ekva项目白皮书

目录

一、概述

1.1 项目背景

根据最新的数据统计，目前随着区块链技术在全球范围内的应用和普及，最为主流的基于区 块链技术的数字货币市场已经超过 1000 亿美金，如比特币（BTC）、以太币（ETH）、莱特币（LTC）、瑞波币（Ripple）等。

区块链是一串使用密码学方法产生的相关联的数据块，这种数据块中包含了一次比特币网络交易的信息集合，而且基于链式构建的数据块结合强有力的算力保障从而达到了“一经确认， 无法篡改”的目的。区块链技术也成为了分布式数据存储、点对点传输、去中心化共识机制、 加密算法等计算机技术的新型应用模式。

区块链作为加密货币的底层技术，用于分布式地存储历史交易信息，区块链中的每个区块包 含若干交易信息，矿工一旦挖到新的区块，就将其加入区块链，并以密码学的方式保证区块 信息不可篡改和不可伪造。为了保证系统的正常运行，区块链将经济因素集成到激励层，为 矿工提供充足的动力去寻址新的区块，激励层主要包含经济激励的发行机制和分配机制。因 此，如何设计高效实用的共识机制和激励层成为区块链机制设计中最重要的两个方面。

然而，目前主流观点所认同的工作量证明机制由于其耗费的巨大电力浪费，也被越来越多的 研究者和设计者们所诟病。另外随着 ASIC 矿机及矿池的出现，比特币及相关的机密货币也呈现出中心化的倾向。长期以来，大多数人对挖矿的理解还停留在 ASIC 芯片或显卡上。直到 2014 年，一个名为 Burst 项目的上线，为大家带来了一种全新的共识机制：容量证明共识机制（POC），如果说工作量证明机制是所有矿工利用自己的算力去寻找一个随机的哈希数来打包区块，那么 POC 共识机制可以理解为，通过某种既定的算法产生数量众多的伪随机数，并将这些随机数存入硬盘，在竞争打包区块的时候，只需要通过扫描硬盘并匹配结

果来打包区块。

事实上，容量共识机制由于不依赖过大的运算量，仅与系统的存储容量相关，磁盘容量越大， 其挖矿成功的概率越高，反之其挖矿成功的概率越低，所以 POC 的运行只需要维持矿机最基本的电能供应即可，其能源消耗仅相当于相同安全级别下传统工作量证明机制所消耗能源的百分之一。

另外，由于POC 机制只需要足够的硬盘空间，这对于大多数参与者来说，门槛足够低，这也是 POC 机制更去中心化的主要原因。

然而现有 POC 机制也有一些比较明显的缺陷，比如其存在于硬盘中的数据都是一些无用的哈希值。如果我们在 POC 的机制设计中，让矿工们硬盘中存储的都是对社会有用的数据，那么矿工在挖矿硬件设备上的投资就可以被用于大范围分布式存储和归档系统，而这正是 ekva 项目的目的。

1.2 当前技术限制

自区块链技术诞生以来，区块链的性能问题和安全问题一直是倍受业界关注和讨论的核心问 题，同时也在制约着区块链行业的发展。事实上，高性能、去中心化、安全这三个要求，在 理论上是不可能同时满足的，这也是被业界称为“区块链技术的不可能三角”。

在数据存储上，由于区块链采用拥有时间戳的“区块+链”的结构，在可追溯、防篡改上具 备安全优势，也易于分布式系统的数据同步，所以它的每一个节点都下载和存储所有的数据 包，利用强冗余性获得强容错、强纠错的能力，使得网络可以民主自治，但它并不像分布式数据库那样随着节点的增加可以通过分布式存储来提高整体的存储能力，它只是简单地增加 副本，这样就带来了巨大的数据存储空间损耗。

从共识机制的角度来看，为了确保安全的前提下解决比特币所采用的工作量证明方式的低效

性，权益证明（POS）、股份授权证明（DPOS）等机制被采用，但是无论是哪种形式的改进，实际上都是对去中心化的让步，形成整体或局部的中心化。

最近的一系列基于容量共识机制的技术的引入与实用，越来越多的研究者对存储类的区块链 项目投入热情，这主要得益于存储类的共识机制低能耗的同时，具有比传统共识机制有着更 高的去中心化程度。然而现阶段大多数项目所基于的容量共识机制，其存储着大量的对社会 毫无价值的哈希值，造成一定硬盘空间的浪费。

1.3 ekva 项目的解决方案简介

ekva 项目在众多技术方案中，选择存储量证明这种去中心化程度最高的共识机制作为其基础，并改进传统的容量共识机制，将原来存储的大量无用哈希数据 文件修改为存储真正对社会有实际意义的数据文件。在不损失去中心化安全性的前提下，让 矿工在挖矿硬件设备上的投资可以被用于大范围分布式存储和归档系统。

二、交易模型

2.1 账户余额模型 vs UTXO 模型

首先，整个区块链网络的记录存储模式可以分为两大类：以以太坊为首的账户余额模型和以 比特币比特币为首的 UTXO 模型。

UTXO 模型是通过链式的方式组织所有交易的输入和输出，每一笔交易的输出最终都能追寻到一个币基交易，在比特币中也即被挖出区块的奖励交易。由于在 UTXO 中没有账户的概念，所以并行地处理交易不会出现任何问题，同时不可变的账本能够让我们在节点快速更新 时，也能分析某一时刻整个网络中数据的快照。

在 UTXO 模型中，当我们需要计算某个地址中的余额的时候，需要遍历整个网络中的全部

相关区块，同时，并行处理交易虽然可行，不过并行地创建交易却会出现很多问题，例如多 笔交易使用了同一个 UTXO，会导致双花，最终只有一笔交易会被网络接收。

相比于 UTXO 模型，账户余额模型更加容易理解，账户余额模型其实就是一个巨大的状态机，其中的状态都是由多个账户组成的，交易仅仅体现的是账户余额的变化。

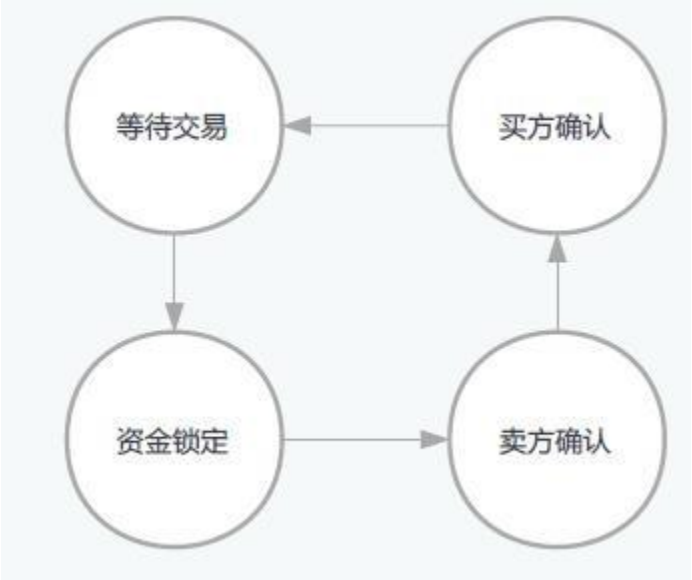
我们认为 UTXO 相对于账户余额模型，并发性能更好；但是在扩展性以及使用便捷性上，账户余额模型表现得更好。

在ekva 项目中，出于对性能和扩展性、便捷性等的综合考虑，我们采用账户余额模型来实现账本的存储。

2.2 抵押冻结担保交易模式

ekva 通过在链上实现担保交易协议，为交易行为提供了去中心化的第三方保证服务，最大程度地保证“一手交钱，一手交货”的交易过程，保护买卖双方的 利益。

担保交易的简单执行过程如下：



(1) 卖方在区块链上挂单，输入售卖资产的信息，包括资源 ID、特征值、卖方帐号、金额等

参数，等待需求方发起交易。

(2) 资产需求方在链上合约中转入指定的金额，合约检查转入金额是否满足售卖要求：若检 查通过，则记录该事件，资产进入资金锁定状态；若检查未通过，则向转入者反馈错误 信息，进入等待交易状态。

(3) 卖方提供数据后，向合约发起确认，并设置有效期。若有限期结束时仍未执行其他操作， 则合约自动进入结算过程。

(4) 买方收取资产完成后，向合约进行确认。

(5) 合约将资金转到卖方账户，过程结束。

以下通过一个具体的案例来描述 Alice 售卖 a 个单位的资产给 Bob 的解决方案：

首先 Alice 创建一个随机的字符串 x，计算哈希值 h=H(x)，Alice 创建一笔可以退还资产的交易 A，但并不公开，Alice 将交易 A 发送给 Bob，让 Bob 签名。一旦 Bob 对交易 A 签名，Alice 公开交易 A。

Bob 创建可以退还数字货币的交易 B，但并不公开，Bob 将交易 B 发送给 Alice，让 Alice 签名，一旦 Alice 签名，Bob 公开交易 B。

情景 1：Alice 按照计划完成兑换，把 x 给 Bob，流程结束。

情景 2：Alice 改变主意，想要终止交易，Alice 只要不向 Bob 公开 x 的值即可。

2.3 交易者的隐私保护

基于区块链技术的信用评价系统的建立，以及抵押冻结担保交易模型的实际可用，都依赖于 对交易者的隐私保护。

在区块链网络中，无论是防串谋的需求还是诚实投票的需求，都需要实现一套保护隐私的安 全投票和验证的协议，来解决验证投票环节中第三方攻击导致的数据泄露以及投票过程未匿

名化的问题。

目前在区块链领域的隐私性解决方案主要有以下一些：

混币方案，这也是起源自早期的基于比特币或以太币的混币服务，后来在门罗币种有使用到， 其目的主要是要让每个地址都要经过搅拌器和许多其他地址进行混合，从混币器出来的交易，除了混币器知道外，别人谁都不知道这个币真实是由谁转给谁的。

直接存储和处理加密数据的方案，这在有些联盟链中使用的比较多，只有交易的参与方才能 解密数据。该方案相对较为可靠，隐私性能得到密码学的严格论证，但缺点是验证节点无法 对交易数据做验证。

零知识证明，零知识证明是一个非常强大的现代密码学工具，在零知识证明算法中，具有私 有信息的一方在不泄露自己私密信息的情况下，能让验证方正确的确认证明方的信息是对 的。不过基于 ZK-Snark 的非交互式零知识证明技巧在业界已表现出效率方面的问题。当前在一台普通的计算机上创建一个 ZK-Snark 的零知识证明大约需要超过 90 秒的时间。另外基于 ZK-Snark 算法的延展性比较差，不能动态创建安全验证参数，必须提前设置一些安全参数，这在实际使用中也会带来一些问题。

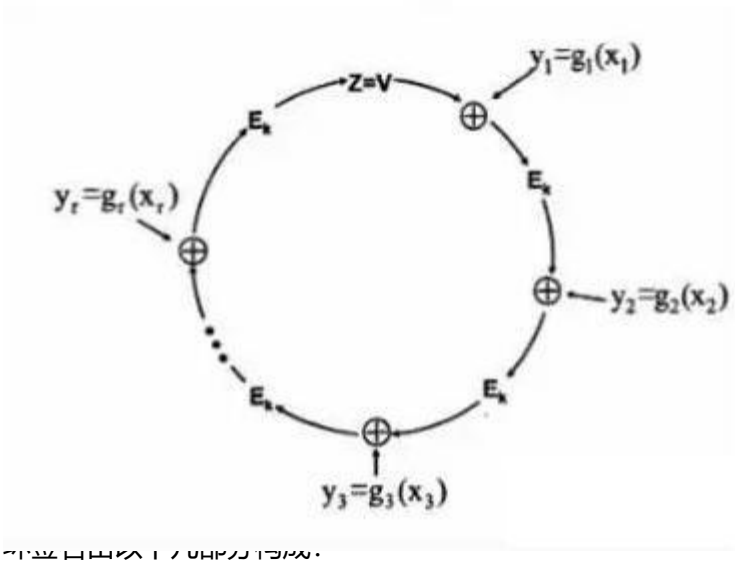
环签名方案，该方案实现了一种无条件匿名性的要求，签名者可以自由指定自己的匿名范围， 外部攻击者即使非法获取了所有可能签名者的私钥，他也不能确定出真正的签名者。很明显， 环签名方式主要是通过匿名的方式来实现隐私性保护。

通过对以上主流的隐私性解决方案的分析，结合ekva 项目的实际解决隐私性的需求，我们采用环签名的方案来保护交易的隐私性。

环签名是一种签名者模糊的签名方案。在环签名中不需要创建环，也不需要分配指定的密钥， 无法撤销签名者的匿名性，除非签名者自己想暴漏身份。环签名方案中签名者首先选定一个临时的签名者集合，集合中包括签名者自身。然后签名者利用自己的私钥和签名集合中其他

人的公钥就可以独立的产生签名，无需他人的帮助。

环签名没有可信中心，对于验证者来说，签名人是完全正确匿名的。环签名提供了一种匿名　泄漏秘密的巧妙方法。环签名的这种无条件匿名性在对信息需要长期保护的一些特殊环境是　非常有用的。



环签名由以下几部分构成：

（1）　密钥生成。为环中每个成员产生一对密钥对（公钥 PK_i 和私钥 SK_i ）。

（2）　签名。签名者用自己的私钥和任意n 个环成员（包括自己）的公钥为消息 m 生成签名 a。

（3）　签名验证。验证者根据环签名a 和消息 m 验证签名是否为环中成员所签，如果有效则接收，否则丢弃。

同时环签名必须满足以下特性：

（1）　无条件匿名性。攻击者无法确定签名是由环中哪个成员生成，即使在获得环成员私钥的情况下，概率也不会超过$1/n$ 。

（2）　正确性：签名必需能被所有其他人验证。

不可为造性：环中其他成员不能伪造真实签名者签名，外部攻击者即使在获得某个有效环签

名的基础上，也不能为消息 m 伪造一个签名。

三、共识机制

3.1 共识机制简介

分布式网络的核心难点在于如何高效地达成共识，现在的社会系统，中心化程度高、决策权集中的社会更容易达成共识，但是容易形成独裁和专制，社会的整体满意度较低。对于中心化程度低的、决策权分散的社会却更难达成一致，像民主投票，但是社会的满意度更高。按照分布式系统的不可能性原理，任何基于网络的数据共享系统，数据一致性、数据的高可用性、分区容忍性是不可能同时满足的，如何在一致性、可用性和分区容忍性之间进行平衡， 是研究共识机制的目标。

区块链作为一种按时间顺序存储数据的数据结构，可支持不同的共识机制，区块链共识机制 的目标就是要使得所有的诚实节点保存一致的区块链视图，必须满足两个基本条件：

A 一致性：所有诚实节点保存的区块链的前缀部分完全相同。

B 有效性：由某诚实节点发布的信息终将被其他所有诚实节点记录在自己的区块链中。

一般的共识机制在满足一致性和有效性的同时会对系统的整体性能产生不同程度的影响，综 合考虑各个共识机制的特点，一般会从以下 4 个维度评价共识机制：

A 安全性：即是否可以防止双花、自私挖矿等攻击，是否有良好的容错能力。其中最主要的安全问题就是要如何防止双花问题。自私挖矿主要指矿工通过选择适当的策略提高自己产 生出块的概率，这是一种影响公平性的理论攻击方法。

B 扩展性：即是否可以支持网络节点扩展，扩展性是区块链系统设计要考虑的关键因素之一，根据对象不同，扩展性又可以分为系统成员数量的增加和待确认交易数量的增加两部分， 以及随之带来的系统负载和网络吞吐量的变化来衡量。

C 交易性能：从交易发起到被记录到区块链中并被最终确认的时间延迟，也可以理解为系统每秒可以有效确认的交易数量。

D 资源消耗：在达成共识过程中，系统所消耗的计算资源，包括 CPU、内存、硬盘等。总之，区块链解决了在不可信的信道上传输可信信息、价值转移的问题，而区块链系统中的共识机制解决了如何在分布式场景中达成一致性的问题。

对目前主流区块链系统共识机制的研究，我们可以将共识机制分为工作量证明（POW）、权益证明（POS）、授权股份证明（DPOS）、基于领导者的共识（PAXOS 及其衍生方案）、PBFT 协议系列、DAG（有向无环图）等六种主要类型。

3.2 存储证明机制介绍

这里我们需要再介绍另一种非常重要的有效工作量证明机制，叫存储量证明，有时也被称为 可恢复性证明。不同于一般工作量证明机制中需要矿工单独计算一个解谜算法，而是设计为 一种需要存储大量数据被运算的解谜算法。

首先我们讨论一个文件 F （可以是基因库数据也可以是大型强子碰撞的实验数据），我们假设所有人都认可 F 的价值并且这个文件一旦确定之后永久不会发生改变。例如，当一份重要的数据文件被发送的时候，由一个可信任的发送者选择 F ，这有点类似于任何一个加密数字货币系统由矿工挖出一个区块，理想情况下，这个文件具备了公共价值。

当然，因为 F 的存储是碎片化的，无论是从隐私安全保护上的考虑还是基于大文件的考虑，大多数参与者都无法对整个文件进行存储，所以存储量证明机制的核心在于，在不需要了解整个文件的情况下，如何使用密码学里的哈希函数来确保每个节点都对 F 的认可。最简单的方法是，每个人都认可 $H(F)$（哈希函数），但更好的方法是用一个大型的 Merkle 树来代表 $H(F)$，所有的参与者都认可 Merkle 树的根值。接下来的问题就是假设每个人都认可

F 的价值，如何证明 F 的任意一部分是正确的。

在ekva 系统中，每个矿工$\mathbb{M}$ 存储着任意 F 文件的子集$\mathrm{F}_M \subseteq F$ 。为了实现这一点，当矿工产生一个公钥 P_M来接收挖矿的回报的时候，他们就对该公钥进行哈希运算以生成一个F_M的虚拟随机数集，他们必须存储这个随机数集以实现挖矿的目的。这个子集就会变成某个固定数量的碎片的一部分，在这里，我们做一个假设，当矿工开始挖矿的时候，他们有办法获得这些碎片，也就是必须存在一个标准文件的源地址，首先，源文　件所在的节点必须在线且正常提供服务。

一旦矿工在本地存储了F_M，这个解谜算法就非常类似于传统的 SHA-256 挖矿的解谜算法了。给定前一个区块的哈希值 x 时，矿工选择一个临时随机数 n ，将其进行哈希运算并产生一个虚拟随机数子集$\mathrm{F}_{M,n} \subseteq F_M$，这个子集包含了 $k_2 < k_1$ 个区块。需要注意的是，这个子集是由所选的临时随机数和矿工的公钥共同计算产生的。最后，矿工对 n 以及 F_k中的区块， 进行 SHA-256 的哈希函数运算，如果计算的结果低于目标难度的，那么也就意味着他们找到了一个有效的存储方案。

检验一个存储量证明的解谜算法的结果需要以下几个步骤：

（1） 检验$\mathrm{F}_{M,n}$ 是由矿工的公钥 P_M和临时随机数 n 共同产生的。

（2） 通过检验其在 Merkle 树节点到全局统一的树根路径，来检验$\mathrm{F}_{M,n}$ 中的每一个碎片是正确的。

（3） 检验 $H(\mathrm{F}_{M,n} \| n)$ 的值比目标难度要小。

我们很容易看出，之所以存储工作量证明的解谜过程需要矿工在本地存储所有的$\mathrm{F}_{M,n}$，对于每一个临时随机数，矿工都需要计算$\mathrm{F}_{M,n}$中随机子集的哈希值，如果通过远程访问一个存储空间来获取文件，就会非常慢，而且几乎是不可能实行。

如果矿工仅仅在本地存储了一半的 F_M，那么在他们找到一个不需要从网络中取回任何文件

碎片的临时随机数之前，他们必须要尝试很多次，降低一定量的存储负担会以计算量指数型 增长为代价，当然，由于 k_2 的 Merkle 树路径要在所有的路径中被传输和检验，如果 k_2 设计的太大，也会使得运算变得非常低效。

k_1 的设定也可以有所权衡。更小的 k_1 意味着矿工需要更少的本地存储空间，因此这种挖矿机制就更加民主化（更多的人可以参与）。然而，这也意味着，大量的矿工即使有能力提供 更大的存储空间，他们也没有动力去存储多于 k_1 个 F 的碎片。

当然，仅仅通过存储部分文件块而不是直接存储整个文件，这是一个效率及隐私与可靠性之 间的权衡问题，如果随机数足够随机而且具备不可控制与不可预测的特性，从而让恶意的矿 工无法判断具体的分布，实际中的系统会对撒谎行为进行惩罚，这里只是做理论的分析，对 具体的实现不再展开。

事实上，对于矿工而言，最保险的方案，就是所有矿工存储所有文件的全部而不是碎片，这 对于在早期节点不太多的时候，从系统稳定性上考虑，可以采用的方案。

3.3 复制证明与时空证明

与存储证明不同，复制证明与时空证明（Proof of Space Time）是由 FileCoin 最早引入区块链领域的技术，时间空间的定义是衡量并计算存储在网络中的数据存储的空间占有和时间 持久性，FileCoin 是通过提供去中心化的分布式存储，矿工通过提供空间和检索服务获取客户的付费，并通过时间空间证明算法获得出块奖励。

复制证明是一种新型的存储证明机制，它允许服务器（证明者 P）说服用户（验证者 V）一些数据 D 已经被复制到它唯一的专用物理存储设备上。其方案是一种交互式的协议。复制证明方案允许用户请求检查存储提供商当时是否已经存储了外包数据，为了能证明数据在一段时间范围内都被存储了，一个很自然的方案就是要求用户重复周期性的（如每个几分钟）

对存储提供商发送验证请求。然而每次交互所需要的通信复杂度都会成为系统的瓶颈，因为 存储提供商被要求提交证明到区块链网络中。

为了解决这个问题，FileCoin 提出了时空证明的方案，它可以让验证者检查存储提供商是否在一段时间内存储了他的外包数据，这对提供商的直接要求是：一是生成顺序的存储证明（也即复制证明）来作为确定时间的一种方法；二是组成递归执行来生成简单的证明。

3.4 POC+POST 的混合共识机制

ekva 项目在 POC 底层共识的基础上，通过建立二级存储网络模型， 将 POC 共识所依赖的哈希值文件转换成对社会有实际价值的文件存储，这也是ekva

项目所独创的 POC+POST 混合共识机制，该机制兼具 POC 共识的去中心化、高安全性、节能环保的特性，有具备着实用的社会价值。

另外通过二级存储网络的建立，将不限地域、不限数量的基于内容寻址的分布式存储网络进 行嵌入，通过主链的空间算力提供去信任化的基础环境，以实现对分布式存储网络资源的顶 层索引、资源调度以及相应的数据交易行为提供撮合和记录，可支持但不仅限于以下应用场 景：

（1） 大量闲置的硬盘存储资源的记录。

（2） 对于硬盘存储资源的调度和管理。

（3） 基于实际内容存储的可寻址的分布式网络。

（4） 可信的去中心化数据存证环境。

（5） 可信的计算环境。

（6） 基于去中心化服务的分布式存储任务调度结算。

在内容寻址网络的构建上，采用一种改进的 Kademlia DHT 算法，显然直接在 DHT 中存储

数据块，会浪费存储和带宽，因为数据必须存储在不需要的节点上，我们在以下几个方面做 出重要改进：

（1） Kademlia 算法将值存储在 ID 最接近的密钥值的节点上，而没有考虑应用程序本身的数据的相似性，忽略了可能的最远或最近的节点。我们是将地址存储到可以提供数据块的节 点上。

（2） 在我们的网络中，只将值得子集分发到最近的节点上，避免热点过于集中。

（3） 我们会根据区域和大小组织一个称为群集的独立 DHT 层次结构，让节点能够首先查询其区域中的对等点，查找附近的数据而不是查询远处的节点，从而大大减少查找的延迟。

3.5 防自私挖矿和截留攻击的机制设计

从经济学角度来看，加密数字货币中的激励机制解决了挖矿者的参与动机的问题，而激励机制的细节设计则直接关系到系统的安全性。其中加密货币中最重要的机制就是挖矿，无论对于何种共识机制来说，矿工想要获得加密货币就需要解决特定的数学难题，也即，找到一个区块，矿工就可以获得对应的奖励。无论对于传统的工作量证明机制还是存储量证明机制， 矿工解决这些数学难题需要具备一定的计算能力或存储能力，通常情况下单个矿工需要花费数月甚至数年才可以成功挖矿到一个区块。而实际上，整个区块链网络可能数分钟（在ekva 中，设置大约 3 分钟）就会出现一个新的区块，所以绝大多数矿工都会徒劳无获。为此，部分矿工组成矿池，将他们的计算能力作为一个整体，如果在合适时间内挖到一个有效区块，他们就按照每个人的计算能力分享挖矿所得的奖励。因此，从博弈论的角度来看，如果激励机制设计有漏洞，就会导致偏离矿池策略能够带来更大的收益， 理性矿工都有偏离矿池策略的动机。

我们认为自私挖矿是针对所有竞争性共识协议的一种通用攻击方式，POC 共识机制作为竞

争性共识机制的一种，也不可避免存在着自私挖矿攻击。于是，我们提出一种新的难度调整 公式，它虽然无法消除自私挖矿的可能性，但即使在发生难度调整的情况下，自私挖矿相较 于诚实挖矿也不会具有优势。

基本上，所有的自私挖矿攻击都是利用了难度调整规则，当前的竞争性共识协议比如比特币 协议低估了网络中的实际算力，由于其仅考虑了被纳入区块链的区块，在存在自私矿工的情 况下，孤块会不断增长，大量诚实算力会丢失，网络用于验证的平均时间会增加，自动完成的难度调整规则会无视孤块的产生，尽管网络总算力是保持不变的，但新的难度会低于应有 的水平，且区块验证时间会减少。所以，单位时间的自私挖矿收入会增加，从而攻击变得有 利可图。

为了减轻这种类型的攻击，我们的想法是在难度调整公式中加入孤块数量的因素。只要诚实 矿工发出孤块信号就足够了，节点不需要广播整个孤块，只是广播他们的区块头部。这会激 励矿工在他们的区块中纳入叔块的存在性证明，在两个具有相同高度的区块之间竞争的情况 下，节点应该始终广播拥有最多容量证明的区块。这一规则在诚实矿工与自私矿工发生竞争 的时候，对诚实矿工是有利的。新的难度调整公式如下：

$$D_{new} = D_{old} \frac{(n_0 + n')\tau_0}{S_{n_0}}$$

其中 n' 是在这段时间内挖矿的孤块总数，而 S_{n_0} 是网络用于验证 n_0 区块的时间。

四、安全性分析

4.1 系统性安全分析

首先，生成攻击（Generation Attack）是一类针对存储证明类型网络的攻击形式，恶意参与节点可以通过高效率低成本的软件，按需生成大量的存储资源，从而欺骗网络的节点验证， 从而使得恶意矿工能获得大量的区块收益。

在容量证明的生态网络中，矿工最为重要的资源就是存储设备，而如何避免恶意节点通过生成攻击，形成答案而取提交答案，从而造成存储证明生态的不公正行为，是每个存储证明网络需要考虑的重要攻击场景。

在我们的网络协议中，我们通过对难度值的调整，将预期的出块时间和存储资源做出适当的调整，从而保证即使拥有数百 PB 算力的网络来说，要通过生成攻击，也是几乎无法做到的事。

另外，对于任意竞争性的共识机制来说，自私挖矿攻击都是非常危险的。自私挖矿攻击是指恶意参与节点在不被主链发现的前提下，秘密地通过互相验证并出块，构建一条私链。在恰当的时机下，其合并到主链，创造一笔双花交易。我们通过在难度调整规则中引入一些变量来防止自私挖矿攻击，从而减轻自私挖矿对系统的危害。

4.2 随机数的安全性

基于分布式存储系统和时间空间证明，为了有效防止串谋合作以及恶意破坏系统，系统必须依赖一种不可控制的随机预言机。可验证随机函数（VRF）算法作为一种基于密码学的新型算法被提出，其最大的优势是快速计算、抗攻击能力、极低的算力需求，目前业界已问世的可验证随机函数的解决方案由图灵奖获得者 Micali 提出，而后经过若干次的改进，已经在一些系统中使用。

这里首先简单介绍一下可验证随机函数的工作原理，先理解一下本节中的“随机”的含义：一个理想的哈希函数，其值域应该是离散的、均匀分布的，给定不同的输入值，其输出值应该没有规律，随机的洒落、分布在值域区间内。

我们再看一个等式$out = f(key, in)$，其中key是密钥，那么要得到结果out，仅仅有输入in是不够的，必须要知道密码 key 才能计算出来，或者说我们已经拥有了结果out 和 in，但

是必须知道 key 才能验证 out 和 in 是否对应匹配的，这就是带密钥的哈希函数。这里引出一个问题：在没有出示密钥的情况下，是否可以验证out 和 in 是否匹配呢？这就是可验证随机函数（verifiable random function），简单地说，这是结合了非对称加密技术的哈希函数，具体操作流程如下：

A 证明者生成一对密钥， pk （公钥）、 sk （私钥）。

B 证明者计算$out = F_{sk}(in)$ 。

C 证明者计算 $proof = P_{sk}(in)$。

D 证明者把out 和 $proof$ 发送给验证者。

E 验证者计算$out = VRF_proof(proof)$是否成立，若成立，继续后面的步骤，否则终止流程。

F 证明者把 pk 和in 发送给验证者。

G 验证者计算$VRF_verify(pk, in, proof)$，结果为 True 表示验证通过，结果为 False 表示验证失败。

所谓的验证通过，就是指 $proof$ 是否是通过in 生成的，通过 $proof$ 是否可以计算出out，从而可以推导出in 和 out 是否对应匹配，证明者给出的数据是否有问题。在整个操作流程中，证明者始终没有出示自己的私钥 sk，验证者却可以推导出in 和 out 是否对应匹配，这就是 VRF 的设计核心所在。

现在我们再回到存储数据的随机选择验证的问题上，在ekva 的网络协议的设计中，有一个基础性的假设：网络中诚实算力的数量始终占优。另外由于系统底　层是基于容量证明的严格公开的网络，所以我们对安全性的分析主要集中在二层网络数据的　完整性验证上进行。对于验证矿工节点和被验证矿工节点以及被验证数据的片段，我们是依　赖一个双向随机选择的机制来进行，以防任何一方或多方合作串谋。我们主要从以下几点简

要描述一下整个系统的随机性与安全性。

A 矿工可以通过对上一区块哈希计算来确定对于验证参数的选择，但由于哈希函数自身性质，攻击节点只需要在区块中添加一个微小的改动就可以很大程度地影响验证参数的选择，从而破坏验证参数的随机性。

针对这种情况，在协议中，所有涉及数据完整性验证的参数的计算取决于未来两个区块的生 成结果，由于任何矿工节点无法完全控制两轮参数的情况，所以对于任何矿工节点想要完全 控制随机参数的选择是不太可能的。另外，通过引入节点质量的参数，节点质量也影响着区 块的质量，也即节点的质量（由其在系统中的信用等级决定）越高，其成功出块的概率也会 越高。

B 即使验证节点的选择是完全随机的，攻击者也有可能在验证节点发布的时候，马上攻击这些节点，从而控制验证节点的选择。

针对这种攻击方式，我们采用的方案是每轮验证的时候设计多个潜在的验证者，也即满足某种条件的验证者并不是唯一的，而每个潜在的合格的验证者是没法提前预知其他验证者的存在，然后每个潜在合格的验证者都将自己的认证信息、验证信息、验证结果发送到网络中， 即使攻击者攻陷了某个潜在验证者，其成功攻陷所有潜在验证者的可能性也是几乎不存在的。

另外需要强调的是，每次验证者和被验证者以及验证的数据片段是经过双向随机选择的结果，这也完全杜绝了串谋作弊的可能性。

五、技术细节

5.1 可恢复性证明

在上面的介绍中，我们只谈了文件的碎片化的分存问题，没有涉及到文件的冗余备份和恢复

机制，理论上，同一块文件会在矿工那里等概率地被保存，只要存储矿工节点足够多，便不存在文件丢失地问题，但是如果所有保存某个碎片的矿工结成联盟勒索文件属主，事实上这个问题是可以通过密钥分存（在现代密码学里称为秘密分享技术）方案来解决的。 Shamir 和 Blakley 于 1979 年分别独立地提出秘密分享的概念，并给出了(k,n)门限秘密

分享方案。在秘密分享方案中，用户将需共享的秘密分成若干秘密份额也称子密钥、碎片，并安全地分发给若干参与者掌管，同时规定哪些参与者合作可以恢复该秘密。之后 Chor 等人在 1985 年提出了可验证秘密分享的概念。可验证秘密分享考虑到基本秘密共享方案中秘密分发者与参与者可能是不诚实的，它在基本秘密共享方案的基础上，增加了一些公开承诺和验证算法，来检测试图伪造秘密份额的用户，包括秘密分发者和参与者。

一个(t, n) 可验证秘密分享方案需要满足两个要求：

A 可验证性：在收到一份秘密份额后，用户能够测试它是否是一个有效的份额。如果一个份额有效，则存在一个唯一的秘密作为秘密重构算法的输出，秘密重构算法作用在任意t个有效份额上。

B 不可预测性：对于多项式时间算法，输入$t-1$个秘密份额，不能获得任何有关秘密的信息。

可验证秘密分享的一个典型例子是 Feldman 在 1987 年提出的一个非交互的方案，它通过增加一个公开验证函数来扩展 Shamir 的方案，是第一个不需要可信机构参与的非交互式可验证秘密分享方案。该方案由四部分组成：系统参数、秘密分发、验证算法和秘密重构：

(1) 系统参数：p是一个大素数，q为$p-1$的一个大素数因子，g为q阶元，三元组(p,q,g)是公开的，t是门限值，n是参与者数量，s是要共享的秘密，秘密空间与份额空间均为有限域$GF(p)$。

(2) 秘密分发：随机选择一个$GF(p)$上的$t-1$次多项式$f(x)=a_0+a_1x+...+a_{t-1}x^{t-1}$，满

足 $f(0)=a_0=s$，然后计算各秘密份额 $s_j=f(x_j)(\mathrm{mod}\ q)$ 并秘密地发送给参与者，其中 $j=(1,2,...,n)$，同时公开函数 f 的系数的承诺 $c_i=g^{a_i}\ \mathrm{mod}\ p$，其中 $i=(0,1,2,...,t-1)$。

(3) 验证算法：各参与者在收到秘密份额后，验证 $g^{s_j}=\prod_{i=0}^{t-1}c_i^{x_j^i}\mathrm{mod}\ p$ 是否成立，若成立则份额有效，否则说明收到的秘密份额不正确。

(4) 秘密重构：当 t 个参与者 $P_1,P_2,...,P_t$ 合作恢复秘密时，每一个参与者 P_j 公开他的份额 s_j 给其他合作者，每个合作者通过执行验证算法判断秘密份额的有效性。由拉格朗日插值 公式算出多项式函数 $f(x)$，最后计算函数值 $f(0)$ 即为秘密 s。

对于ekva 系统来说，每一个矿工都不是存特定的文件块，而是存特定文件块的部分，这么做的目的就是，假设有 m 个矿工存储了文件块 F_i，存在自然数 n（$n<m$），只需要n 个部分就能恢复文件块。

显然，通过秘密分享的技术方案大大增强了整个存储系统的容错性，文件属主不再依赖于特定的矿工，同时任何一个矿工均无法独自恢复文件块，这会在所有矿工之间形成心理博弈， 从而也很难形成联盟。

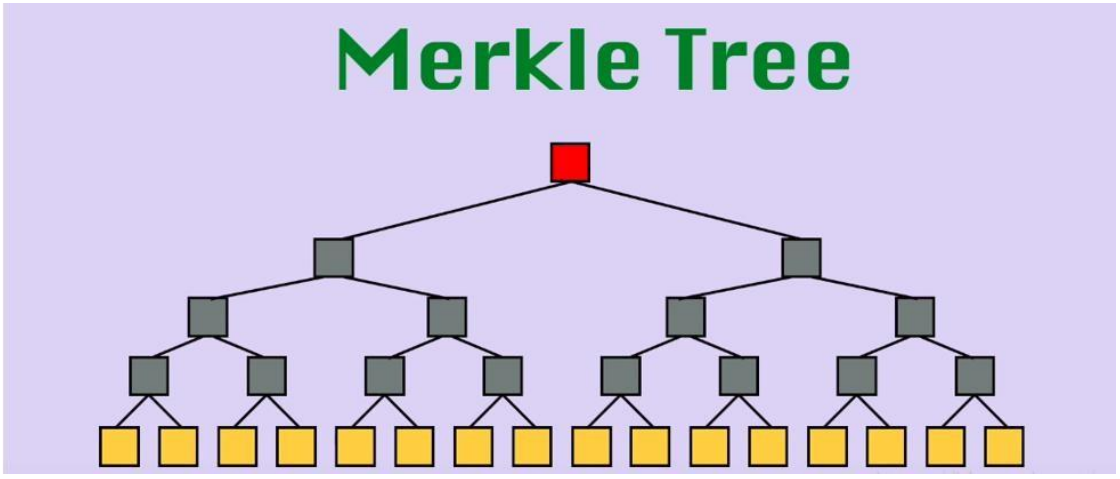
5.2 数据完整性证明

在前面的章节中，我们深入讨论过存储量证明机制和冗余恢复机制，去中心化存储的核心还 需要保持数据的完整性证明。其实，数据的完整验证机制可以分为数据的持有性证明和数据 的可恢复性证明。

数据的持有性证明就是要验证不可信的矿工是否正确地持有数据，避免存储服务提供者恶意 删除或篡改数据。

矿工为了证明自己拥有某个文件块，在网络上直接传输该文件块本身可以有效提供证明，然

而这是一件效率极低地事情，而 Merkle 树利用哈希函数地抗碰撞特性可以解决这个问题，以下是一颗典型的文件 Merkle 树结构。



红色是根节点，灰色是非叶子节点，黄色是叶子节点。叶子节点的数值是根据数据块的值作 为参与，经过哈希运算得出，而灰色节点和红色节点的值是以两个孩子节点的哈希值作为输 入，经过哈希运算得出。

与直接传输文件块不同，有了 Merkle 树之后，我们传输文件块的哈希值即可。只要可以沿着从叶子到根的路径上全部通过，则可以证明自己拥有该文件。以下的分析为了简单起见， 我们假定，文件不再分割成文件块。

在去中心化存储中，为了完成数据持有性证明，也即如何辨别存储者是否真的存储了文件，还是提前算好了哈希值并保存后把源文件删除（用以节省自己的存储资源），这就需要一个挑战-应答机制。

首先挑战者向应答者发起一个关于文件 F 的挑战。

设文件 F 被分割成 N 块， s 为集合 $\{1,2,...,N\}$ 中随机的任意值， r 为任意随机数，则挑战的问题为：

$$challenge = (F[s] || r)$$

如果应答者真的保存了这个文件，那么他就可以拿出 $F[s]$ 文件块的哈希值，如果他在之前

私下保存了所有文件块的哈希值之后把文件删除了，此时挑战参数中的随机数 r 就派上了用场，挑战问题不仅要求应答者回答 $F[s]$ 文件块的哈希值，还要求其做如下运算：

$h = H(F[s]||r)$

也即应答者需要计算文件块与随机数拼接后的哈希值，这样一来，即使应答者保持了文件块　的哈希值也是没有用的，随机数 r 会打乱一切，除非应答者真正地保存了 $F[s]$ 文件块，否则，应答者是不可能计算出其哈希值地，，也即应答者需要应答：

$response = (h \parallel H(F[s]))$

挑战者在收到应答者的应答后将会做一系列的检查。首先挑战者有整个文件 F 的 Merkle 树，其次他还保留着原始的挑战参数，因此他能很快确定应答者是否撒谎。当然，挑战者可　以在任意时刻发起任意次数的挑战。

值得注意的是，在上述的挑战-应答模型中，有一个非常关键的因素，就是随机数的特性，这里的随机数必须是无法预测与无法控制的随机数生成机制。

在我们的系统中，我们采用 VRF（可验证随机函数）机制来生成随机数。VRF 的目的就是要生成一个真正随机而又无法被预测的值。

5.3 数据检索模型

分布式哈希表（DHT）技术是一种分布式存储方法，在不需要服务器的情况下，每个客户端负责一个小范围的路由，并负责存储一小部分的数据，从而实现整个 DHT 网络的寻址和存储。其中一致性哈希通常是 DHT 的一种实现。

一致性哈希算法是在 1997 年由麻省理工学院提出，指出了在动态变化的 cache 环境中，哈希算法应该满足的 4 个适应性条件：

（1） 平衡性：平衡性是指哈希的结果能够尽可能分布到所有的 cache 中，这样使得所有的

缓冲空间都得到有效利用。

（2） 单调性：单调性是指如果已经有一些内容通过哈希分派到了相应的缓冲中，又有新的 缓冲加入到系统中。哈希的结果应该能够保证原有已分配的内容可以被映射到原有的或者新 的缓冲中去，而不会被映射到旧的缓冲集合中的其他缓冲区。

（3） 分散性：好的哈希算法应该尽量避免不一致的情况发生，也就是尽量降低系统的分散性。

（4） 负载：好的哈希算法应能够尽量降低缓冲的负荷。

从表面上看，一致性哈希针对的是分布式缓冲的问题，但是如果将缓冲看作去中心化系统中 的节点，将映射的内容看作各种共享的文件和数据资源，就会发现两者实际上是在描述同一 个问题。

在我们的系统中，核心数据的索引的主要思想如下：

（1） 将内容索引抽象为$<K,V>$对，其中K是内容关键字的Hash摘要，$K=Hash(Key)$，V是存放内容的实际位置，例如节点的 IP 地址等。

（2） 所有的$<K,V>$对组成一张大的 Hash 表，因此该表存储了所有内容的信息。

（3） 每个节点都随机生成一个标识（ID），把 Hash 表分割成许多小块，按特定规则（即K和节点 ID 之间的映射关系）分布到网络中去，节点按这个规则在应用层上形成一个结构化的重叠网络。

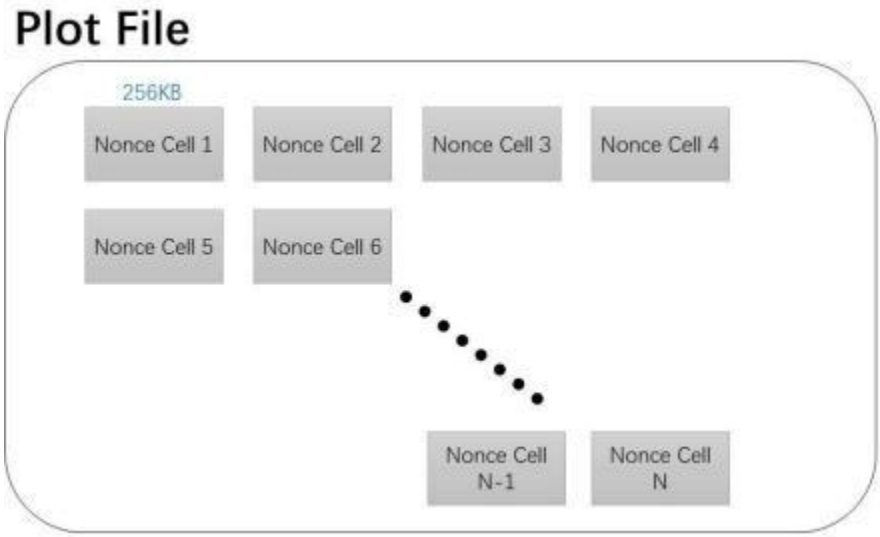
（4） 给定查询内容的K值，可以根据K和节点 ID 之间的映射关系在重叠网络中找到相应的V值，从而获得存储文件的节点 IP 地址。

5.4 Plot 文件生成

按照容量证明共识协议的原理，其采用磁盘空间存储代替内存算力计算的方式挖矿，需要事 先将算好的哈希值存储到硬盘中（这一过程，我们称之为 P 盘），也即在磁盘中生成一系

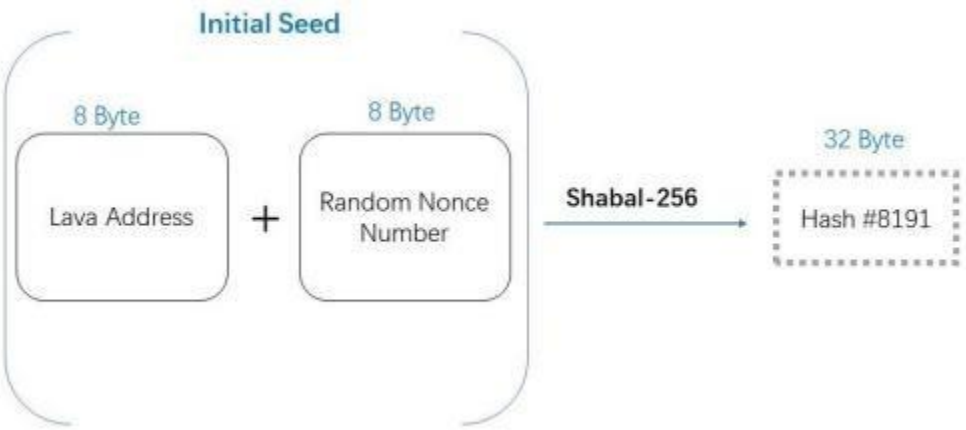
列 Plot 文件的过程。

Plot 文件是针对某一列给定的 Nonce 由一系列哈希运算结果排列而成的数据阵列，每一个 plot 文件都会存储着若干 Nonce，每一个 Nonce 占据 256KB 的存储空间，其中每一个基本的 Nonce 数据区域称为一个 Cell。参与节点的存储空间越大，就能存放越多的 Cell，其成功出块的概率会更高。



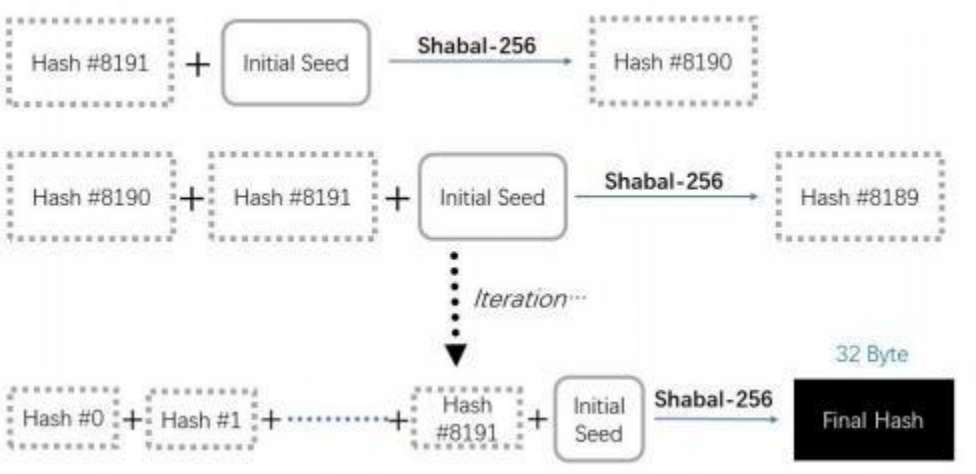
Cell 的生成过程设计 256bit 的哈希函数，这里我们选择的是 shabal256 的哈希函数，shabal256 算法是一种很慢的算法，允许输入任意长度的有序位序列，甚至是一个空序列，也适应任何长度的字节流。Shabal256 算法的设计目标就是保持简单的同时保证其安全性，shabal256 的结构有着良好的次原象攻击抗性的，而且是能够被证明的， 这一点也是 shabal256 算法与其他哈希算法相比的主要优势之一；另外，shabal256 中没有使用其他加密算法中非常流行的 S-Box，它的性能成本非常高，这也是其很适合容量证明共识的一个重要原因。

生成 Cell 的起点是用户地址，将地址与一个随机数种子拼接，构成一个初始种子。然后对初始种子进行一次 shabal256 计算，得到第一个哈希结果#8191。



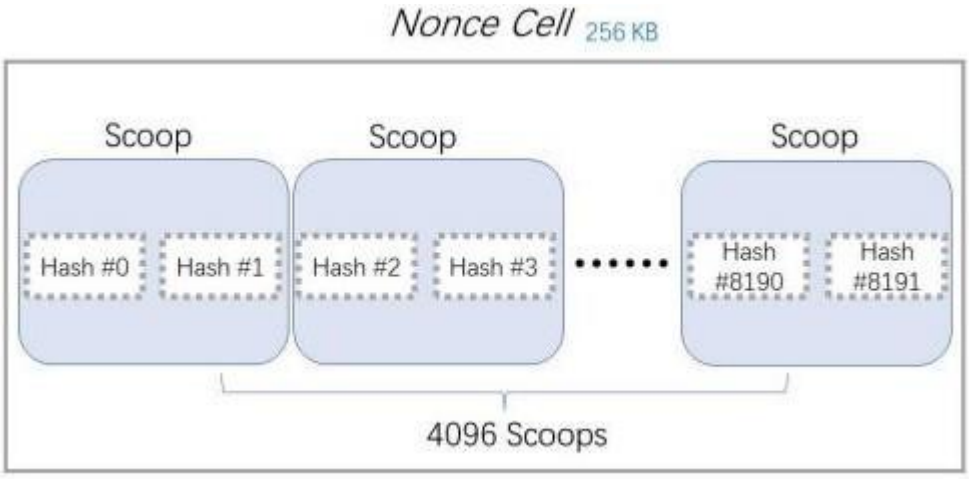
将#8191 添加到初始种子前面，形成一个新的种子，进行 shabal256 计算后得到第二个哈希结果#8189。

依次类推，每一次都是将上一个哈希结果添加在最新的种子前面，直到生成最后一个种子，进行 shabal256 计算后得到最终哈希值 Final Hash。



每一个哈希结果长度都为 32Byte，在进行哈希计算时，一旦种子长度超过 4096Byte，只取最后 4096Byte 长度。

然后，将刚才计算出的 8192 个哈希结果（#8191、#8190、……、#1，#0）分别与 Final Hash 进行异或运算，保存得到的 8192 个结果（编号保持不变）。

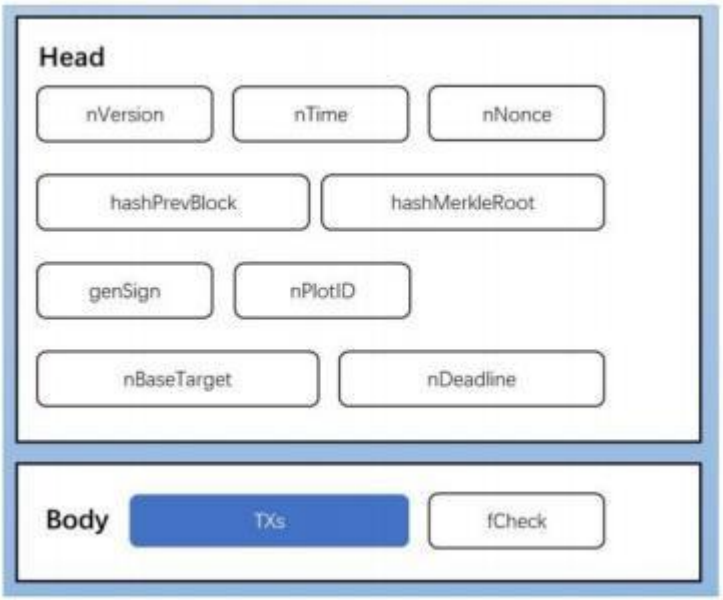


在生成 Cell 的过程中，系统必须利用缓存记录所有的中间结果，才能得到最终的哈希结果。由于每个 Cell 包含 8192 个 Shabal256 的哈希结果，每个哈希结果长度位 32Byte，因此每个 Cell 将占用 256KB 的空间。

反复进行生成 Cell 的操作，再将所有的 Cell 进行优化排列，填满 Plot 文件。至此，Plot 文件的准备工作已全部完成。

5.5 区块数据结构介绍

一个典型的区块数据结构如下



每个区块都由头部Head 和主体Body 两部分构成，其中 Body 里主要就是交易列表的记录。

这里简要介绍一下头部结构种的一些字段。

nVersion：版本号，主要用于跟踪软件、协议的更新。

nTime：时间戳，主要表示该区块产生的近似时间（精确到秒的 Unix 时间戳）。

nNonce：用于容量证明的随机数，用于验证该区块的合法性。

hashPrevBlock：父区块的哈希值。

hashMerkleRoot：该区块种交易的 Merkle 树根的哈希值。

genSign：当前区块的 generation Signature。

nPlotID：矿工的标识 ID，也是在 P 盘过程种使用的 PlotID。

nBaseTarget：代表该区块的难度系数。

nDeadline：该区块的强制性等待时间，代表自上个区块时间戳后，新的区块时间戳需要等待的延时。

5.6 区块生成与验证

DNP 网络分发协议的矿工是通过 POC 容量共识证明机制，通过提供更大的硬盘存储空间容量来提高生成区块的成功率。

DNP 的矿工依赖于静态存储的特殊数据参与区块生成的竞争。这些特殊数据就是以上章节中介绍的 Plot 文件的生成结果。

在挖矿过程中，共识机制通过待产出区块的数据，随机性地指定 Plot 文件数据阵列中的特定位置，参与的竞争者检索自身 Plot 文件中的对应数据，生成一个 Deadline；Deadline 是节点广播新区块需要等待的时间，因此产生最小的 Deadline 成为矿工们的目标，生成最小的 Deadline 也意味着出块成功。

由于 Plot 文件可以一次生成、长期保存并可任意重复使用，而出块过程中所依赖的工作仅

限于网络广播、检索以及简单的验证性的计算，因此容量证明共识机制对于高性能计算资源　以及电力能源的消耗可以被降低到最低限度。

以下详细介绍一次区块生成的详细过程。

矿工在准备好 Plot 文件后，即可开始参与区块的生成与验证工作。

首先，挖矿程序接收被广播的交易消息，验证后打包到区块中并生成对应的根哈希值和状态变化哈希值。然后挖矿程序会生成新区块的高度、父区块哈希值以及 Generation Signature（以下简称为 GenSig），另外计算当前的挖矿难度系数。接着挖矿程序对 GenSig 和高度联合做 Shabal256 哈希运算得到 GenHash，而后以 GenHash 对 4096 取模，得到当前生成区块的 Scoop Number。

接着挖矿程序在本地检索并取出 Plot 文件中所有对应 Scoop Number 的 Scoop 数据，分别后附 GenSig 并做 Shabal256 运算，得到 Target，将 Target 除以一个代表难度系数的参数 BaseTarget，得到 Deadline。

Deadline 是一个强制性等待时间，代表自上个区块时间戳后，新的区块时间戳协议等待的延时，延时没有得到满足之前，生成的区块是不合法的，不会被网络中其他矿工接受。

挖矿程序如果接收到网络广播的新区块，会对该区块提供的数据进行校验。其中校验的数据包括被选中的 Cell 的初始种子以及 Deadline。任何网络节点的挖矿程序都可以在较短的时间内完成对新区块的验证。

六、通证与发行

6.1 通证

ekva 作为分布式网络分发协议，是一个典型的依赖于去中心化社区自治运作的协议。也是一种全新的分布式自治经济组织，其投资回报模式不是股权投资模

式，投资购买通证也并非购买公司股权，而是购买一种通用数字货币及相关分布式服务。ekva 系统借助去中心化存储技术打造一种可信的分布式存储服务， 它有着本地的通证体系，叫做 ekva 。ekva 是生态中唯一的权益通证，也是持有人去中心化文件存储服务的许可证，同时生态中通过支付ekva 做文件存储的方式可以有效地减少垃圾文件，额外地通胀和区块交易费就作为矿工和社群建设者的奖励。

6.2 发行机制

发行名称：ekva 主节点：21

超级节点：151

发布细节：

发行总量	1.9 亿
开发团队	10%。方式：预挖
推广团队	5%。方式：随挖矿的每个块产出
矿工	85%。方式：挖矿

七、路线图

2019.09.06

节点上线

2019.09.15

官网上线

2019.10.30

Beta 版钱包 APP 完成开发并通过测试

开始基于 IPFS 开发分布式存储

2020.04.10

FCP 主网上线，挖矿开启

2020.04.30

矿机销售