豆匣协议技术白皮书

Sharder Protocol - 跨链分布式存储协议

V1.1

官方网站：https://sharder.org

Telegram: https://0.plus/sharder_cn

Twitter：https://twitter.com/SharderChain

Medium: https://medium.com/@SharderChain

官方微信：豆匣快讯

联系邮箱：hi@sharder.org

Github：https://github.com/Sharders

豆匣基金会 foundation@sharder.org

前言

当今世界已经是一个数据驱动的社会。信息技术的发展以及人类生活的智能化带来数据的爆炸性增长。一方面，数据的增长速度远超过了存储空间的增长速度，存储空间的缺口使得相当数量的存储需求没有被满足；另一方面，大量个人及企业的存储空间处于闲置或者未充分利用的状态。当前绝大多数的数据都是用中心化方式存储于大公司的数据中心，由此带来一系列问题：价格昂贵，不能永久储存，易发生数据泄露，窥探隐私，滥用数据等；同时中心化的存储代码不开源，可能潜在致命的漏洞无法被察觉，甚至数据都存在被篡改的风险。更为重要的是具有高安全性和高可用性的云存储价格不菲，定价不够透明。

无论企业还是个人用户手上都有被淘汰、闲置、不饱和的各种存储设备（办公电脑、3.5 寸磁盘、移动 U 盘、移动磁盘等）。豆匣协议（Sharder Protocol）是豆匣团队基于对区块链技术得而理解，在比特币和众多先行的区块链项目基础上提出的分布式存储协议。用于构建高安全性、高隐私性、高可用性、跨链部署的自治分布式存储网络豆匣网络（Sharder Network）。豆匣先进的存储系统不仅可以帮您存储照片和文件等常规数据资料，还可以帮您存储对您来说弥足珍贵的所有东西，例如您的生物数据（包括基因信息、成长记录、医疗数据等），甚至于您的所思所想。我们的愿景是：珍藏您的故事 （Store your story），构建跨链共享存储经济生态改变整个存储世界和人类数据价值融通交换的方式。

目录

1、 概述

豆匣协议（Sharder Protocol）是跨链的分布式存储协议。Sharder 取自数据分片之意，对应中文名称豆匣意为存储数据对象的匣子。当前的各种公链、存储网络、个人节点都可部署或运行豆匣客户端。在豆匣协议中定义了各种对象、数据存取的操作函数、存储校验机制、共识机制、贡献度量化、数据授信机制。同时也对数据加密、数据分片、多链架构、文件系统、智能合约、自由市场、安全性、可用性、伸缩性等方面进行了抽象和设计。

本份白皮书希望让即使没有计算机、编程、数学、区块链背景的读者也能理解豆匣协议是如何构建分布式存储网络。

2、 摘要

愿景 构建全球化的、高安全性、高隐私性、高可用性、跨链部署的分布式存储网络系统（豆匣网络 Sharder Network）。珍藏您的故事 （Store your story），构建跨链共享存储经济生态改变整个存储世界和人类数据价值融通交换的方式。

豆匣币（Sharder 简称 SS） 豆匣协议内置加密数字代币，总量为 5 亿。更多详情可以访问我们的[官网]。

豆匣链（Sharder Chain） 第一个部署了豆匣协议的商用区块链，也是豆匣网络中的第一个豆匣池（Sharder-Pool$_0$），是构成豆匣网络的基石。

豆匣网络（Sharder Network） 各种部署了豆匣协议的豆匣池最终组成了去中心化分布式的豆匣网络。豆匣网络不仅提供高质量低价格的数据服务。还会研发云存（Bean Cloud）、矩阵（Sharder Matrix）、智脑（Sharder Brain）、数据集市(One Fair)等 DApp 应用，围绕数据构建跨链共享经济生态，豆匣终将会彻底改变整个存储世界和人类数据价值融通交换的方式。

豆匣市场（Sharder Market） 围绕存储空间、数据、数字资产的自由市场。

商业应用 云存（Bean Cloud）数据存储、存证和保全平台。矩阵（Sharder Matrix）个人生物数据存储应用。智脑（Sharder Brain）数据安全、数据分布调整、数据分析、数据搜索、数据预警等智能大数据服务。数据集市（One Fair）透明、公开、自由、公平的点对点交易市场。让沉默的数据便捷安全快速地进行融通，形成价值交换。。

代码开源 豆匣协议是个开源项目，我们的 Github：https://github.com/Sharders。

3、 设计原理

节点不可靠假设：一个松散的但有较强鲁棒性的网络组织结构，允许单点故障和短时间内节点处于不可用状态。

所有权和隐私性：数据所有者具有数据的所有权和完全访问权，数据是加密并具有隐私性的。经过所有者授权后其他角色才能访问和使用数据。

可量化的贡献度：参与协议各方的贡献度都应该有相应的量化标准和可被观测的贡献度。比如采用 PoST 和 PoR 作为存储空间和存储时间的量化证明。

最终状态一致性：允许数据对象在不同节点处于不同状态，但其状态能快速收敛获得全网一致性。

可监测和可恢复：能检测整个网络的可用性和数据对象的全网状态，并根据策略一定程度自主修复。

可审计和监管：可在某些特定领域或场景进行一定程度的监管和审计，前提是数据所有者知悉并同意。

可扩展 API：具有很高扩展性和易用性的 API。

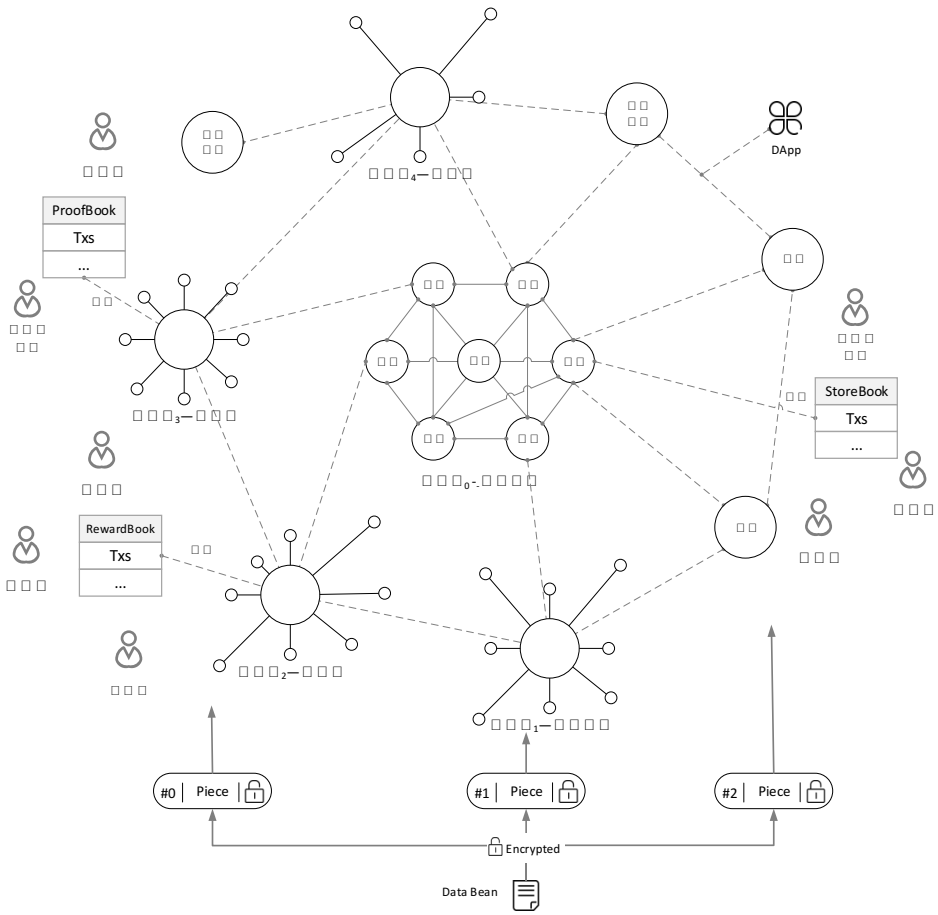
4、 豆匣协议

豆匣协议构建了分布式存储网络，可以提供高性价比的存储空间，安全可信的数据存储，透明公开的上链信息，希望能围绕存储空间、数据、数字资产形成自由市场。豆匣协议允许接入传统的存储资源，更希望让国内外的公链（量子链、以太坊）、存储网络（IPFS、

阿里云、百度云）、个人存储（闲置磁盘、云盘）也能接入豆匣网络。更希望有海量的 DApp 基于豆匣链构建。

4.1 协议概览

- 豆匣协议提倡开放。
- 协议由以下部分构成：角色、网络、数据、贡献量化、奖惩、多链。
- PoR（Proof of Replica）和 PoST（Proof of Storage & Time）作为数据备份和存储时长的量化凭证。
- 数据分片、多备份、数据纠删以保证数据的安全性和可用性。
- 和现存各种网络和公链形成多链生态，完成数据和价值传输。
- Sharder-PAIR 和 Sharder-UTXO 授信框架为满足企业或监管机构的审计和监管需要。



图片 1 豆匣协议概览图

4.2 角色定义

豆匣协议将构成存储网络的各参与者根据职能进行了角色定义。多个角色可以由同一个实体节点承担。全节点需要根据其信用 PoC【3.8.3 信用证明】来进行评定。

节点类型	出块者	观察者	存储者	证明者
全节点	√	√	√	
存储节点			√	
观察节点		√	√	
证明节点		√		√

表格 1 角色和节点关系表

数据豆（Bean） 数据豆（Bean）是未分片（Piece）前的数据对象。除开分片外不能被单独拆分，对外部系统具有唯一性。

数据所有者（Data-owner） 数据所有者是数据豆的拥有者。数据所有者对数据豆进行签名并有权随时进行检查，以确保数据确实安全地进行了存储。

数据所有者可以根据数据的重要性对安全存储提出不同等级的要求，豆匣协议会根据相应的要求选择合适的数据安全策略对数据豆进行存储。并会选取合适的存储节点以保证满足用户的存储要求。当然越高等级的数据安全性数据所有者需要更高昂的数据存储费用。

存储者（Storer） 存储者提供磁盘空间存储数据以此获得相应报酬。存储者需要接受数据所有人或则观察者的随机校验以提供存储证明。比如：接受 PoST 的校验，以证明数据在约定时间内一直存储于磁盘内。下文称呼的“存储节点”是运行了存储客户端的物理节点。

观察者（Watcher） 观察者需要观察和检测整个网络中数据豆的状态，检测存储的数据豆是否满足安全策略，并修复安全缺陷。所以观察者需要稳定在线，也是全网状态快速收敛的必要角色，同时是数据豆索引服务的不二人选。

观察者不定期对存储者进行心跳检测，以确保数据的可用性。还可以接受数据所有者的委托，帮助数据所有者发起数据校验，以确保数据是安全和可用的。基于性能考量，这些工作大多数都在链下完成。我们希望观察者是一个独立的节点，这样可以和出块

节点及数据存储节点形成制衡关系。分散的权利和网络组织结构能进一步确保数据对象的安全性，同时降低网络被恶意攻击的可能。

出块者（Miner） 即是区块链网络中我们通常称呼的“矿工”。出块者需要运行客户端（命令行或则 GUI）保存所有的区块信息，并承担处理交易和打包出块的工作。豆匣网络的稳定性、连通性、吞吐率等性能指标和出块者的内存和计算速度有极大的关系。所以出块者一般由全节点担任，以确保稳定在线、高吞吐和较高的处理效率。只有连入豆匣链(Sharder-Pool$_0$)的全节点才能争夺出块权,出块共识机制会选用 PoS 或 DPoS。

证明者（也称证明人） 为上链数据提供证明使之具有公信力。证明人所提供的证明数据和原始数据对象会关联在一起并记录到链上，具有可追溯和不可篡改性。

证明人多数场景下由现实世界具有公信力的组织或政府机构担任。比如数据版权领域，最具权威的证明人是国家版权总局，证明人节点对接国家版权总局作为版权总局的代理节点存在于豆匣网络中，提供相应的版权证明服务。在数据存证场景中，证明人由公证处和司法机构担任。经过它们进行过公证后的链上数据就具有社会公信力和司法效力。

全节点（Full-Node） 全节点是运行了豆匣客户端，能稳定在线，具有较好的网络带宽和处理性能的物理节点。全节点默认打开了出块者角色，可以根据自身的硬件情况有选择地打开存储者和观察者角色。

豆匣池（Sharder Pool） 由多个部署了豆匣协议的节点组成的小型网络（类似于星际矿池）。现有的公链或存储网络只要部署了豆匣协议也就形成了一个豆匣池。所有节点都应归属于某个豆匣池，节点如果没有显示声明加入特定的豆匣池默认会加入 0 号豆匣池。

豆匣池可以选择不和其他豆匣池连通，形成一个封闭网络系统，类似于私链组成私有云存储网络。豆匣协议不提倡封闭，选择封闭反而需要支付费用。

豆匣网络（Sharder Network） 所有部署了豆匣协议节点组成了豆匣网络。豆匣网络宏观上是个大型的存储系统，希望能形成存储买卖的自由市场，不仅能合理地满足企业和个人用户日益增长的存储需求。同时也能更大限度地有效利用闲置或过时的存储硬件设备，减少电子垃圾的同时为存储提供者带来一定的经济回报。

豆匣链 （Sharder Chain） 豆匣链是第一个部署了豆匣存储协议的商用区块链网络即 0 号豆匣池。豆匣链在豆匣协议中充当了分布式账本的作用，需要永久保存的信息和对象状态都记录在豆匣公链中。同时还充当了多链结构的中间锚定网络。

未来我们希望豆匣链能成为自主和自治的数据公链，能有许多去中心化的商业应用基于豆匣链构建。

豆匣市场（Sharder Market） 豆匣市场是一个去中心化的交易市场，在过去几年比特儿（BitShares）和以德（EtherDelta）向人们证明了去中心化交易所是可以稳定运行的。我们希望豆匣网络中也会演化出由供需产生定价的自由交易市场，早期由豆匣链提供集中竞价和撮合服务，逐渐演变成由买卖双方自行出价，豆匣链作为一个交易记录和价格采集者仅仅提供参考价格和历史成交价，并为交易双方提供智能合约服务。任何一个拥有完整区块信息的节点，都可以提供价格参考和交易撮合。

更远的未来，我们希望提供检查、证明、纠删等数据服务者也能加入自由市场，提供各式各样的数据服务。

4.3 网络拓扑

我们需要构建一个拥有数量众多和随时有节点加入和退出的对等网络。因此一个好的路由表维护和查找算法是重要的。我们优先选择 Kademlia 协议（以下简称 Kad）[1]作为基础来构建 P2P 对等网络（Chord 算法也是备选项）。Kad 以异或算法（XOR）为距离度量基础构建的分布式哈希表（Distributed Hash Table），大大提高了路由查询速度。这对于存在大量存储节点的豆匣网络是非常需要的。Kad 网络的实现也会分为两步，首先我们会构建基于简单路由表的 P2P 网络，在开放存储节点客户端的同时完成 Kad 网络的开发。

Kad 协议中 K 桶的节点列表维护正好符合我们对节点的在线要求，不过未来可能会根据 PoC 中对节点的信用评级来作为排序和换出的一个权重值，以帮助观察者挑选合适的最近节点进行数据分布的调整。

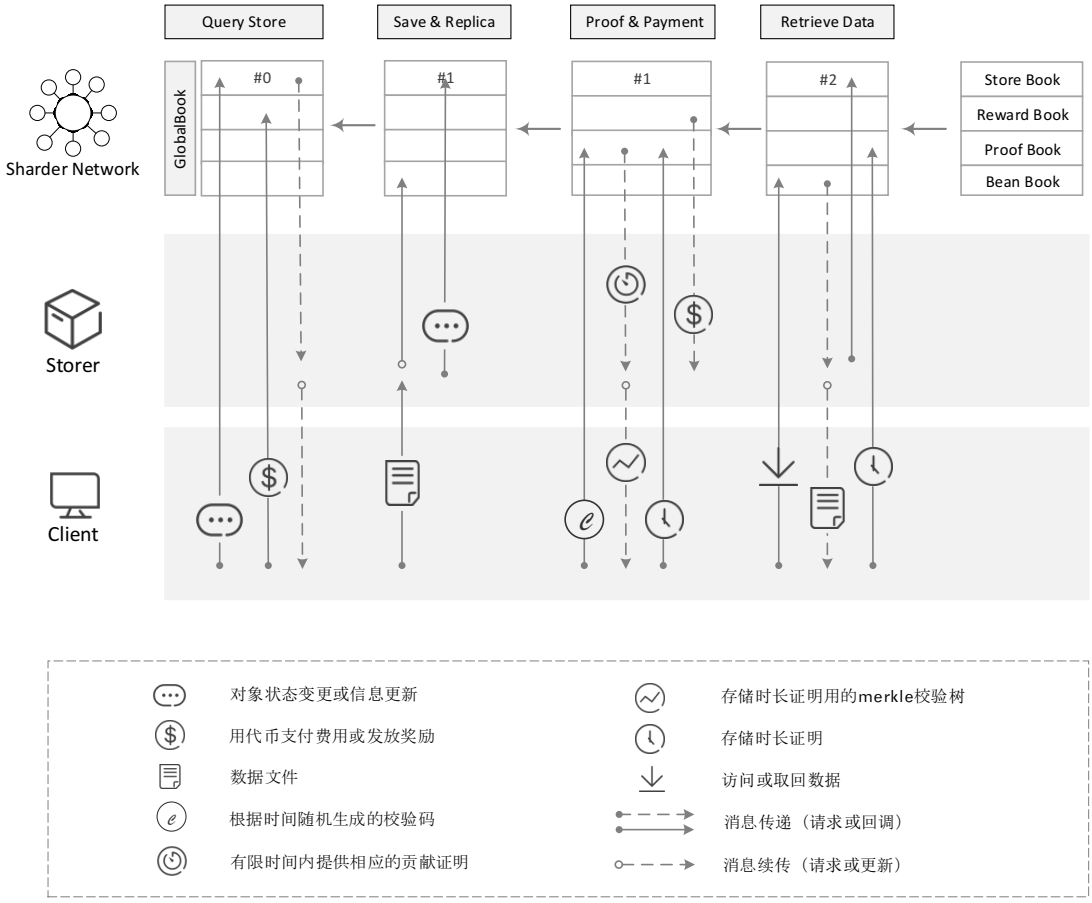
4.4 数据对象操作

$$PRC_{Bean} = (Put, Get, Watch)$$

·Put(data) → key：客户端执行 Put 协议存储数据，key 是这个数据的唯一标识。

·Get(key) → data：客户端使用数据唯一标识 key 执行 Get 协议取回数据。

·Watch()：观察者执行 Watch 协议对已存储的数据进行校验，并同步该数据对象的全网状态。并根据不同安全策略对已发现的数据丢失、数据错误、存储者不可用等异常情况进行修复。



图片 2 数据对象操作说明

4.4.1 数据存储

1、客户端（Client）发起存储数据请求，请求记录到存储账本（Store-Book）。

2、客户端（Client）支付存储费用，豆匣协议返回匹配的存储节点（Storer）。

3、客户端（Client）上传文件到存储节点（Storer）。

4、 存储节点接受完数据后更新存储账本（Store-Book）和数据对象（Bean-Book）的全局状态。

5、 根据安全策略，广播数据备份任务（Replica-Task）到网络。

6、 其余存储节点进行数据备份，并检查是否满足安全策略定义的副本数，未达到的话继续广播数据备份任务到网络。

4.4.2 数据取回

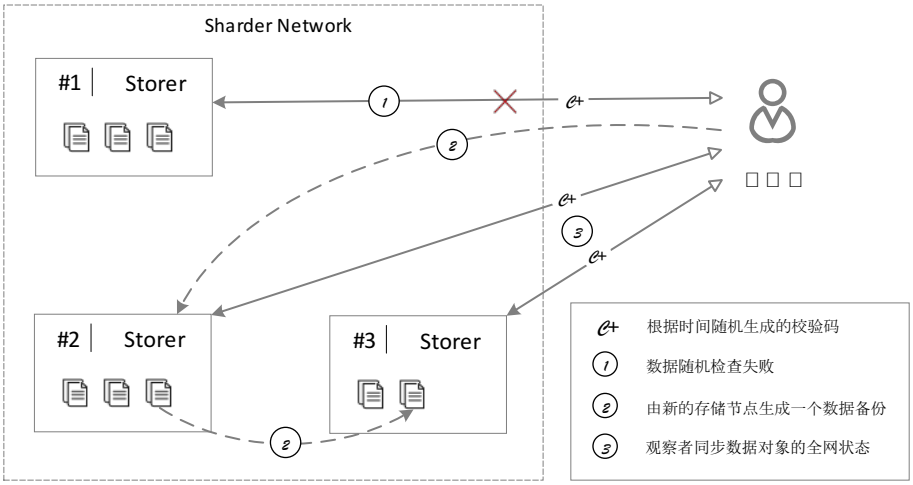
1、 客户端（Client）发起取回数据请求，豆匣协议从对象账本（Bean-Book）中获取最新的数据对象返回给客户端，并向存储节点同步此数据取回请求。

2、 主动模式下，客户端和存储节点建立连接，并从存储节点获取数据。被动模式下，存储节点会将数据推送给客户端。

3、 存储节点（Storer）在客户端取回数据后，会更新存储账本（Store-Book）中。

4、 存储节点接受完数据后更新存储账本（Store-Book）和数据对象（Bean-Book）的全局状态。

5、 客户端（Client）在取回数据后会更新证明账本（Proof-Book）以证明存储节点确实保存了数据对象。



图片 3 观察者检查和调整数据图

备份调整 观察者在存储节点不可用的情况下会要求其他存储节点保存一份数据备份。有时候旧节点只是临时断开网络或意外宕机，当旧节点重新连入网络时会出现到底哪个节点应该继续存储数据备份并获得报酬的问题？对于这种情况，豆匣协议会根据该

节点的豆匣信用等级来进行判断。原则上是信用等级高的存储节点会继续存储数据并获得存储报酬，信用等级低的则需要删除数据。

资源访问 数据和数据分片在全网随机存储和备份，想要取回数据第一步是需要知道从哪些存储节点取回数据，也就是需要数据索引服务。观察者会是一个理想的数据索引服务提供者，观察者为了让全网的数据状态快速收敛一直都在不断“观察”全网数据对象的状态并不断“调整”全网的数据分布。所以观察者可以向用户提供最实时的数据对象资源访问地址。

4.4.3 数据检查

1、 客户端（Client）或观察者（Watcher）根据时间随机生成校验码 C，并把随机验证交易记录到证明账本（Proof-Book）。
2、 豆匣协议要求对应的存储节点（Storer）根据校验码 C 生成相应的存储证明 M。
3、 存储节点（Storer）在有限时间内将存储证明 M 提供给客户端（Client）或观察者（Watcher）进行验证。
4、 客户端（Client）或观察者（Watcher）验证通过后更新证明账本（Proof-Book）。
5、 验证通过后豆匣协议会生成奖励交易（Reward-Book）并将部分存储报酬解锁给存储节点（Storer）。

引入 merkle 树[6]和 zh-SNARK[7]让存储节点进行存储证明。存储证明的随机检查由数据所有人或观察者发起。详见 PoR【3.8.1 备份证明】和【3.8.2 存储时长证明】。观察者需要根据安全策略，有规律地对全网的数据对象进行检查【3.6 数据可用性】。维护数据的全网状态一致性，同时还有义务修复存在的或潜在的安全性和可用性问题（如：① 数据分片丢失或不可用，② 存储者长期不可用并已超过预设的阈值）。

4.4.4 状态收敛

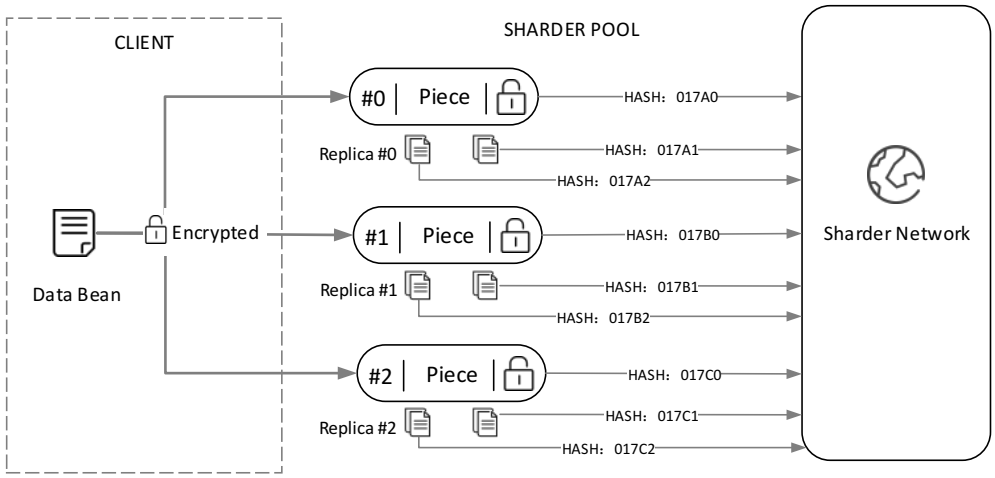
对一份数据的备份耗时，以及观察者对数据进行检查并调整数据分布的时间耗时可以认为是数据对象的收敛性证明问题，以下我们进行概要性证明。

假设网络中有个 N 个节点，存储一份数据的时间为 S_t，则极端情况下在进行了 N-1 次询问后，网络中的最后一个节点才应答被认为是可用的。则备份一份数据文件的时间复杂度是 O(N)+S_t，S_t 在网络稳定的情况下是个常数，所以时间复杂度可以简单认为是 O(N)即在网络中寻找可用节点的时间总开销。这对一个经常有加入和退出节点的网络 O(N)是不可忍受的。不过 Kad 网络中的 k 桶（k-bucket）的引入可以帮助缩短查询可用节点的开销。假设 t 是目标查询节点，由于每次查询都能从更接近 t 的 k 桶中获取信息，这样的机制保证了每一次递归操作都能够至少获得距离减半的效果，从而保证整个查询过程是可快速收敛的且收敛速度为 O(logN)。

4.5 数据安全性

数据加密 数据文件在客户端中会默认进行加密（AES-256-CTR）后再存储到存储节点。意味着数据存储者实际上无法查看该文件的内容。对于敏感数据，数据所有者可以自行选择使用硬件加密的方式生成加密文件数据后再存储到豆匣网络中。

数据豆分片



图片 4 数据豆分片

数据豆分片（简称“数据分片”）的策略和安全策略紧密相连，如果数据所有人对数据的安全性要求很高，分片能很大程度保证数据的安全性。为了保证数据分片的可用性我们引入了数据纠删。

我们认为存储节点不会为过小的文件作弊，存储节点删除数据文件只保留相应的 R 证明并不能带来显著的经济收益。大多数情况下普通存储节点的性能瓶颈是带宽和磁

盘 I/O，意味着普通存储节点的磁盘空间并未存满数据分片文件。但是过多的小文件确实会拖慢数据的读写，这个问题可以依赖豆匣文件系统【3.17.1 豆匣文件系统】提供高性能的并行数据处理来进一步解决。

多备份 假设豆匣网络中有 b 个存储节点，数据被分片成 p 份，每个数据分片的备份数为 n。则能成功取回数据的几率 R_s 公式如下：

$$R_S(\mathrm{b},\mathrm{p},\mathrm{n}) = \frac{\binom{b-p}{n-p}}{\binom{b}{n}}$$

b:网络中的存储者数量　p:数据的分片数量　n:数据的备份数量

--

代码片段

```
double fac(int p){
    return p == 0 ? 1  : approximation(p * fac(p-1));
}

double choose(int h,int k){
    return fac(h) / fac(k) / fac(h-k);
}

double rs(int b,int p,int r){
    return choose(b-p,r-p) / choose(b,r);
}

double retrieve(int boxerCount, int pieceCount, int replicaCount) {
    return rs(boxerCount,pieceCount,replicaCount);
}
...
```

取回概率

Boxer	Piece	Replica	Retrieve
100	10	10	5.776904234533874E-14
100	10	50	5.934196725858287E-4
200	10	50	3.7276043023296E16
200	50	90	5.7872010853195E44
300	80	90	4.094234910939596E131
500	50	200	3.146459521303754E45
...			

安全策略 最基本的安全策略就是按照通常数据灾备的方式一份数据至少存在三份拷贝：同节点或则临近节点一份，不同地域的节点存储一份，跨国家的节点存储一份。不过更高的安全策略意味着使用更多的存储空间，更复杂的数据“观察”和“调整”。豆匣协议中允许数据所有者根据自身需求定义数据安全策略，目前允许设置的参数有：数据备份数、数据分片数。安全策略会直接影响到观察者如何修复丢失数据，也会影响数据对象在全网的状态收敛速度。

4.6 数据可用性

数据纠删 纠删码（Erasure Coding，EC）[2]是一种数据保护方法，它将数据分割成片段，把冗余数据块扩展、编码，并将其存储在不同的位置，比如磁盘、存储节点或者其它地理位置。为了确保数据的可用性而又不过度占用存储空间（可以增加存储节点的空间利用率），豆匣网络对数据豆分片会进行数据纠删处理。

Reed-Solomon（简称 RS）码[3]是较为常用的一种纠删码，它有两个参数 n 和 m，记为 RS(n,m)。n 代表原始数据块个数，m 代表校验块个数。以下是全备份和 RS 纠删码的性能比较，具体算法实现请参见[4]和[5]，此文不再赘述。

类型	磁盘利用率	计算消耗	网络消耗	恢复效率
全备份（3 备份）	1/3	非常低	较低	较高
RS 纠删码	n/(n+m)	高	较高	较低

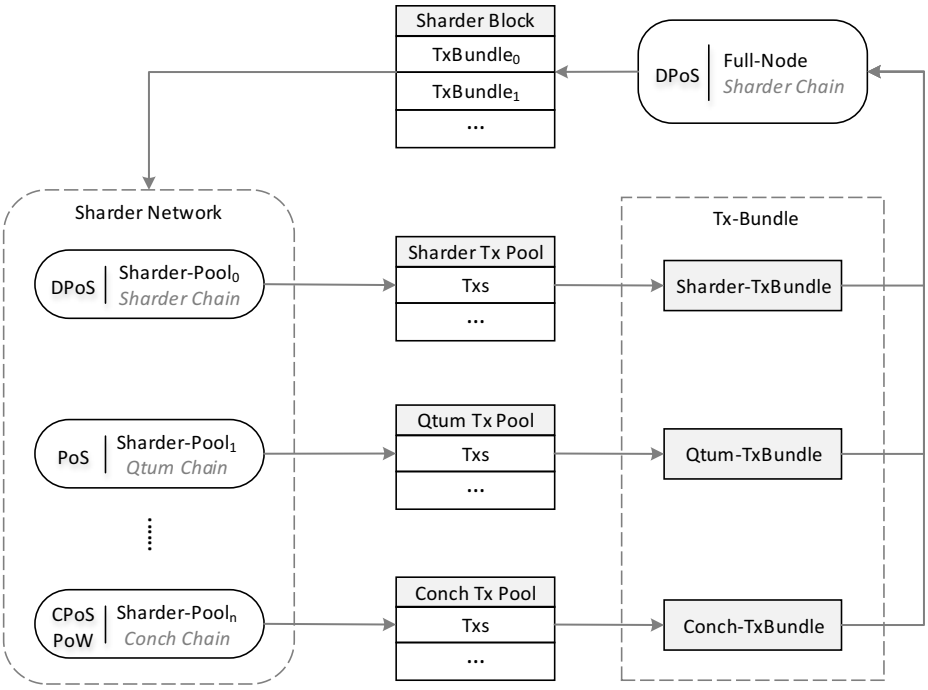
分布调整 观察者会不断调整数据的备份和分布，以确保当前的数据文件是安全的并至少有一个可以访问的资源。

4.7 共识和出块

我们认为比特币网络中的 PoW 共识出块，虽然向人们展示了一个简单明了的经济激励框架和共识机制，能保证一个无主分布式网络很好地工作。但是随着矿工开始使用昂贵的硬件设备和更进一步地硬件“军备竞赛”，消耗掉大量电力和计算资源仅仅是为了争夺出块权，我们认为这不仅是一种资源浪费还过度消耗了硬件资源增加了大量的电子垃圾。我们希望能在保证区块链网络安全的同时提供低消耗的共识出块算法。

共识出块 多链共识出块的方式如下图所示，由交易包（Tx-Bundle）、豆匣区块（Sharder Block）组成。这种方式允许每个豆匣池在内部执行不同的共识算法，一个交易包中包含了所属豆匣池中的交易记录。由全节点生成包含了不同交易包的豆匣区块并公布到网络中，每个交易包（Tx-Bundle）需要包含豆匣池和节点的身份信息：Node-ID，Pool-ID，Area-ID。

一个全节点只能连入一个豆匣池，连入了豆匣链（即 0 号豆匣池）的节点可以打包生成豆匣区块。未来我们会探索让豆匣池单独打包出块，可行的实现思路是在每个豆匣池中部署至少一个连接了豆匣链的代理节点（Sharder Agent）。



图片 5 多链共识出块

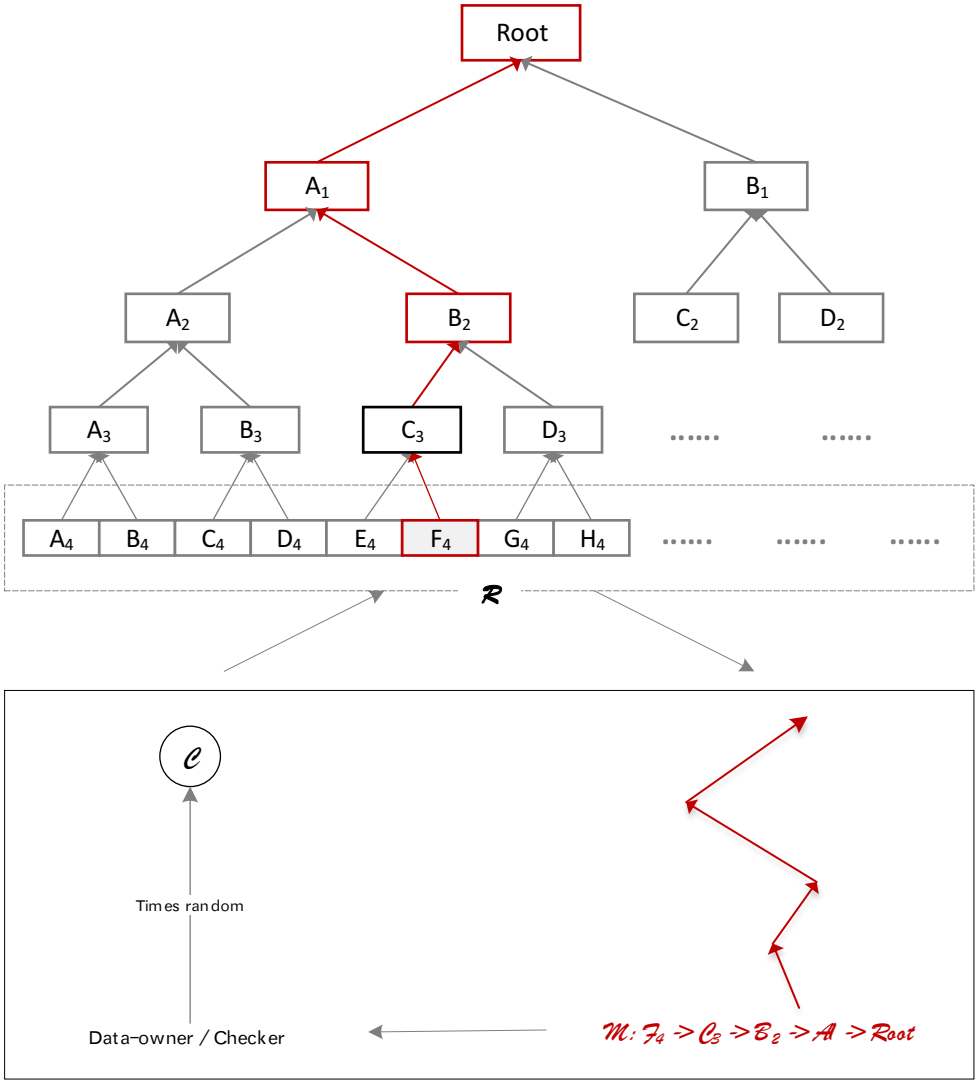
信息上链： 不是所有的数据和信息都需要上链，尤其是在豆匣网络中大部分的数据和操作结果都不会上链。比如：数据文件就不会保存在链上，链上保存的对象 URI 是一个指向了当前可用资源地址的指针。

除开基本的区块信息外，上链的有：账务交易、对象数据、存储交易、证明交易。值得注意的是一个存储交易会对应一个对象数据但是可能会导致一个或多个奖励交易（基于 PoST 的存储报酬发放算法）。

4.8 贡献度量化

引入 merkle 树[6]和 zh-SNARK[7]构成 PoR(备份证明 Proof-of-Replica)和 PoST（存储时长证明 Proof-of-Storage&Time）作为存储者（Storer）存储数据的量化凭证。信用等级高的存储节点允许采用 PoR 用较短时间就可以提供证明，信用评级较低的会要求使用 PoST 提供存储时长证明。

4.8.1 备份证明 Proof-of-Replica



图片 6 备份证明 PoR 流程图

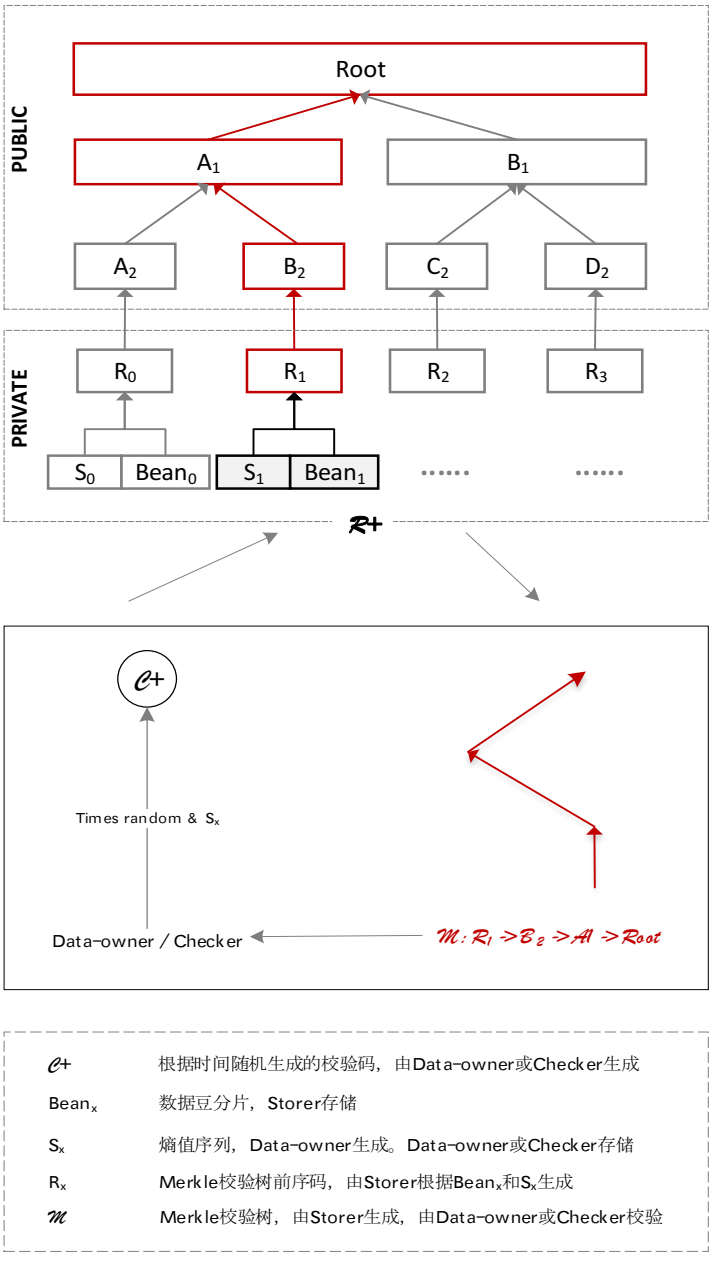
数据所有人可间隔一段时间就向豆匣网络请求相应的备份证明：

- 数据所有人基于时间生成一个校验数 C 发送到豆匣网络。

- 存储者需要根据 C 找到对应的数据分片并生成$\underset{M}{\rightarrow}$(merkle 校验树)。
- 如果校验通过豆匣网络会更新存储账本(Store-Book)和奖励账本(Reward-Book)，并解锁存储账本（Store-Book）中该笔交易的部分奖励作为存储报酬。

4.8.2 存储时长证明 Proof-of-ST (storage and time)

PoR 虽然能保证数据存储者至少会保存数据一次，但是无法避免作恶者进行欺骗，考虑以下场景：



图片 7 存储时长证明流程图

1、 存储者在第一次按要求对数据进行备份后，对所有的数据分片和可能的拆分序列计算其 merkle 检验数，并删除数据分片文件仅保存 merkle 校验树。

2、 存储者收到证明指令后，请求其他保存了数据备份的节点获取数据，并计算出 C 相应的$\underset{M}{\rightarrow}$(merkle 校验树)。

以上场景中，作恶者使用极低的计算和存储成本就可以获得存储报酬。因此引入 PoST，以确保只要数据存储者没有存储数据分片文件就无法正确计算出 merkle 校验树，也就无法获得存储报酬：

- 数据分片后生成一个熵值序列 S，然后使用 S 和数据分片再生成哈希值 R。

- 每隔一段时间数据所有者向豆匣网络发送 S_x（基于时间的熵值，全局唯一），存储者需要根据 S_x 和对应的数据分片计算出 R_x，并根据 R_x 生成相应的 merkle 校验树。

有了前置熵值序列，还可以实现由观察者代理进行数据检查。数据所有者可以提供一部分熵值序列交由观察者，由观察者来完成 PoST 的证明校验。为了更加安全地执行，未来会依靠智能合约来实现代理检查逻辑。

4.8.3 信用证明 Proof-of-Credit

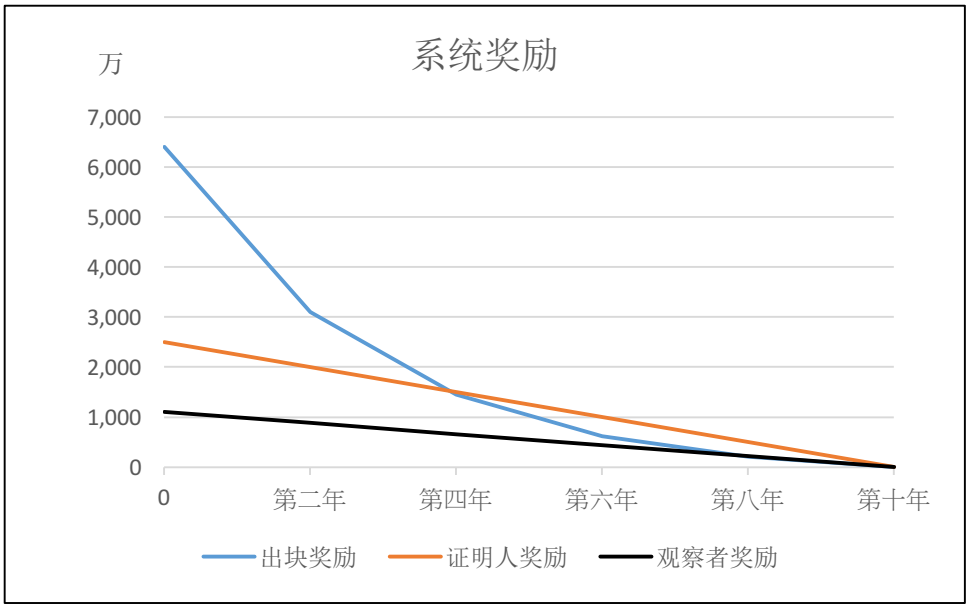
在豆匣协议中信用证明是和账户绑定的，在海螺链 CPOS 评分体系[8]基础上根据不同角色相应的计算公式侧重点会有所不同：

- 存储节点：存储总量、存储时长、在线时长、被惩罚量。
- 全节点：最大交易处理量、出块速度、分叉收敛速度、在线时长。
- 观察节点：索引服务性能、在线时长。
- 数据所有者：存储数据量、交易量。
- 证明人：证明量。

4.9 奖惩机制

4.9.1 系统奖励

豆匣网络为了鼓励更多的节点加入，形成更加安全健壮的网络。对豆匣网络做出贡献的各节点会发放系统奖励。



图片 8 奖励分配图

出块奖励 出块的时间设置为 5 分钟（最快能 10 秒产出一个区块）。每天能产出 288 块区块，一年能产出 105,120 块区块。出块人每产出一个豆匣区块获得 218 个 SS 奖励；每产出 210,240 个区块奖励减半；于 1,208,160 高度后出块奖励会固定为 14 个 SS。预估 10 年完成 64,000,000 个 SS 的奖励发放。出块奖励发放完后，存储交易手续费和 B 端用户支付的技术服务费将成为维护豆匣网络的经济激励。

观察奖励 在豆匣网络运行之初，官方节点将作为观察者。待全网状态稳定后，由全节点竞选观察者，系统预留了 25,000,000 个 SS 作为观察者奖励，奖励数量将根据观察者的信用证明算法进行计算，观察者 SS 奖励会被智能合约自动执行和分配。

证明奖励 豆匣网络预留了 11,000,000 个 SS 作为证明人奖励，激励和邀请具有公信力的组织或机构担任证明人，为链上数据提供证明服务。未来豆匣市场也允许数据所有者和证明人之间自由设定价格进行公证交易。

4.9.2 系统惩罚

豆匣会对以下危害网络的情况进行系统惩罚，包括罚没其豆匣币并降低其信用评级。

丢失数据 不再支付后续存储数据的报酬，降低其信用评级，低到阈值后该用户会被加入黑名单并无法再连入豆匣网络。

恶意攻击 不管是蓄意攻击而不出块，还是大量通过非正常手段获取 SS，对豆匣屋里网络和程序进行攻击都将会触发最高级别的惩罚：该节点和关联用户被加入黑名单并无法再连入豆匣网络，还会罚没该账户上的所有豆匣币。

欺诈 欺诈通畅会被发现于事后，目前无有效办法追回已支付的报酬和系统奖励。只能降低其信用评级，降低到达阈值后该用户会被加入黑名单并无法再连入豆匣网络。对于欺诈最好的防范办法是增大其欺诈成本，也可能会采用保证金模式避免节点恶意欺诈。

4.9.3 交易报酬

通过豆匣网络中的自由交易市场提供服务而换取相应的报酬。

存储报酬 提供存储空间可获得存储报酬，存储空间的单价由自由市场决定。

服务报酬 在足够活跃的自由市场中，节点可以选择提供个性化服务（比如：独立的数据索引服务、定制化轻钱包客户端）而获得相应的服务报酬。

4.10 豆匣币 （SS-Sharder）

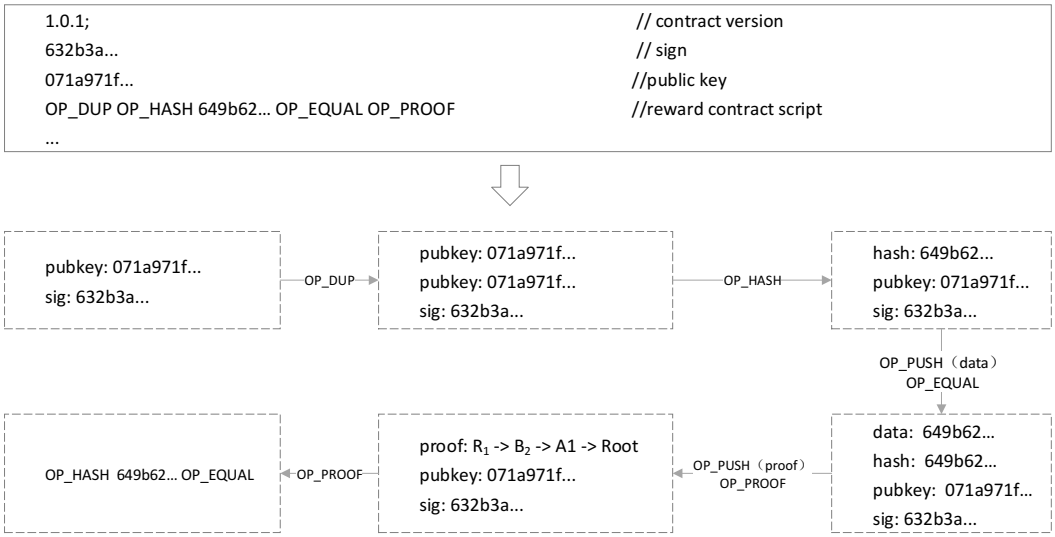
豆匣币是豆匣协议中的内置加密数字代币。主要用于激励豆匣协议构建的存储生态，奖励对豆匣网络做出越多贡献的角色，同时使用豆匣币作为经济惩罚的手段，避免恶意节点作恶和智能合约中可能出现的无限循环的逻辑炸弹等。同时豆匣币 SS 还作为多链（多豆匣池）结构下的锚定代币。

4.11 智能合约

智能合约经过以太坊的实践，已经证明其是可信和高效的。豆匣协议中的智能合约分为两个阶段实现，第一阶段会实现非图灵完备的智能合约，主要用于保障使用内置代币 SS 进行奖罚、为上层的交易模型提供支撑。第二阶段会实现合约虚拟机、图灵完备、预言机（oracle）、哈希锁等高级特性。

第一阶段的智能合约会使用经典的 FILO（先进后出）栈结构。以下是基于 FILO 基本的原子操作定义：OP_INIT（构造空栈）、OP_EMPTY（判栈空）、OP_FULL（判栈满）、

OP_PUSH（压入栈）、OP_POP（弹出栈）、OP_DUP（复制栈）、OP_COUNT(计数器)、OP_HASH（算 HASH），OP_PROOF（贡献度证明），OP_CHECKSIGN（验证签名），OP_EQUAL(判相等)。随着场景变多，会持续增加原子化的操作符支撑更复杂的操作。一个发送奖励合约的代码片段如下：



图片 9 非图灵完备智能合约

执行过程如上所示，最终正确提供了 proof 证明的存储节点将会得到相应的奖励。不过奖励需要提供符合这笔交易的公钥（该公钥需要通过 OP_HASH 运算符算出结果为 649b62…），第一阶段的智能合约基于 UTXO 模型（简单说即是基于地址的），需要由上层的豆匣账户进行交易和地址的归并。

智能合约是构成豆匣自由市场的重要组成部分，不仅 Store-Book 这类全网的交易订单对象会由智能合约来保障其订单状态变更和费用支付。以后的各种全网数据 Book 对象都会基于智能合约进行操作和约束。智能合约也会基于账户模型并实现图灵完备。智能合约的第二阶段我们会具体设计和实现。

4.12 客户端

会提供 GUI（图形交互界面）和 CLI（命令行）两种方式的客户端：

存储节点端 存储节点版本只向豆匣网络提供本地的存储空间。存储节点版本不需要保全所有的节点信息，但需要保存数据文件和相应的贡献度校验用的相关文件。运行该客

户端会作为存储节点向豆匣网络提供你的本地磁盘空间，未来还会尽可能多地集成国内外流行的个人网盘和云存储。

全节点端 运行全节点客户端意味着你不仅向豆匣网络提供存储空间，同时你还需要执行打包交易出块和观察者的所有功能。

证明人端 证明人可以查看并对授信部分的链上数据提供证明。对于有独立信息系统的，会采用 API 方式进行集成。证明流程由智能合约完成。

观察者 可以在全节点客户端中打开观察者角色的功能，并竞选上岗。观察者需要监控豆匣网络中数据对象的状态变化，并向网络中发送指令修复存在的安全性和可用性问题。

未来还会提供完善的 SDK 包和 API。所有客户端都会包含完整的钱包功能，当然也会提供手机版轻客户端，轻客户端需要连接到全节点才能正常工作。

4.13 多链生态

部署了豆匣协议的豆匣池构成了多链生态。

交易交换 每个豆匣池内部的交易包（Tx-Bundle）包含了当前豆匣池中定义的交易类型。最终交易包会被打包成豆匣区块（Sharder Block）并广播到豆匣网络中。

价值交换 每个豆匣池可以定义内部流通的代币，依靠豆匣币可以在豆匣池间完成不同代币的交易形成价值交换。我们首先会让豆匣链和海螺链构建多链生态，未来还会引入跨链的信息和价值的互换。基于同态通道的闪电网络和智能合约应该是跨链的好选择。

4.14 自由市场

自由市场是一个点对点的自由交易市场。由各种全网账本（Book）、智能合约、交易对手方组成。目前的共识架构和全局账本设计暂时不能支持超高频交易，涉及全网账本状态更新和同步需要一定时间和节点共识才能完成。我们会在未来调整架构将信息流和资金流拆分，交易撮合移到链下，链上只做资金清结算，以适配未来高频或超高频的交易场景。

全网账本 目前有存储交易账本（Store-Book）、奖励交易账本（Reward-Book）、证明交易账本（Proof-Book）。

交易方 交易双方可以使用智能合约挂单购买。如果账户完成了挂单，交易信息被公布到了网络。即使节点离线，交易也会成功。

智能合约 由于没有中心化的坐市商，因此需要依靠豆匣网络自身来完成交易的匹配和撮合。目前撮合时不考虑购买方的挂单先后，仅根据价格和存储空间的需求量进行撮合匹配。

自由定价 理想状态下交易双方可以自行定价，由自由市场完成撮合，由智能合约保障交易和交割。不过早期豆匣链会作为价格的采集方和公示方提供参考市价和撮合服务，便于买卖双方进行定价和完成交易。

手续费 手续费是执行智能合约所需的豆匣币 SS。如果由交易/价格撮合商来完成，手续费由撮合商定价并收取。所有的手续费使用豆匣币 SS 来结算。未来可能引入免除交易费的模式：帮助他人进行撮合或则确认交易即可免除交易手续费。

豆匣链 在自由市场中，豆匣链始终会是存储空间的售卖方，以保证有充足和稳定的存储空间以提供给用户使用。

4.15 授信框架

当前大量的商用场景仍需要一定程度的监管和审计，这种监管和审计应该是在双方知情的情况下进行的。豆匣协议提供了一个基础框架帮助完成账户授信和审计。便于个人、企业、监管机构能够在日常的使用中对用户的授信数据进行访问和使用。

账户授信 信息的分级和审计需要在数据所有者的知晓和授权下完成。借鉴 BIP39[9]多级分层确定性钱包的思路让数据所有者进行授信。我们会参考 BIP44[10]的路径定义方式和以太坊 EIP85[11]中的讨论，引入账户的概念构成如下路径：

m/purpose'/coin_type'/ account' /change/address_index

审计 由于交易双方的匿名性，即使区块和交易信息公开的也无法很好的对信息进行审计。比如交易信息，审计方需要获得交易双方的授权下，才能解锁账户信息和交易信息、

账务信息等进行比对。审计方需要重复授信过程获得双方的账户授信后才能完成交易的审计，所有的审计结果也会记录上链。当然为了商用还可进一步设计批量账户授信等模型，尤其是在 C 端用户信任 B 端地场景下可以避免审计者重复申请授信的冗长过程。

KYC 豆匣协议中不对用户身份进行识别，相应的身份识别可由上层使用者自行决定如何实现。

信息分级 信息分级类似于数据读写权限控制，将不同数据对象分级使用不同的衍生密钥（不同的 HD 路径）进行加密，这样获得衍生密钥仅能解读这一部数据。这会导致比较复杂的私钥生成和数据加密逻辑，未来会进一步讨论实现方案。

4.16 恶意攻击

51%攻击 这是所有区块链系统都会面临的问题，想完全避免这一问题是不可能的。为了降低被攻击的几率，豆匣协议采用 PoS 和 DPoS 来出块，后期还会结合 PoC 来动态选择合适的出块者。

Sybil 攻击 豆匣网络要求一个交易至少向周围临近的 3 个节点进行校验，除非攻击者能计算出被攻击节点附近的网络拓扑并伪装成与之临近的节点，否则这个方案可以极大几率降低 sybil 攻击。随着越多节点加入，sybil 攻击的概率也会大大降低。在早期官方节点的地址列表会公开于官方和内置在发行的客户端中。只要 3 个检验节点中包含一个官方节点就可以有效避免 sybil 攻击。

数据欺诈 如果存储节点在收到随机校验请求后，在极短时间内从附近的数据存储节点获取数据文件并计算出正确的 merkle 验证路径，该节点就能通过校验并获得存储报酬。采用 PoST 可以极大降低这种概率，对于进行数据欺诈的节点还会受到相应惩罚，比如：降低其信用评级，或则永远不被允许接入豆匣网络。

数据劫持 拒绝提供最后一个数据分片使得无法还原分片前地数据豆对象，以此来向数据所有者勒索高昂的费用。在豆匣协议中数据进行了分片和多备份，存储节点不一定知晓哪一块数据分片才是最后一块。即使存储节点知道，也可从其他节点取回数据分片地

备份。除非所有拥有该分片的存储节点都被恶意攻击者控制，此种情况地概率较低，随着豆匣网络更加离散，数据劫持地概率会进一步降低。

数据抹除 当存储节点认为当前的数据分片存储的奖励额度过低或则纯粹因不再想存储该数据分片，选择从磁盘删除该数据分片。豆匣协议的多备份可以降低此种情况给数据所有者带来的损失，还可以调整 PoST 的奖励策略，将大量地存储报酬在存储时限到达时进行发放。最有效的是对该存储节点进行惩罚，降低其信用评级，减少该节点未来的可能收益。

4.17 远景

4.17.1 数据可用性

根据 CAP 理论，我们必须在一致性 (Consistency)、可用性 (Availability)、分区容忍性 (Partition Tolerance)中做出取舍。我们假定一种策略：N = 副本数，W = 一次成功的写操作必须完成的写副本数，R = 一次成功的读操作需要读的副本数。策略即是我们对NWR的数值进行设置,得到一种CAP的取舍。比如Amazon选择的是N3W2R2,意味着当两个数据副本失效时，受影响的这部分数据就变成只读，无法再写。未来会继续研究和参考当前领先的云存储服务商（Amazon、Facebook、Aliyun）进行优化，以确保在更好的性能的基础上拥有更好的数据可用性。

为了降低数据纠删时计算资源和网络 I/O 的开销，在实现了经典的 RS 纠删码后，会根据实际需要考虑是否实现 SIMD 技术加速和 LRC（Locally Repairable Codes）纠删算法，如 FaceBook 和加州大学提出的 XORing Elephants[12]。

4.17.2 数字资产管理

在很多房地场销售大厅已经有智能设备帮助你自动开设银行的电子账户并锁定一定的金额。这部分锁定的金额是购房意向金也是你资产的证明。购房者不用像以往在正式签订购房合同前就需要向地产商缴纳一定的意向金，而地产商也锁定了购房者的购房意向。但是非银行体系的数字资产仍然没有好的办法便捷地进行资产证明，甚至你很

难证明交易账户里的的持仓代币是属于你的。豆匣协议通过证明人角色提供 POA，加上区块链溯源和不可篡改的特性构成了完整的证据链条和可信数据，未来会提供完整地数字资产管理和证明地方案。

可信地数字资产加上智能合约，可以在弱信任或无信任的情况下完成自动交易。还能衍生出很多资产管理和证明地方式，如 :在指定时间和指定条件下从一个地址转移特定数量的资产到某一个公开的地址（这和零知识证明有异曲同工之妙，能证明你是该数字资产地址的持有者），达成某种条件后（达到某个时间、或则某个预测）自动执行智能合约里提前约定的操作。

4.17.1 豆匣文件系统

豆匣文件系统 SFS（Sharder File System）将会在 CloudAqua 的基础上进行升级，不仅可以增加单节点的读写吞吐性，还可以提高数据库吞吐性能，允许并发多进程读写。能提高大量碎片文件的 IO 性能。SFS 兼容 Ext4、HFS+、NTFS 常见的日志文件系统，有了 SFS 后也更加容易部署于不同操作系统和物理环境的节点上。

4.17.2 人工智能

近几年随着硬件的发展，人工智能在监督学习、对抗网络等领域中都有了长足发展。区块链系统本身就是一块块有着多种多样开放数据的天然土壤，而豆匣协议构建地也正是存储数据的分布式网络。如何给这些上链数据贴上标签、分类、训练 AI 是非常有挑战和吸引人的事情。AI 的不断学习也有助于帮助豆匣网络更加智能、安全、高效。短期内能预见的是 AI 能帮助改善安全策略，让观察者更好地“观察”和“调整”豆匣网络，提前预警和及时干预可能出现的恶意攻击。帮助豆匣网络成为一个自治和智能的区块链存储网络。

5、 豆匣链 (Sharder Chain)

第一个部署豆匣存储协议的存储公链也是 0 号豆匣池 (Sharder-Pool$_0$)，是构成豆匣网络的重要基石。豆匣协议的所有特性都会在豆匣链中首先实现和测试。同时为上层地商业应用还提供如下特性：易用的账户模型、数字资产、担保交易、定制化 API、运营支撑系统等。

5.1 节点和网络

豆匣链会使用改良版的 Kad 协议组成点对点对等网络，为了在早期快速形成稳定的网络和足够数量的全节点，豆匣还会发布低功耗微节点矿机 Sharder Hub 以及存储和挖矿并举的存储挖矿一体机 Sharder Box。

Sharder Hub 能更方便地将闲置的磁盘资源接入豆匣网络，还能保证更稳定的在线时长。Sharder Hub 内置了豆匣客户端，零配置即可接入豆匣网络开始挖矿和共享存储空间。Sharder Box 不仅可以共享存储空间获得存储报酬，还可以通过出块和竞选观察者提供服务获得多重奖励。

5.2 功能模块

图片 10 豆匣链功能结构图

区块链层 有必要地区块链模块组成。包括：点对点对等网络，UTXO 模型，分布式账本和全局 Book 模型，原生的豆匣币。

数据层 实现了豆匣协议里定义的数据操作，数据对象的分片和备份，观察者角色，证明人角色等。

资产层 友好的豆匣账户模型，将代币、数据豆对象和豆匣账户关联形成数字资产模型，提供数字资产的管理。

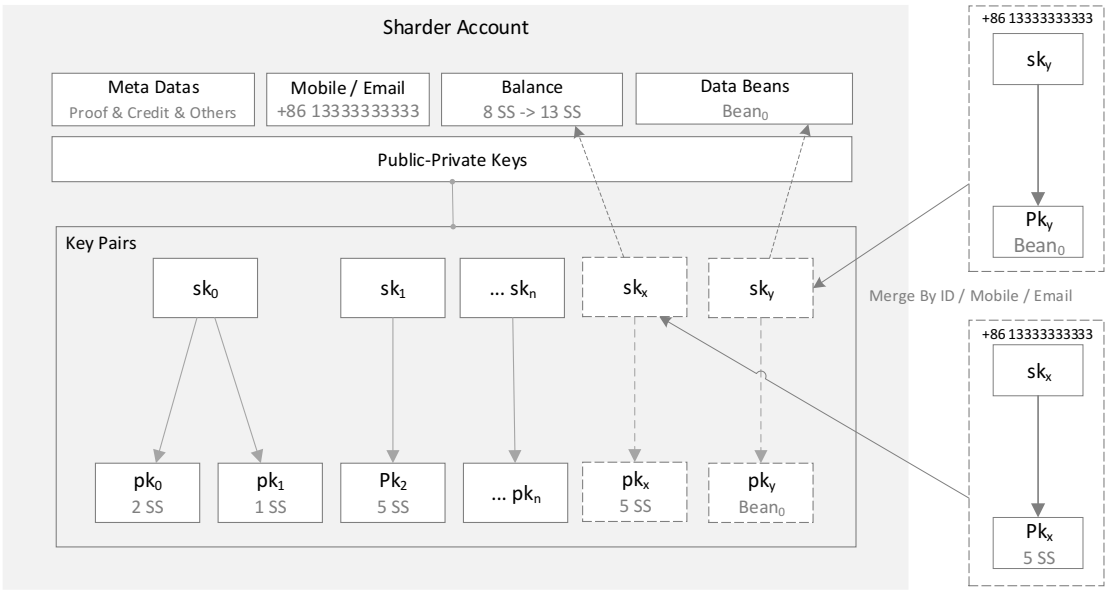
组件层 抽象和封装了一些基本组件，并基于智能合约提供各种交易模型。

接口层 对外地 **API，** 便于合作伙伴和商户使用区块链和存储服务。

贡献量化 **量化**各种角色对豆匣网络的贡献。不同角色拥有不同地量化指标函数，贡献会获得奖励，作恶会受到惩罚。贡献度会被包含在豆匣账户中。

运营支撑 便于商户接入豆匣网络，对于整个商户的生命周期进行服务。还提供数据统计和分析帮助商户和豆匣改善运营质量。

5.3 豆匣账户



图片 11 豆匣账户图

豆匣账户不再是一个个零散的地址（由对应的私钥对管理），手机号和邮箱作为账户的标识，记账模型沿用比特币的 UTXO 模型，UTXO 拥有极强的可追溯性和审计性。未来会在 UTXO 模型之上构建账户模型。

豆匣账户下管理的公私钥就如比特币的 HD 钱包。豆匣账户的安全性由种子钥匙保证，种子钥匙的私钥由用户持有（为了方便用户记忆，会实现 BIP44 提供一组助记符）。账户模型会自动根据账户的标识归并余额和数字资产（不再出现比特币 UTXO 模型导致的找零地址），统一管理在豆匣账户下。用户只需安全保存种子私钥就可安全使用豆匣账户。豆匣中使用椭圆曲线秘钥算法[13]生成秘钥，签名使用 EC-KCDSA[14]来实现。

账户模型需要由性能较好、存储量较大、在线时长稳定的全节点进行全链扫描才能归并出账户信息。观察者也可以提供更高效地数据索引和缓存服务。

5.4 数字资产

能够预见到，随着人们生活的数字化，各种实体资产也将逐渐数字化。豆匣网络旨在帮助企业和个人存储各种数据，并进一步进行数字资产管理。

当然豆匣账户下的数字资产只能在豆匣网络内才能管理和交易，和外部世界的数字资产流通仍然需要外部的证券交易所、新兴的数字资产交易市场、中间商来完成。豆匣链会和这些交易所进行对接，也会和一些去中心化的交易所及跨链交易撮合协议合作来完成数字资产管理和交易。

5.5 担保交易

为了让豆匣链上层地 DApp 和商户能更方便地使用智能合约进行交易。我们在智能合约之上封装了担保交易模型，值得注意的是这里提到的交易（Trade）并不等同于区块链系统的交易（Transaction）。豆匣链中的担保交易使用智能合约来替代类似于淘宝或 Paypal 这类中间背书机构。

担保交易创建时会生成一份智能合约，智能合约会锁定卖方交易地址上的资产（SS 余额、数据资产、其他资产），当买家支付了约定的 SS 后，智能合约会将资产转移到

买家的地址。目前只能自动交割豆匣网络内的数字资产。未来随着更多证明人作为背书机构接入，担保交易能够交易的数字资产会逐渐增加。

预授权担保交易：买卖双方都需要支付一定数量的 SS 作为保证金。竞拍模式下，所有参与交易的保证金地址和数字资产地址都会被智能合约锁定，状态被同步到全网。智能合约持有卖家和买家的时效性私钥 tpk，类似于信用卡的预授权模式。

图片 12 时效性密钥图

交易成功时，智能合约会自动使用时效性私钥完成代币转账和数字资产的交割。反之则取消交易解锁地址。如果存在作恶和恶意竞拍，保证金会被罚没，账户地信用值 PoC 会降低。

6、 豆匣社区

比特币从中本聪的论文提出一直运营到现在，比特币社区功不可没，甚至可以说没有社区也就没有今天的比特币。所以我们也希望借助社区的力量、大众的智慧持续改进。

豆匣理事会 组建豆匣理事会，由豆匣基金会成员、币圈资深人士、社区成员三方组成。由豆匣理事会牵头维护豆匣社区的正常运营，组织社区成员开展规范讨论、发展规划建议、改进意见等。并合理使用社区基金奖励社区繁荣发展。

线上渠道 官网、电报群、豆匣社区、官方 QQ 群、官方微信群等线上渠道构建社区成员自主发表想法和意见的社区。

奖励规则 理事会对社区用户的贡献奖励进行记录，审查，公告。每个用户在社区贡献方面的奖励计算公式暂定为：S=N%×M+50×N。S 为豆匣币奖励数量，N 是整数因素（0<N≦5），M 是该地址原持币数量（M≥100）。奖励机制会由豆匣理事会持续完善和修改，每月基金会都会公告通报基金使用情况及工作汇总。以发布的《豆匣社区白皮书》为准。

社区远景 豆匣社区的发展既需要区块链技术爱好者参与代码编写、审查、测试，也需要更多的公链、企业、个人参与豆匣链的测试，同时还需要社区用户的积极参与推广宣传、规划讨论、提出改进意见扩大豆匣协议的普及度。最终形成利益共享的繁荣的豆匣社区。详细的社区运营细则和奖惩规范请参见《豆匣社区白皮书》。

7、 应用领域

纵观中国第一梯队的互联网科技公司，是大量长尾用户、大批量交易、高频地使用推动了它们快速发展和技术革新。我们坚信有更多的商户和用户使用，更多网络部署豆匣协议，才能长远发展。

我们认为在公益领域、物联网领域、供应链领域、共享经济等领域，区块链的去中心化、不可篡改、可追溯、全网参与等特性能推动这些领域的长足发展。我们也正和合作伙伴一起研发以下领域内的区块链商业应用。

7.1 云存（Bean Cloud）

数据存储、存证和保全平台。服务于 P2P 网贷、小贷、消费金融、电商、ERP 系统，可以将电子合同、支付凭证、投资记录等电子数据记录上链，利用区块链可溯源不可篡改的特性对保存于豆匣链上的数据出具保全证书和司法证明。

7.2 矩阵（Sharder Matrix）

围绕个人的数据存储应用。您的生物数据(包括基因信息、成长记录、医疗数据等)，甚至于您的所思所想都可以记录到矩阵中。随着数据的不断累积形成个体独特的豆匣数据矩阵。

7.3 智脑 (Sharder Brain)

随着人工智能 AI、智能硬件和物联网的发展，以及未来无监督学习的突破。我们坚信智脑能在数据安全、数据分布调整、数据分析、数据搜索、数据预警（数据安全性预警、个人生命特征预警）等方面为个人和企业提供智能的数据服务。

7.4 数据集市 (One Fair)

基于豆匣公链和豆匣协议的自由市场最终会形成围绕个人的数据集市。在数据集市里能透明、公开、自由、公平地进行点对点交易。交易的内容包括：存储空间、数字资产、可信数据、有价信息等，甚至个人也可售卖自己的生命体征数据给医疗研究机构。数据集市 One Fair 终将让您的数据能更加便捷快速地融通，沉默数据丧失了流动性就如现金资产一般只会越来越贬值。

8、 发展规划

8.1 路线图

图片 13 项目路线图

我们认为区块链是如此迷人和拥有无限遐想的领域，在这个领域里有太多值得去探索的，其上的商业应用则更加广袤。我们希望能持续耕耘于区块链领域。

8.2 盈利模式

豆匣基金会以为非营利的方式对外开放豆匣协议（豆匣协议是开源、免费的）。但是为了持续迭代、研发、运营，会基于豆匣公链研发闭源和收费的商业应用，从以下几个方面盈利以持续发展：

盈利模式	描述
代币的增值	生态的逐渐完善会将内在价值体现到代币的票面价值上，并持续增值。
存储交易手续费	当存储买卖交易达到一定数量和活跃度后，会收取一定的豆匣币作为手续费作为维护豆匣网络的开销。
商业应用服务费 云存、矩阵、智脑、数据集市	云存数据存证平台会向商户收取年服务费，随着商户数的不断会带来可观的利润。
技术服务费	基于豆匣链构建的商业应用，会收取一定的技术服务费。并面向企业提供定制化的付费区块链技术服务，如：定制化智能合约、交易模型等。
广告流量费	豆匣链在有了一定量的 B 端商户和 C 端用户后，会投放广告获得相应的广告费和流量费。

9、 致谢

感谢成文过程中张夏、左欣荣、爱萍等提出的修订意见。感谢王凡花时间研究和测试 Reed-Solomon 算法。本文借鉴和参考了分布式 Web 系统 IPFS[15]和分布式云存储 Storj[16]的设计和 Github 上的代码，在此特意提出并感谢。

参考文献

[1] I. Baumgart, S. Mies. S/kademlia: A practicable approach towards secure key-based routing, (2007). http://www.tm.uka.de/doc/SKademlia 2007.pdf.

[2] Wiki. Erasure Code. https://en.wikipedia.org/wiki/Erasure_code

[3] James S. Plank*. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems, (1996). http://web.eecs.utk.edu/~plank/plank/papers/CS-96-332.pdf.

[4] James S. Plank. Tutorial on Erasure Coding for Storage Applications, (2013)http://web.eecs.utk.edu/~plank/plank/papers/2013-02-11-FAST-Tutorial.pdf

[5] Wiki. Reed-Solomon Error Correction. https://en.wikipedia.org/wiki/Reed-Solomon_error_correction

[6] R.C. Merkle. Protocols for public key cryptosystems, (April 1980). http://www.merkle.com/papers/Protocols.pdf

[7] Zcash Blog. Explaining SNARKs. https://z.cash/blog/snark-explain.html

[8] CPOS. Conch Chain. http://www.conchchain.org/

[9] Bitcoin. bip-0039. https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

[10] Bitcoin. bip-0044. https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki

[11] Ethereum. Eips. Standardizing HD wallet paths for Ethereum Standard Tokens. https://github.com/ethereum/EIPs/issues/85

[12] University of Southern California & Facebook. XORing Elephants: Novel Erasure Codes for Big Data. https://arxiv.org/pdf/1301.3791.pdf

[13] Yung, M., Dodis, Y., Kiayias, A., Malkin, T., & Bernstein, D. J. (2006). Curve25519: New Diffie-Hellman Speed Records. In , Public Key Cryptography - PKC 2006 (p. 207).

[14] KCDSA Task Force Team. The Korean Certicate-based Digital Signature Algorithm. http://grouper.ieee.org/groups/1363/P1363a/contributions/kcdsa1363.pdf

[15] IPFS. https://ipfs.io

[16] Storj. https://storj.io

附录

附录 A 网络操作定义

1. PING - 节点在线检测。

2. STORE - 存储键值对到 DHT。

3. FIND NODE - 从 DHT 返回自己桶中离请求键值最近的 K 个节点。

4. FIND VALUE - 从 DHT 中返回相应键的值。

附录 B 数据操作定义

1. PUT - 存储数据。

2. GET - 取回数据。

3. WATCH - 检查和调整数据。

3.1 SETUP - 为了生成校验码而进行的初始设置。

3.2 PROVE - 生成证明。

3.3 VERIFY - 校验证明。

3.4 REPAIR - 调整数据分布。

附录 C 交易操作定义

1. ADD ORDER - 生成交易订单。

2. MATCH ORDER - 匹配交易订单。

3. PROC ORDER - 处理交易订单。

4. REPAIR ORDER - 修正交易订单。

5. DROP ORDER - 作废交易订单。