证组。

5、 块验证组验证最终块

参见 SPOW 共识机制的验证组共识出块部分。

智能合约

在 DAPP 迅速发展的过程中，正面临着和传统 App 发展时遇到的一样的问题或更多新的问题，TAS 将致力于在保证契约（智能合约）性、安全性、公平性的前提下，为 DAPP 开发者打造一个完整和高效的开发者生态，并解决 DAPP 发展过程中遇到的传统 App 开发过程中遇到的问题和面临的新问题。

- **合约日常升级**

区块链带来了智能合约（DAPP），在传统的 App 开发领域，App 升级一直是 App 开发里很重要的一环。目前 DAPP 还不能很好的支撑一个合约的升级，TAS 将建立一套合约升级方案，让合约创建者可以发布合约升级通知，合约参与者可以及时的得到通知，并自由选择是否更新为新合约。

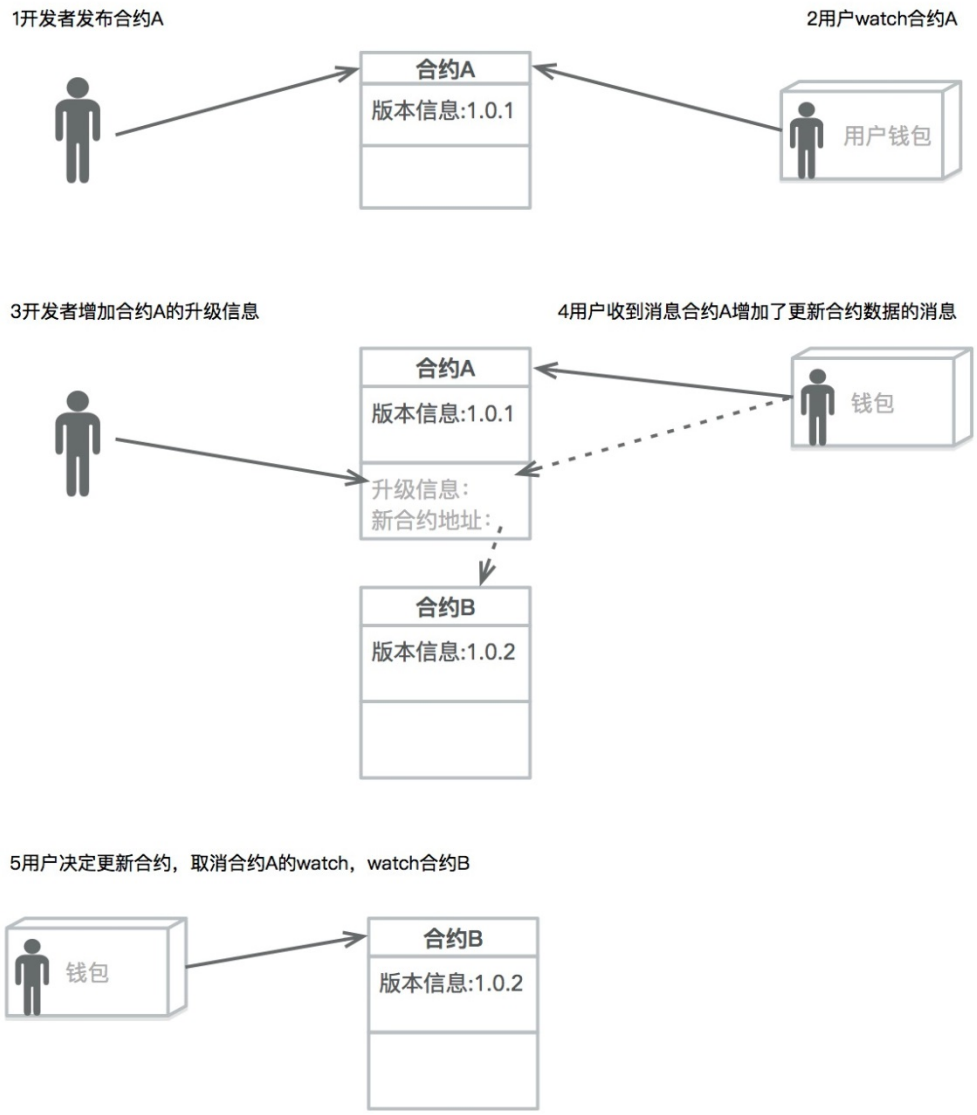


图 7. 合约日常升级

● 合约严重问题修复（硬分叉）

在传统的 App 开发过程中 ,除了 App 的正常功能性升级、小 bug 修复升级外，还有严重 bug 导致的用户强制升级、服务端数据回滚。在以合约为基础的 DAPP 维护过程中，不能单方面由开发者决定是否进行强制升级和数据回滚操作，那么解决 DAPP 开发过程中的严重问题修复是必须要面对的问题。

TAS 将提供一套完善的接口（或工具链），在 DAPP 出现严重问题时让开发者可以方便的对正在维护的 DAPP 进行硬分叉处理，包括不限于合约的已有完整

拷贝、已有数据的完整拷贝、已有数据的回滚拷贝，并和日常升级一样通知所有参与此 DAPP 的用户。

- **高效虚拟机**

目前大部分支持智能合约的区块链，都采用模拟栈式架构的虚拟机，模拟栈式架构解决了一致性的问题，但是同时也带来了运行效率低下的问题。为了提高运行效率，TAS 研发了基于 JVM 技术的 TVM，在保证一致行的基础上，显著的提高了合约的运行效率。

- **应用组件库**

在 DAPP 依赖的基础库会随着 TVM 发布，不会为开发者带来额外的上链存储负担，但是在真正的软件开发领域，还存在第三方库依赖。TAS 发布的合约将支持以类库形态发布，并容许其他智能合约进行类库级别的依赖，再次减少 DAPP 上链存储的负担。未来还会增加付费类库合约引用的功能，鼓励开发者构建和发布方便其他开发者的类库合约。

- **多合约协同**

对于大型商用 DAPP 来说，不太可能把所有代码放在一个智能合约里，而是根据不同功能、分层设计等理念把代码拆分成水平和垂直的多个子智能合约，协作完成商用逻辑。而合约本身的契约性，导致在 DAPP 维护过程中合约是不可修改。那么一个良好的 DAPP 设计都会考虑在 DAPP 实现过程中使用多合约实现，每个合约完成不同的功能，以便在需要 DAPP 升级时，可以用最小的代价升级（替换）原有的合约。

一个完整的多合约之间的通信、依赖关系和打包方案，是对开发者必不可少的支持。

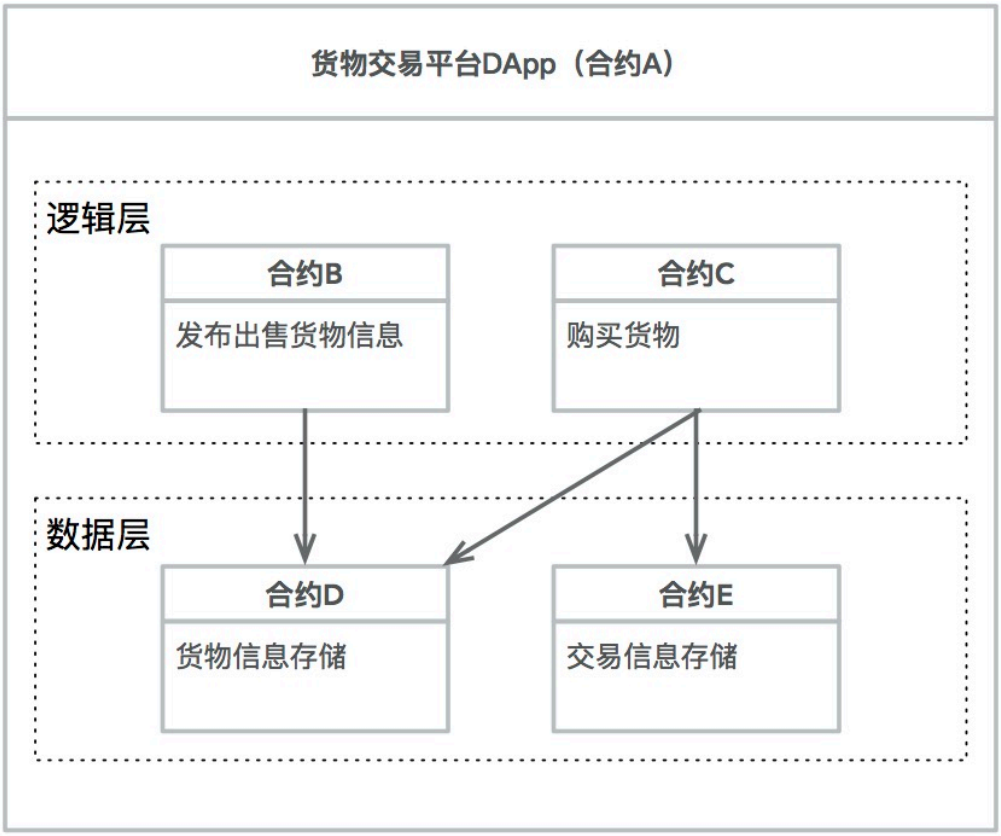


图 8. 多合约协同

社区治理

分叉问题一直是区块链发展中面临的巨大难题和挑战。 对于设计目标是支持大规模商业化应用的公链来说，如何实现自组织治理，避免硬分叉对DAPP带来的影响，是每一个区块链系统设计越来越重要考虑的问题。TAS把自组织治理纳入整体的系统设计内，研发了去中心化的智能协商治理协议SNGP(Smart Negotiation Governance Protocol)。

SNGP通过智能合约实现了一套可编程的社区治理协议。 通过SNGP不仅可以调整系统和DAPP的配置，还可以进行系统行为的升级，从而实现自组织的自我更新和迭代，避免随意的硬分叉对商用DAPP的影响。

在实现上，SNGP被设计为一种投票协议，即满足一定条件的自组织成员可以发起提议，自组织成员通过设备指纹或保证金质押进行DPOS二级选举投票，在得票率超过阈值后，整个自组织系统进行自动调节或升级。同时投票的最终结果会对用户的投票决策进行奖惩激励，以保证用户每一次会在正确的方向上进行委托或者投票。在更长远的计划里，我们会支持HTTP和RPC两种通讯方式从外部引入结果，以及公投宣判人角色，在一定程度上解决链上和链下的闭环问题。

SNGP作为一种投票协议，接受的输入是一个提议或预测，输出是该提议的投票结果或最终的自然结果。 所以SNGP的作用不仅限于自组织的治理，实际上，任意一种可以用投票处理的场景，如预测博弈、竞选博弈等等，都可以用SNGP公平公正的解决。

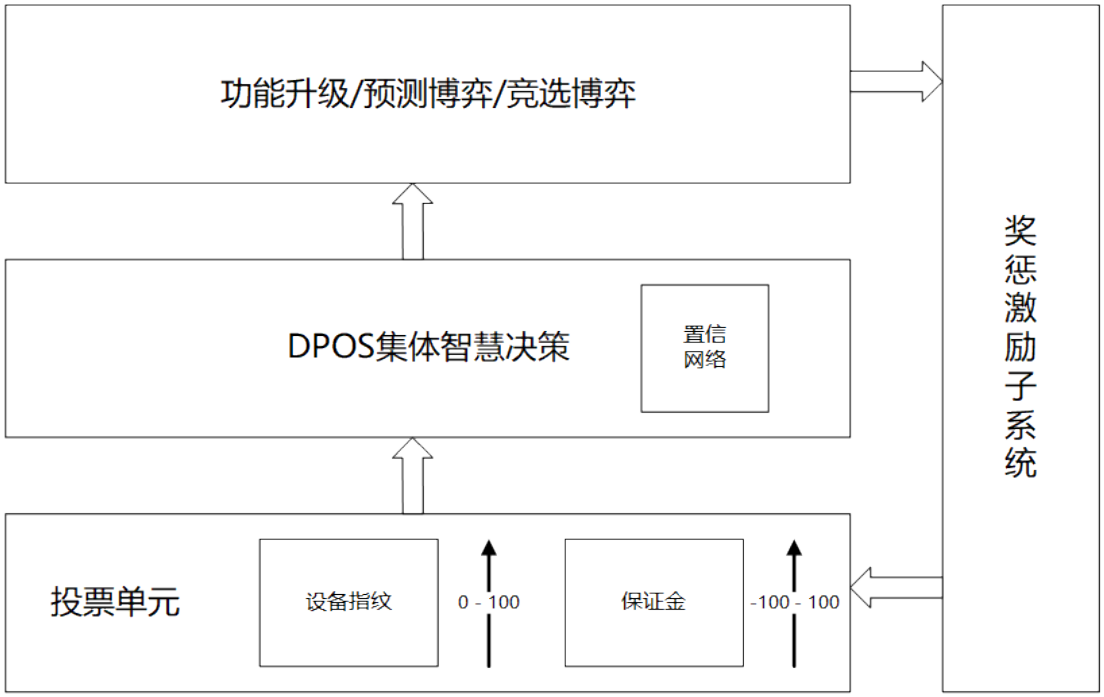

功能升级/预测博弈/竞选博弈
DPOS集体智慧决策
置信
网络
投票单元
设备指纹
0 - 100
保证金
-100 - 100
奖惩激励子系统

图 12. 自组织治理流程

代币经济模型

基于 TAS 链发行 TAS 代币，能够方便地计算 TAS 链上数字化经济活动，能够方便地参与 TAS 社区治理和生态体系建设。

健康和谐的代币经济模型有助于将开发者、投资者与社区建设成为一个可持续的平台，TAS 团队承诺将长期发展 TAS 公链技术。

TAS 团队在新加坡注册成立 TAS 基金会，该基金会是服务于 TAS 开源社区的非营利组织，TAS 基金会负责为 TAS 社区筹集运营资金，对参与并推动 TAS 开源项目的企业、组织或个人提供支持。该基金会作为治理主体，全面负责管理 TAS 技术开发和应用开发，维护 TAS 持有人权益，宣传推广 TAS 品牌。

TAS 代币发行量为 100 亿个，分配及使用情况如下：

（1）私募 40%，对象为天使投资人、机构主要为 TAS 的早期发展提供资源和技术支持（无锁占比 11.25%，18.75%锁定 6 个月，然后每月释放 3.75%；15%锁定 1 年后释放）

（2）公募 5%，用于提供 TAS 发展资金和技术支持；

（3）团队预留 18%，用于支付研发费用、工程师薪酬；（锁定 6 个月后每月释放 0.9%）

（4）社区奖励 20%，主要用于对社区中重要参与者和贡献者奖励，激励与支持社区参与者开发各类商业应用，定期举办社区开发者活动，推进 TAS 生态发展；

（5）基金会预留 17%，TAS 基金会的运营支出，包括：代码安全审计、市

场品牌推广、法律财务等合规审查、第三方审计、持有人权益维护以及初始社区治理和生态搭建等方面；

所有的锁定时间都是以 ERC20 智能合约递交开始计算。

路线图

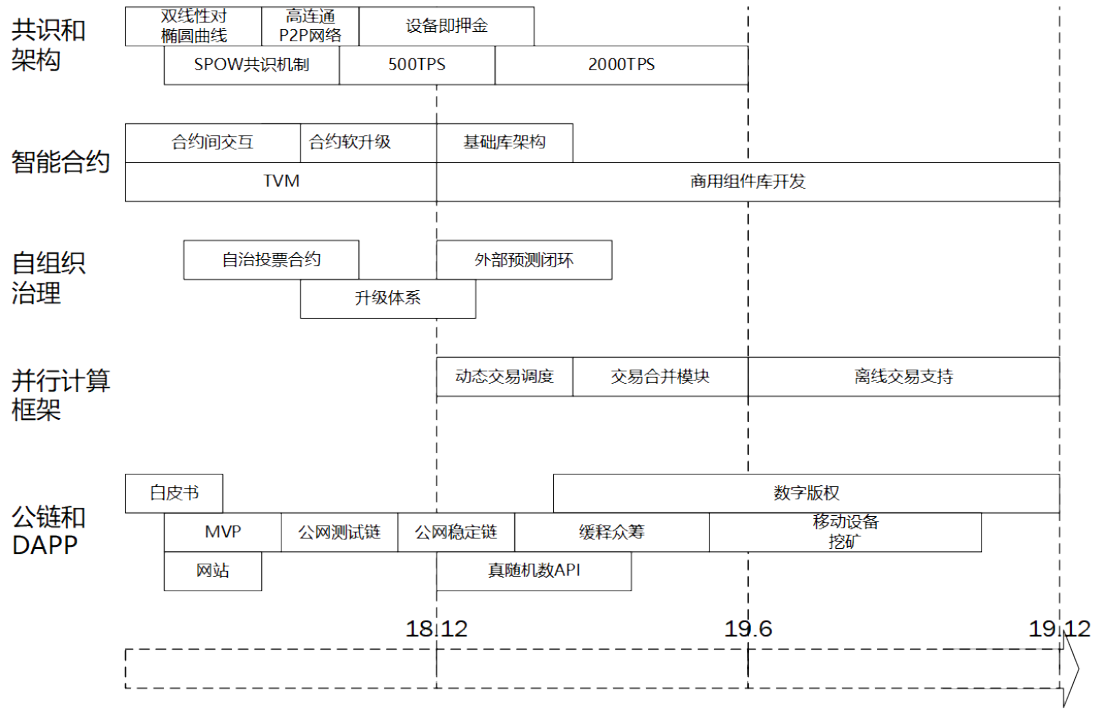


图 12. TAS 实施路线和重要时间节点

参考文献

[1]. Satoshi Nakamoto, “Bitcoin: A peer-to-peer electronic cash system, In: URL http://bitcoin.org/bitcoin.pdf,” 2008.

[2]. Vitalik Buterin. “Ethereum: A next-generation smart contract and decentralized application platform.” In: URL https://github.com/ethereum/wiki/wiki/White-Paper , 2014.

[3]. NIST, “Sha-3 standard: Permutation-based hash and extendable-output functions,” 2015.

[4]. F. Vercauteren, Optimal pairings, IEEE Transactions on Information Theory, vol. 56, no. 1, pp. 455–461, 2010.

[5]. l lvm. https://llvm.org/. Accessed: 2017-08-01.

[6]. Bitcoin Computation Waste, http://gizmodo.com/the-worlds-most-powerful-computer-network-is-being-was-50 2013.

[7]. S. Micali. Algorand: The Efficient Public Ledger. https://arxiv.org/abs/1607.01341.

[8]. S. Micali, M. Rabin and S. Vadhan. Verifiable Random Functions. 40th Foundations of Computer Science (FOCS), New York, Oct 1999.

[9]. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies.

[10]. Kademlia: A Peer-to-peer Information System Based on the XOR Metric, In URL https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[11]. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). In URL https://datatracker.ietf.org/doc/rfc3489/

[12]. Sharding FAQs https://github.com/ethereum/wiki/wiki/Sharding-FAQs

[13]. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004

[14]. wanghao, "Digital product selling and sharing method based on point-to point document exchange system", In: URL https://patents.google.com/patent/CN1889119A/en, 2006

附录