# 豆匣协议（Sharder Protocol）

跨链分布式存储协议

---

V1.1 预览版


官方网站： sharder.org

官方社区： community.sharder.org

官方微信： 豆匣

联系邮箱： biz@sharder.org

Github： github.com/Sharders

# 前言

企业和个人需要存储的电子数据日益增多，虽然目前已有很多公有云存储能将数据存储在云端。但是敏感数据基于安全考虑仍然不愿意放到公有云存储中（就像没人会把自己地址的私钥放到云存储上一样，即使是严格加密后的私钥文件）。公有云存储的代码不开源，对于文件数据的安全性也没有明确地描述。同时公有云上存储的数据可能会不经用户的同意和授权就被云存储服务商私自读取和使用。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。

同时无论企业还是个人用户手上都有被淘汰、闲置、不饱和的各式存储设备（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），旨在为用户提供高性价比的存储服务。

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。豆匣协议构建的豆匣网络能帮助用户安全方便地存储数据。并在未来能帮助人们管理这些数据形成的数字资产，最终形成一个围绕存储空间、数据、数字资产的自由市场。

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），给企业和个人用户提供高性价比的存储服务。

**豆匣币（Sharder Storage 英文简称 SS）** 豆匣协议内置加密数字代币，总量为 2.5 亿。更多详情可以访问我们的官网 sharder.org。

**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）** 部署了豆匣协议的豆匣池构成的分布式存储网络。

**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** 企业数据保全和存证平台云存，原创数字版权交易平台 One Fair。

**代码开源** 豆匣协议基于现有的开源项目和开源库研发。同时我们也会在 Github 上开源。Github 地址：https://github.com/Sharders。

# 3、 设计原理

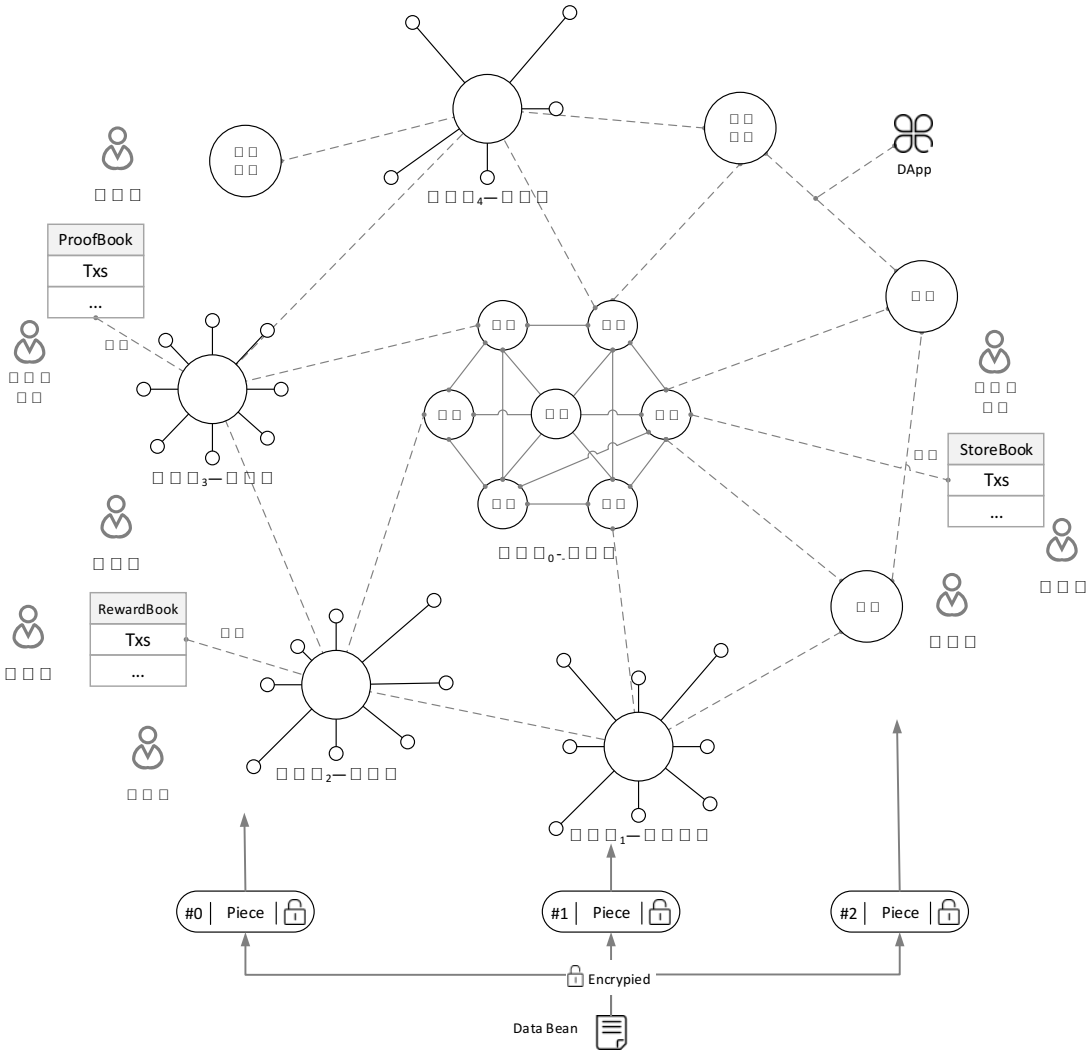
- **节点不可靠假设：**基于一个松散的网络组织结构，允许单点故障和节点一定时间不可用。但全网仍需要有较强的鲁棒性。
- **所有权和隐私性**：数据所有者具有数据的所有权和完全访问权，数据是加密并具有隐私性的。经过所有者授权后其他角色才能访问和使用数据。
- **可量化的贡献度**：参与协议各方的贡献度都应该有相应的量化标准和可被观测的贡献度。比如采用 PoST 和 PoR 作为存储空间和存储时间的量化证明。
- **最终状态一致性**：允许数据对象在不同节点处于不同的状态，但是其最终状态能快速收敛获得全网一致性。
- **可监测和可恢复**：能检测整个网络的可用性、数据对象的全网状态，并根据相应的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运行时，能进行一定程度的监管和审计，前提仍是数据所有者知悉并愿意在此种监管框架下进行数据的存储。
- **可扩展 API**：易用的 API 具有很高扩展性。

# 4、 豆匣协议

豆匣协议首先构建的是分布式存储网络，可以提供高性价比的存储空间，安全可信的数据保存，透明公开的上链信息。其次豆匣协议希望能对数据进行证明变成用户的数字资产，最终围绕存储空间、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（闲置磁盘、云盘）通过豆匣协议加入豆匣网络。

## 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。
- 定义 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。
- 设计了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）** 在豆匣协议中的数据豆（Bean）是未分片（Piece）前的数据对象。数据豆除开分片外不能被单独拆分，数据豆对外部系统具有唯一性。
- **数据所有者（Owner）** 数据所有者是数据豆的拥有者。所有数据豆都有所有者的签名。数据所有者有权随时对保存的数据进行检查，以确保数据确实安全地存储在存储节点。

  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（Boxer）** 存储者提供磁盘空间用于存储数据，并以此获得相应的报酬。存储者还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，以证明用户存储的数据在约定时间内是一直存储在磁盘上，并随时可以被访问和使用的。下文称呼的“存储节点”实际上是运行了存储客户端的物理节点。
- **观察者（Watcher）** 观察者需要观察整个网络系统的运行情况，包括检测和存储数据豆的状态，基于数据安全策略检测安全性，并修复已存在的或潜在的安全性缺陷。因此观察者必须能稳定在线的，同时也扮演了让全网状态快速收敛的必要角色，是数据豆索引服务的不二人选。

观察者会不定期的对存储者进行心跳检测，以确保数据的可用性。同时观察者还可以接受数据所有者的委托，帮助数据所有者对存储者发起数据存储校验，以确保数据是安全和可用的这些工作大多数都是在链下完成。我们希望观察者节点是一个独立的节点，这样可以和出块节点及数据存储节点形成制衡关系。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**）** 出块者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、吞吐率和出块者有极大的关系。所以出块者一般由全节点来担任，以保证稳定在线和较高的处理效率。目前只有连入豆匣链（Sharder-Pool$_0$）的全节点才能争夺出块权，出块权的争夺上会采用 PoS 或 DPoS。
- **证明者（也称证明人）** 是为了让豆匣网络中存储的数据具有公信力，从数据衍变成数字资产而在豆匣协议中内置的一个角色。证明人所提供的证明数据和原始的数据对象会关联在一起并记录上链，具有可追溯和不可篡改性。

  证明人很多时候需要由豆匣网络外的机构或有公信力的组织担任。只要部署了证明人节点加入豆匣网络即可行使证明人的权利。比如数据版权领域，最具权威的证明人应是国家版权总局，应该由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。在数据保全的场景中，公证处和司法机构则是具有公信力的证明人。只要经过它们出具相应数据的公证书就可认为是该用户的数字资产具有司法效力。
- **全节点（**Full-Node**）** 全节点是一些稳定在线，具有较好的网络带宽和处理性能的物理节点。在豆匣协议中不仅承担了重要的出块者角色， 同时可以扮演存储者和观察者角色。不过我们希望观察者和出块者是相互制衡的，最好不由同一个全节点同时担任这两个角色。
- **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（类似于当前比特币网络里的矿池）。现有的公链或存储网络只要部署了豆匣协议也就组成了一个豆匣池。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豆匣链构成的 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。形成的封闭豆匣池就类似于私链一般，形成了一个私有云存储网络。按照豆匣协议提倡的开放标准，豆匣池选择封闭是需要支付费用的。

- **豆匣网络（**Sharder Network**）** 由所有部署了豆匣存储协议的全节点和豆匣池组成了豆匣网络。豆匣网络宏观上是个大型的存储网络系统，希望最终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还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需要记录到豆匣公链中。宏观看豆匣链在豆匣协议中充当了多链结构的中间锚定网络，类似于传统通信网络中的骨干节点。

  当然更长远地我们希望豆匣链基于豆匣网络之上是自治和自洽的帮助管理数字资产的网络。
- **豆匣市场（**Sharder Market**）** 豆匣市场是豆匣网络中的去中心化交易市场，在过去几年比特儿（BitShares）和以德（EtherDelta）都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最终也会演化出真正基于供需所产生的存储定价和自由交易市场，可能的演化进程是早期由豆匣链提供撮合服务，逐渐变成由存储买卖双方自己出价而豆匣链仅仅是一个交易采集者和价格分析者，提供参考价格和历史成交价给买卖双方。当然任何一个全节点只要有完整的豆匣网络中的区块信息，都可以提供价格参考和撮合。

  当然更远的未来，我们希望为数据存储提供检测、证明、纠删等服务者也能加入自由市场，围绕豆匣网络中的数据提供各式各样的服务。服务的定价交给市场经济这张看不见的手，而服务的质量和最终交割交给豆匣协议来保证。区块链技术的分布式账本、智能合约、加密代币、密码学能够让买卖双方在无需信任对方的情况下完成存储资源的交易。

## 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。

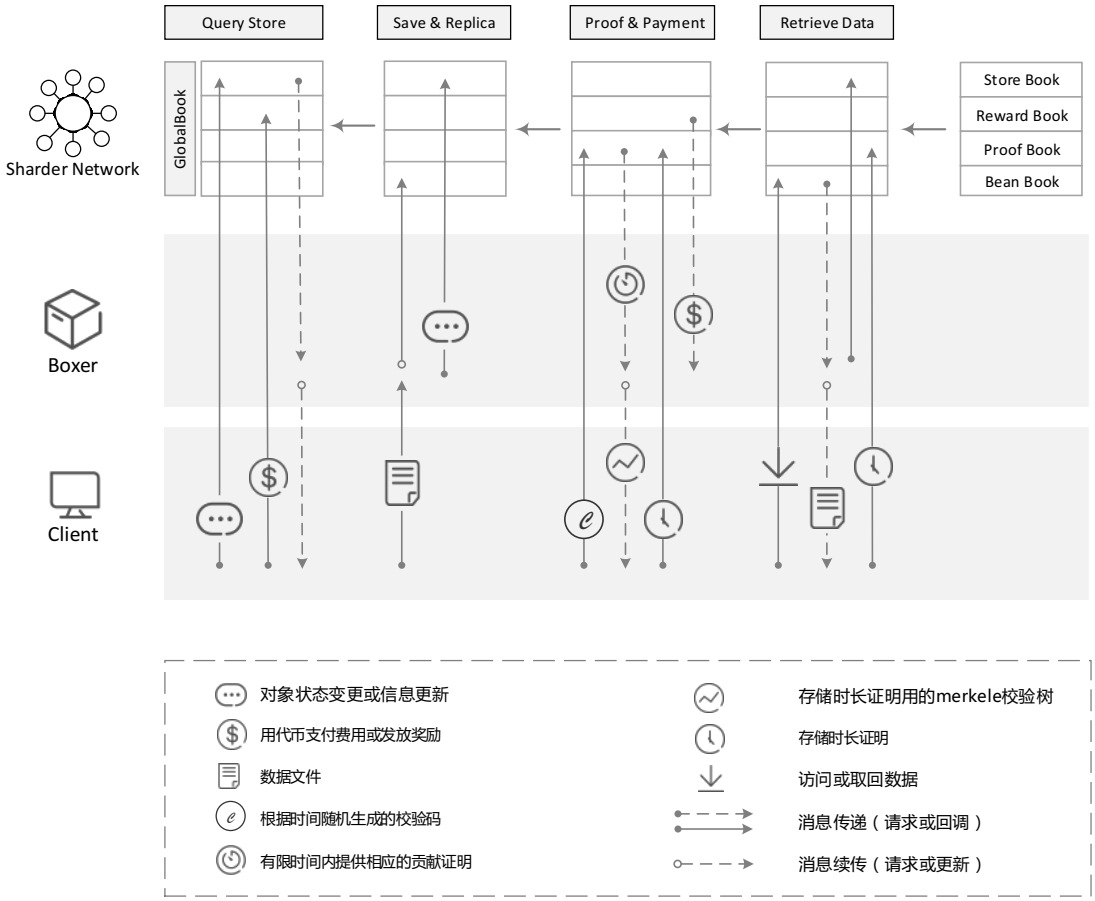
## 4.4 数据对象操作

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

•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(data) → key: 客户端使用数据的唯一标识 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。状态同步并根据数据的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（Boxer）。

3、 客户端（Client）上传文件到存储节点（Boxer）。

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

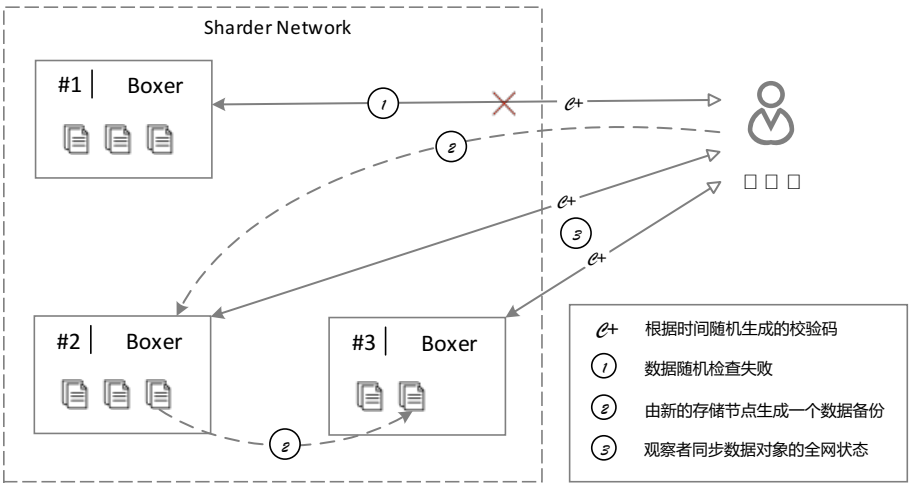
1、 客户端（Client）发起取回数据请求，豆匣协议从对象账本（Bean-Book）中获取最新的数据对象返回给客户端，并向存储节点同步此数据取回请求。

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（Boxer）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可能会要求其他的存储节点再备份一份数据。但某些情况下原来的存储节点只是临时离开网络或意外宕机，当重新连入网络时就会出现到底谁才应该继续作为数据的保存者并获得存储报酬的问题？对于这种情况，豆匣协议会根据该节点的豆匣账户信用等级来进行判断。原则上是信用等级高的存储节点会被允许继续存储该数据并获得存储报酬，信用等级低的需要删除数据。

**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个或则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和不断“调整”全网的数据分布。所以观察者可以向用户提供最新的数据对象的资源访问地址。

### 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（Boxer）根据校验码 C 生成相应的存储证明 $\mathcal{M}$。
3、 存储节点（Boxer）在有限时间内将存储证明 $\mathcal{M}$ 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（Boxer）。

引入 merkle 树[X]和 zh-SNARK[X]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

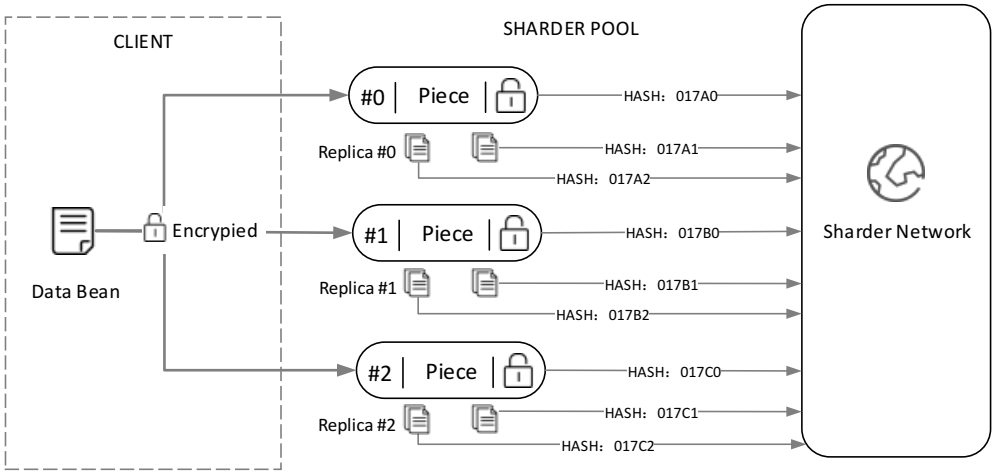
存储数据时候对一份数据的备份耗时，以及观察者对数据进行检查并调整数据分布的时间耗时可以认为是数据对象的收敛性证明问题，以下我们进行概要性证明。

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，数据豆被分片成 p 份，每个数据分片的备份数为 n。则能成功取回数据豆的几率 $R_s$ 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:数据豆的分片数量　n:数据豆的备份数量

---

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”。所以豆匣协议中允许数据所有者根据自身的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数据丢失的选择。同时也会影响到数据对象在全网的状态收敛的速度。

## 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。

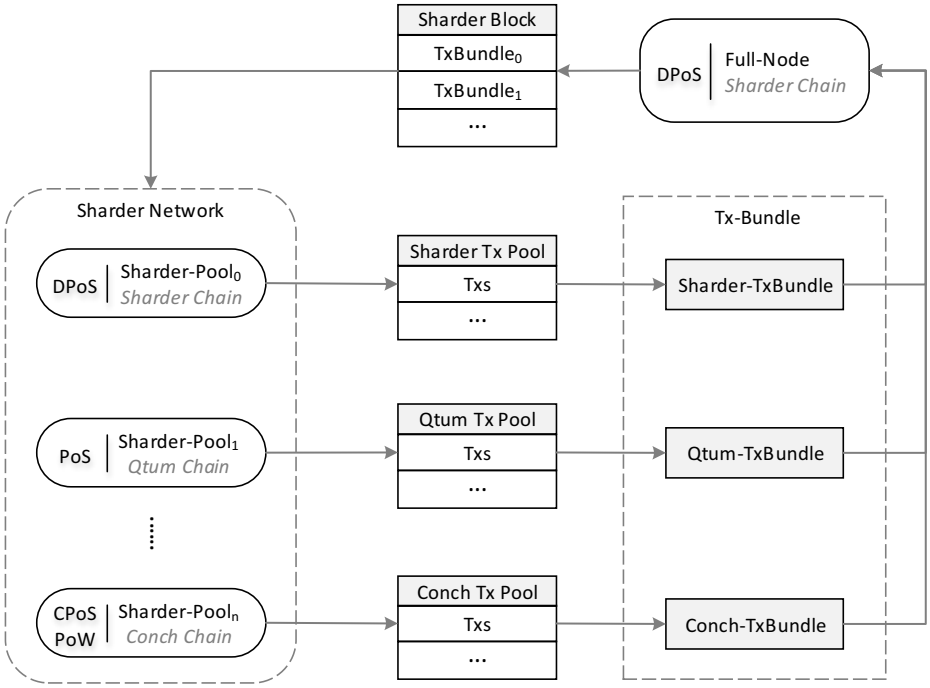
## 4.7 共识和出块

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，消耗掉大量的电力和计算资源仅仅是为了争夺出块权，我们认为这是一种浪费。随着矿工对硬件的“军备竞赛”不仅增加了对硬件资源的过度消耗也大量增加了电子垃圾。我们希望提供共识出块能在保证网络安全的同时，又让每次为了争夺出块权的计算变得更有价值。

- **共识出块** 多链共识出块的方式如下图所示，核心由交易包（Tx-Bundle）、豆匣区块（Sharder Block）组成。这种方式允许每个豆匣池在自己内部有各自的共识

生成交易包，交易包在一个豆匣池里就等于一个包含了许多交易记录的区块。最终全节点会生成豆匣区块并公布到豆匣网络中，每个交易包（Tx-Bundle）里需要包含豆匣池和节点的数据信息： Node-ID，Pool-ID，Area-ID。

一个全节点只能选择连入一个豆匣池，如果为了能打包豆匣区块，该节点需要连入豆匣链（Sharder-$Pool_0$）。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都部署至少一个连入了豆匣链的代理节点（Sharder Agent）。




图片 5 多链共识出块

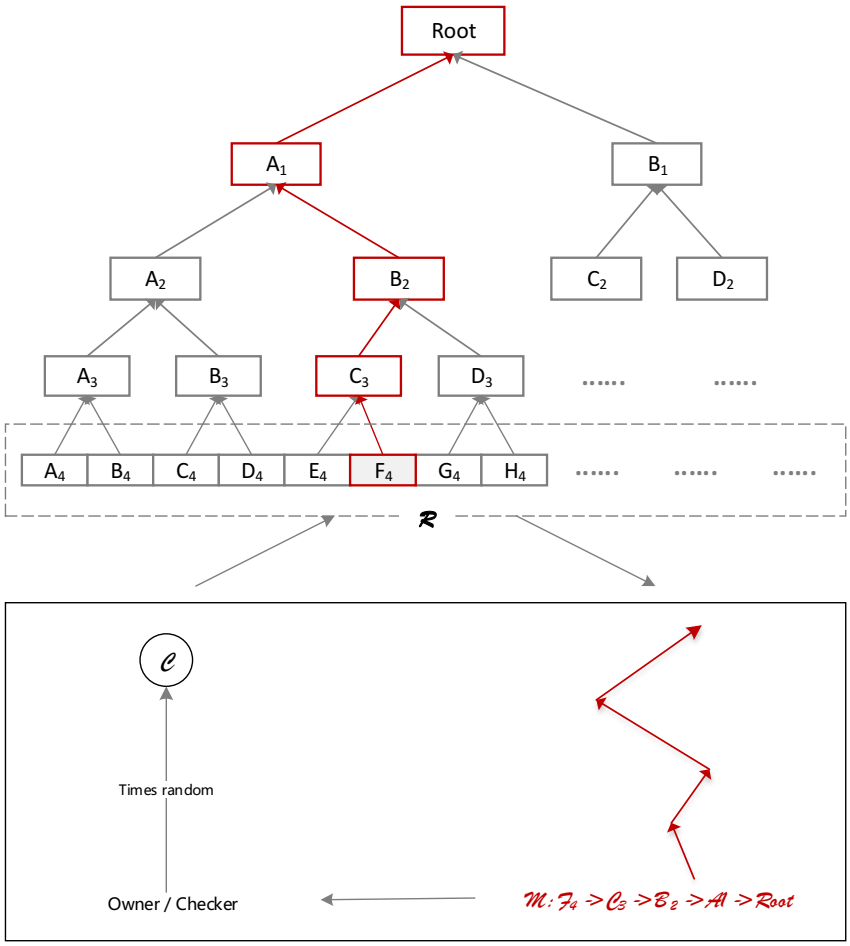
- **信息上链：** 不是所有的数据和信息都需要上链，尤其是在豆匣网络中大部分的数据都不会上链。比如：数据文件就不会保存在链上，链上保存的对象 URI 就如一个指针，指向了当前这个数据对象的可用资源地址。

  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 的检验而发放存储报酬）。

## 4.8 贡献度量化

引入 merkle 树[6]和 zh-SNARK[7]构成 PoR( 备份证明 Prood-of-Replica )和 PoST ( 存储时长证明 Proof-of-Storage&Time )作为存储者( Boxer )存储数据的量化凭证。可信存储节点可以采用 PoR 能够用较短时间就可以提供备份证明，对于信用评级较低的存储节点会要求使用 PoST 的方式保存数据和提供存储时长证明。

### 4.8.1 备份证明 Proof-of-Replica



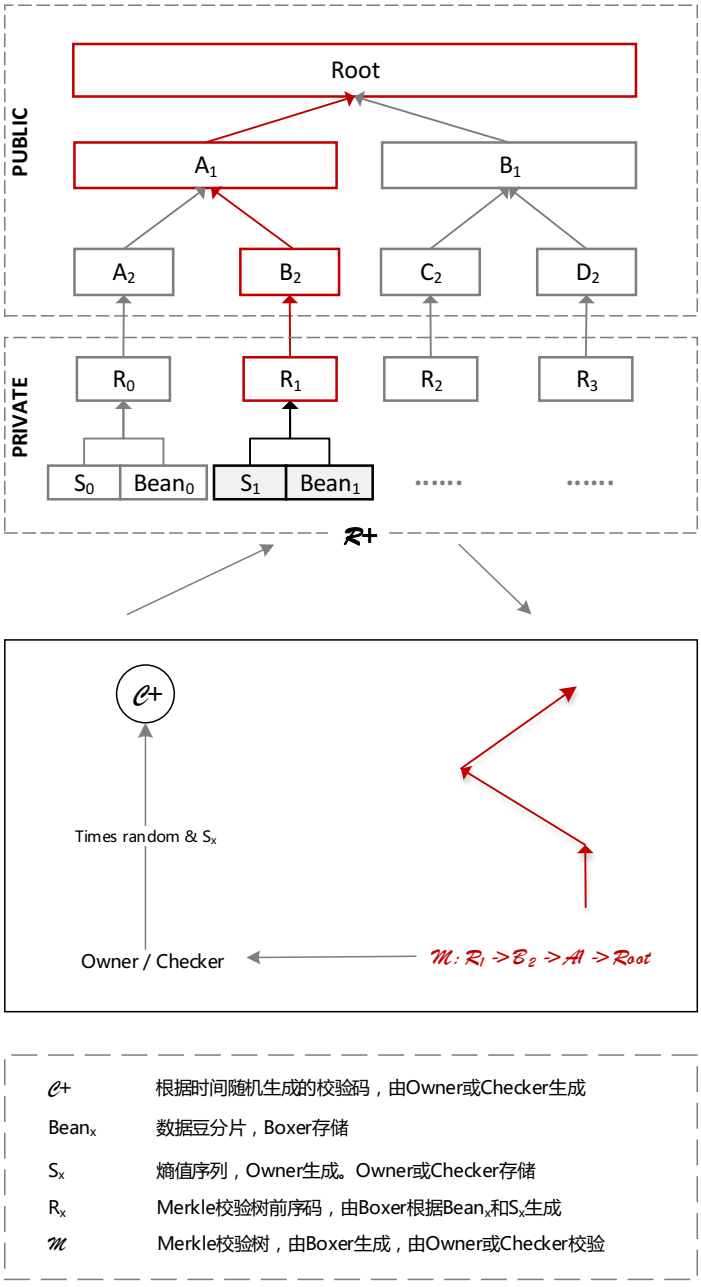

图片 6 备份证明 PoR 流程图

为了确保存储者会一直保存数据备份，数据所有人可以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$\underset{M}{\rightarrow}$(merkele 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），使得存储账本（Store-Book）中该笔交易的部分奖励金额能解锁，作为存储者的存储报酬。

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




图片 7 存储时长证明流程图

PoR 虽然能保证数据存储者至少会保存数据分备份一次，但是无法避免恶意存储者进行欺骗，考虑以下场景：

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 merkele 检验数后，删除数据分片文件只保存 merkele 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$\underset{M}{\rightarrow}$(merkele 校验树)返回。

因此我们提供了 PoR 的加强版 PoST，以保证只要数据存储者删掉了数据分片文件就无法计算出正确的

merkele 校验树，无法通过校验自然无法获得存储手续费。

- 数据所有者在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，不能重复）请求相应的存储时长证明。此时存储者需要根据 $S_x$ 和对应的数据分片计算出 $R_x$，并根据 $R_x$ 生成相应的 merkele 校验树。

有了前置熵值序列，可以实现观察者代理数据所有者进行检查的功能。数据所有者可以在检查时间提供一部分熵值序列交由观察者，由观察者来完成 PoST 的证明校验即可。这种代理方式未来也可以通过智能合约来更加方便地实现。

### 4.8.3 信用证明 Proof-of-Credit

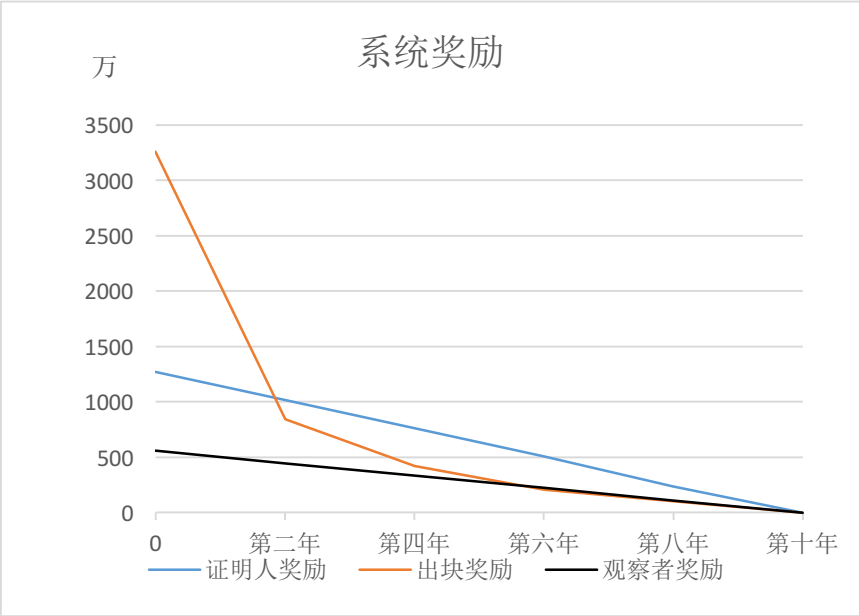
在豆匣协议中信用证明是和账户绑定的，在海螺链 CPOS 评分体系[8]基础上根据不同角色相应的计算公式侧重点会有所不同：

- 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（最快能达到 10 秒产出一个区块）。每天能够产生 288 块区块，一年能产出 105,120 块区块。每产出一个豆匣区块对出块人提供 80 个 SS 的奖励,每产出 210,240 个区块奖励减半，840,960 块出块奖励会保持在 5 个 SS。预估 10 年完成 32,587,200 个 SS 的奖励发放。出块奖励发放完后，将由存储交易手续费和 B 端用户提供的技术服务费作为豆匣网络维护的经济激励。
- **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，豆匣网络将遵循豆匣协议指定观察者，系统预留了 5,593,220 个 SS 作为观察者奖励，奖励数量将根据观察者在线时长、索引服务性能的不同来进行计算，这笔 SS 奖励会被智能合约自动执行分配给观察者。观察者奖励发放完成后，观察者、证明人、数据所有者、存储者之间可在豆匣市场进行自由交易。
- **证明奖励** 豆匣网络预留了 12,711,864 个 SS 作为证明人奖励，激励和邀请有公信力的组织或机构担任证明人，使单纯的存储数据衍变成真正意义的数字资产。豆匣市场也允许数据所有者和证明人之间设定他们的需求价格，来自由公证交易。

### 4.9.2 系统惩罚

豆匣网络会对以下危害网络的情况进行系统惩罚，罚没其豆匣币并降低其信用评级。

- **丢失数据** 失去后续存储数据应得的报酬。并会降低其信用评级，到达阈值后该用户会被加入黑名单以后无法再加入豆匣网络。
- **恶意攻击** 不管是为了蓄意发起攻击而不出块，大量的通过非正常手段获取 SS，对豆匣协议所部署的物理网络和程序进行攻击都将会触发最高级别的惩罚。不仅会导致该节点和关联用户被加入黑名单以后无法再加入豆匣网络。会罚没该账户的豆匣币。
- **欺诈** 欺诈经常会被发现于事后，目前无有效办法追回已经支付的报酬或则系统奖励。只能降低其信用评级，到达阈值后该用户会被加入黑名单以后无法再加入豆匣网络。对于欺诈最好的防范办法是增大其欺诈的成本，会根据实际的运行情况可能会要求某些情况下需要节点缴纳一定的保证金。

### 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** 提供存储空间的存储节点可以获得数据所有者所支付的存储报酬，存储报酬由市场价决定。作为普通用户如果愿意运行存储客户端的话，可用通过提供自己的存储空间在豆匣网络中获得相应的免费存储空间对自己存储的数据进行备份。
- **服务报酬** 未来在豆匣网络拥有了足够活跃的自由市场后，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）而获得相应报酬。

## 4.10 豆匣币 （SS-Sharder Storage）

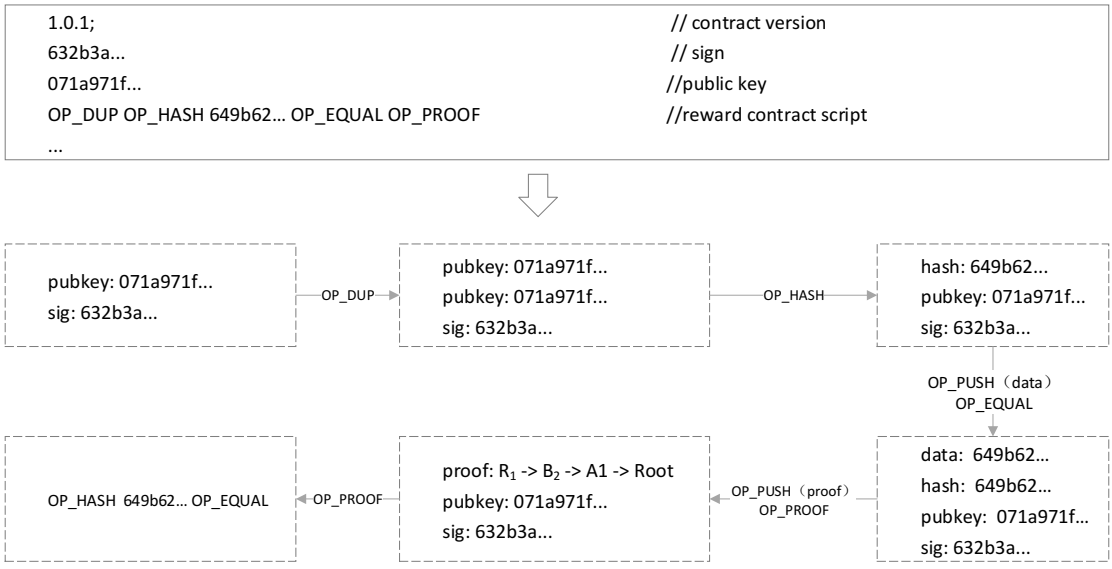
豆匣币是豆匣协议中的内置加密数字代币。主要用于激励豆匣协议构建的存储生态，奖励对豆匣网络做出越多贡献的角色，同时使用豆匣币作为经济惩罚的手段，避免恶意节点

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 StoreBook 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的变更和费用的支付。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，观察者会是智能合约的其中一方。智能合约也会基于账户模型并实现图灵完备。在未来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。每种客户端有以下几种类型的版本：

- **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。运行该客户端的客户端仅仅作为存储节点存在于豆匣网络中。在该版本中除开可以使用你本地的磁盘空间外，我们还会尽量多的集成目前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。
- **观察者端** 运行该客户端后可以成为独立的观察者，监控豆匣网络中数据对象的状态变化、可用性、并向网络中发送指令修复存在的安全性和可用性问题。
- **证明人端** 运行该终端后证明人可以查看并对需要进行证明的数据进行证明操作。当然如果证明人有独立的信息系统，则会采用 API 的方式进行集成。

将来为了方便其他豆匣池和存储网络能够部署豆匣协议，我们还会提供相应的 SDK 包和 API。所有的客户端都会包含钱包和转账功能，还会提供手机版的轻客户端帮助用户方便地进行转账和管理数据文件和数字资产，轻客户端需要连接到全节点才能正常工作。

## 4.13 多链生态

多链生态由部署了豆匣协议的豆匣池构成。

- **交易交换** 每个豆匣池内部的交易包（Tx-Bundle）包含了豆匣池内部特殊的交易数据类型，但是这些交易数据都会被包含到豆匣区块中再被广播到豆匣网络。不同豆匣池同步豆匣区块后在本地都会有相应的交易包。
- **价值交换** 每个豆匣池可以有自己内部发行的代币，依靠豆匣链和豆匣币完成币币交易形成价值交换。我们希望不用在豆匣池内再发行豆匣币 $SS_n$ 就能完成这一过程。目前我们只能在部署了豆匣协议的豆匣网络中，在不同的豆匣池之间构建多链生态。我们首先会实现并测试豆匣链和海螺链构成的多链生态。未来还希望能够在豆匣网络外，完成交易、价值、账户、数字资产的跨链。基于同态通道的闪电网络和未来强大的智能合约应该是豆匣网络和其他区块链网络互通的好选择。

## 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、交易双方组成。豆匣网络里的自由市场不是超高频的交易市场，目前的架构因为涉及全网账本的状态更新和同步也不能支持超高频的交易。如果未来豆匣市场被证明是高频甚至是超高频的，我们会调整架构将信息流和资金流分开。交易部分的撮合移到链下，链上部分只做资金的清结算。

- **全网账本** 当前有以下全网账本， 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** 交易双方必须是运行了豆匣网络客户端的用户才能进行交易，交易方式可以是是使用智能合约挂单购买交易。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** 由于没有了中心化的坐市商，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，仅根据价格进行撮合匹配。

- **自由定价** 价格可由买卖双方自行制订。但是如果不存在一个参考的市价，买卖双方无法很好的进行定价进而促成交易的完成。早期豆匣链会作为价格的采集方和公示方，起到撮合的作用。
- **手续费** 如使用智能合约，手续费是执行智能合约所需的豆匣币。如果由中间的交易/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手段，比如帮助他人进行撮合或则确认交易即可免除本次的交易手续费。
- **豆匣链** 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（尤其是企业用户）。

## 4.15 授信框架

区块链是开放自由的，但是当前的各种场景中仍然需要一定程度的监管和审计。当让这种这种监管和审计应该是在双方知情的情况下开展和完成的。基于此豆匣协议提供了一个基础的框架（我们称它为框架的原因也是因为它确实足够简单，但是我们认为有必要加入这一概念。在以后研发和运营过程中，获得更多经验后我们会进一步完善。）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代理用户对数据进行访问和使用。

- **账户授信** 首先信息的分级和审计需要在数据所有者的知晓和授权下。借鉴 BIP39[9]多级分层确定性钱包的思路来让数据所有者能够进行授信。考虑未来数字资产都是管理于某一账户下的，我们会参考 BIP44[10]的路径定义方式和以太坊 EIP85[11]中的讨论，选用如下路径：

  m/purpose'/coin_type'/ account' /change/address_index
- **审计** 区块和交易信息虽然是公开的，但是由于交易双方的匿名性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。审计方需要重复账户授信的过程直到获得双方的授信后才能完成交易信息的审计。所有的审计结果也会上链，意味着审计的操作也是开放的。当然豆匣池或则上层的商业应用可以进一步设计批量账户

授信，避免审计者重复账户授信的冗长过程，因为当前的很多使用场景下，C 端用户是信赖 B 端平台的。

- 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部署豆匣协议的豆匣池自行决定如何实现。但是可以使用豆匣网络加密存储脱敏后的用户身份信息文件。
- **信息分级** 信息分级等同于数据读写权限的控制，一个可行的思路是不同的数据对象根据分级使用不同的衍生密钥（不同的 HD 路径）进行加密，这样获得衍生密钥的授信就仅能解读这一部数据。但这会导致比较复杂的私钥生成和数据分片的加密逻辑，将会放到未来进行详细设计和实现。

未来会根据实际的运行情况在此框架内部加入更复杂的角色权限控制系统，从数据权限和功能权限两个方面进行更细颗粒度地控制。

## 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。在豆匣协议中也存在 51%的攻击问题，要想完全避免这一问题是不可能的。为了降低被攻击的几率早期在豆匣协议中采用 PoS 和 DPoS 来出块，后期还会结合 PoC 来挑选合适的出块者。
- 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，可以极大的降低 sybil 攻击的几率。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。随着越多节点的加入，sybil 攻击的概率会大大降低。在早期为了避免该攻击，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避免 sybil 攻击。
- **数据欺诈** 如果存储节点在收到随机校验后，在极短的时间内从附近的数据存储节点获取数据并计算出正确的 merkele 验证路径，即使该节点并未存储数据也能通过校验并获得奖励。采用 PoST 可以极大降低这种概率，存储节点必须持有前置熵值数据文件而该文件只应存储于本节点，无法从其他节点获取。对于进行数据欺诈的节点在被要求提供数据的时候会因无法提供数据还会受到相应惩罚，比如：降低其信用评级。

- **数据劫持** 存储节点不向数据所有者提供最后一个数据分片使得无法组装成可用的数据豆对象，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，也可以从其他备份的存储节点取回最后的数据分片。除非所有拥有该分片的存储节点都被恶意攻击者控制。
- **数据抹除** 存储节点在一定的存储时长后，认为当前的数据分片存储的奖励额度过低（当前存储等量数据分片的奖励变高）或则纯粹因为不再想存储该数据分片，而删除该数据分片而放弃后续的存储奖励。面对这种情况数据分片分片的多备份可以降低给数据所有者带来的损失，还可以调整 PoST 的奖励策略，奖励随着时间的增加而增多。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。具体说我们假定一种策略：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 纠删码之后。会根据实际情况考虑实现 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

我们看到当前已经存在证明你银行资产的方式。在很多房地场销售大厅已经有智能设备可以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。资金还在购房者的银行账户里，购房者不用像以往一样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。这对于双方来说都方便了许多。但是非银行体系的数字资产仍然没有很好的办法便捷地进行证明。甚至你很难方便和可信地证明证券账户里的的持仓股票是属于你的。豆匣协议通过证明人角色对存储于豆匣网络中的数据进行 POA，从而提供可信地数字资产证明。区块链溯源和不可篡改的特性构成了完整的证据链条能极大地帮助证明人完成 POA。

当有了具备公信力的数字资产证明后，就能容易地进一步进行数字资产交易。基于智能合约，可以在弱信任或无信任的情况下完成数字资产的自动交易。场景是生成一份智能合约，并从一个资产地址扣除一定的定金/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，可以依赖智能合约的虚拟机和预言机来实现）自动执行合约。

### 4.17.1 豆匣文件系统

豆匣文件系统可以增加单节点的读写吞吐性，至少可以提高区块链层里的数据库吞吐性能，比如可以允许并发多进程读写。对于大量的小文件或碎片文件的读写问题也能提高其 IO 性能。有了豆匣文件系统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、对抗网络等学习模型都有了长足发展。但在无监督学习等领域仍然有待开拓。区块链本身就是一块有着多种多样开放数据的天然土壤，而豆匣协议更是一个分布式存储协议构建的正是存储数据的分布式网络。如何给这些上链和存储于豆匣网络中的数据贴上标签、分类、训练 AI 是非常吸引人的事情。AI 的不断学习也有助于帮助豆匣网络更加智能、安全、高效。短期内能够预见的是 AI 的训练能帮助改善安全策略，让观察者更好的“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网络。

## 5、 豆匣链 （Sharder Chain）

第一个部署了豆匣存储协议的商用区块链网络。是构成豆匣网络（Sharder Network）的一个重要的豆匣池（Sharder Pool）。豆匣链还充当了分布式数据库的作用，可以认为豆匣链是构成豆匣网络的基石。豆匣协议的特性都会在豆匣链中实现和测试。

豆匣链定向为企业用户提供 toB 区块链存储服务，因此增加了一些相应的商用特性：易用的账户模型、数字资产、担保交易、商用定制化 API、运营支撑。

## 5.1 功能模块


图片 10 豆匣链功能结构图

**区块链层** 构成了基础的区块链功能。组成了点对点的对等网络，有了 UTXO 模型的分布式账本和全局 Book 模型，内置了豆匣协议里原生的豆匣币。

**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。

**资产层** 构建了友好的豆匣账户模型，将代币余额和数据对象和账户关联形成数字资产模型，提供基础的资产证明。

**组件层** 为了方便的使用豆匣协议和豆匣链，完成了一些基本组件的抽象和封装。并基于智能合约提供了担保交易模型。

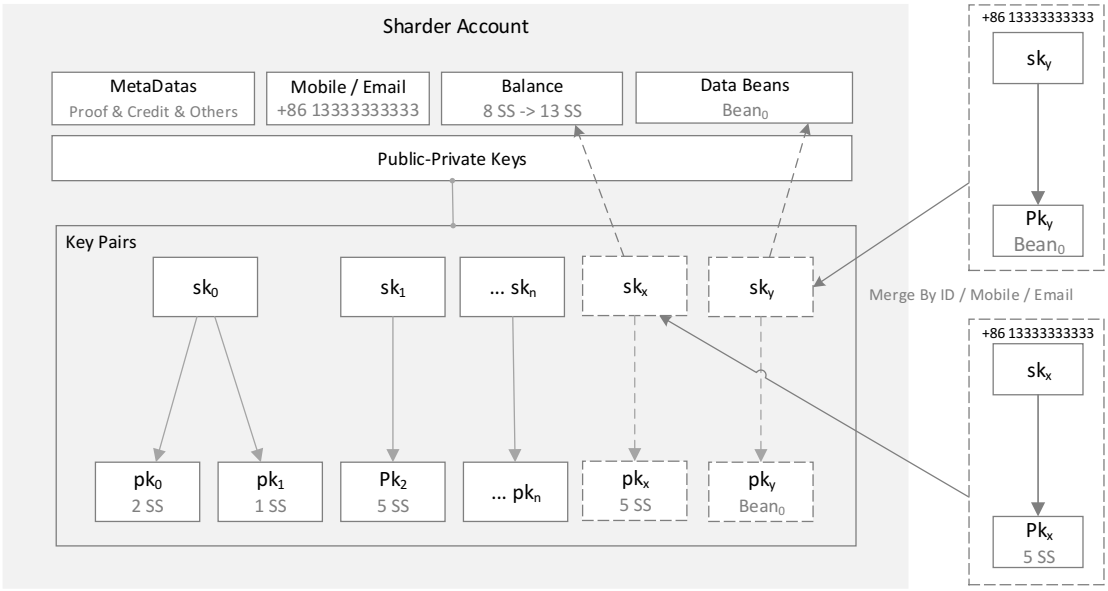
**接口层** 便于合作伙伴和商户使用区块链服务和分布式存储服务。

**贡献量化** 从单纯的算力证明演变成各种角色对豆匣链和豆匣网络的贡献度证明。根据相应的贡献度而获得相应的奖励。当然作恶也会受到惩罚。贡献度会和豆匣账户进行关联。

**运营支撑** 便于商户接入豆匣链使用豆匣链的服务，并提供相应的数据统计和分析帮助豆匣链改善运营质量。

## 5.2 豆匣账户

为了符合使用习惯豆匣账户不再是一个个零散的地址（由对应的私钥管理）。而会以手机号和邮箱同时作账户的身份标识，记账模型沿用比特币的 UTXO 模型， UTXO 这种记账方式有极强的可追溯性和审计性。考虑未来数据资产和其他数字资产应该隶属于某个账户，在 UTXO 模型之上引入账户模型。




图片 11 豆匣账户图

豆匣账户下管理的公私钥就如比特币提出的 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，会进一步实现 BIP44 提供一组助记符）。豆匣的账户模型会自动根据账户的身份标识归并余额和数字资产（不会再出现让普通用户困扰的比特币 UTXO 模型导致的找零地址），统一管理在一个豆匣账户下。用户只需要安全保存创建豆匣账户的种子私钥就可以保证整个账户的安全性。使用椭圆曲线秘钥算法[13]生成秘钥，签名使用 EC-KCDSA 来实现[14]。

当然这种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，获取所有区块上的交易信息才能归并出账户信息。

## 5.3 数字资产

能够预见到，随着人们生活的数字化，各种实体资产也将数字化。豆匣协议提供了分布式数据存储网络帮助企业和个人存储各种数据。在豆匣链中会更进一步帮助用户将这些数据资产化。一个可行的方案是借助区块链公开透明、可溯源、不易作伪的特性和豆匣协议中的证明人角色，对这些数据提供 PoA（Proof-of-Asset）使这些数据在日常世界具有相应的公信力和法律效力。

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，对于豆匣网络外的世界只能提供数字资产的证明。和外部世界的数字资产流通仍然需要外部的证券交易所或新兴的数字资产交易市场来完成。豆匣链会和这些交易所进行对接或则依赖一些去中心化的交易撮合协议来完成数字资产交易。

## 5.4 担保交易

为了让豆匣链的使用者：DApp、商户能更方便地使用智能合约进行交易，并符合目前的使用习惯。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）。我们早已习惯淘宝的担保交易，它一定程度保证了卖房资金的安全性。但是淘宝的担保需要淘宝作为中间的背书机构才能保证买家在收到货后才会将资金支付给卖方。豆匣链中的担保交易使用智能合约来替代淘宝在担保交易中的背书机构。

担保交易创建的时候会自动生成一份智能合约，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），当买家按照智能合约支付了约定的 SS 后卖家的收款地址后，智能合约会将资产转移到买家的地址。当然这里可自动交割的数字资产仅限于豆匣网络内的数字资产。未来随着证明人的接入，依托于豆匣网络外部担任证明人角色的征信机构，担保交易能够交易的数字资产会逐渐增加。

预授权模式的担保交易：买卖双方都需要支付一定数量的 SS 作为保证金。当在竞拍模式有多个卖家参与的情况下，所以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，即是这些地址的状态会变更并被同步到全网。智能合约同时会持有卖家和买家的时效性私钥 tpk，可以形象地理解为信用卡的预授权模式。


图片 12 时效性密钥图

交易成功的时候，智能合约会自动使用买家和卖家的时效性私钥完成代币的划转和数字资产的转移。反之则锁定的代币地址和数字资产地址会解锁交易会取消。相应的保证金会被罚没，同时会影响账户对应的信用值（PoC）。

## 5.5 监管和审计

当前的大量的使用场景仍是 C 端用户信任 B 端平台，C 端用户在使用 B 端平台过程中的个人信息、数据、文件、交易信息、支付流水等数据 B 端平台都能访问和使用。在豆匣链中会实现豆匣协议中定义的授信框架，并提供批量授信机制以完成 C 端用户向 B 端平台授信，B 端平台向监管机构授信的逻辑。

豆匣链会采用 API 的形式将授信功能开放给上层的商业应用，商业应用需要自行保管相应的授信私钥，每次涉及 C 端用户的账户的操作都需要使用该授信私钥进行双重签名。如若丢失需要在 C 端用户的再次授权下才能重新获得操作权限。由于 B 端商户保存了相应的授信私钥，所以 B 端用户能向 C 端用户提供私钥找回服务。这在一定程度上也方便了不了解区块链的普通 C 端用户使用。

## 6、 豆匣社区

比特币从中本聪的论文提出到上线一直运营到现在，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、大众的智慧持续改良豆匣协议。

**线上平台** 官网、豆匣社区(基于 RocketChat 搭建)、官方 QQ 群、官方微信群等渠道构建社区成员自主发表想法和意见的社区。并组建豆匣理事会，由豆匣基金会成员、币圈资深人士、社区成员三方组成。主要负责豆匣理事会的日常工作和维护豆匣社区的正常运营。在社区里可以进行类似于比特币社区一样的规范讨论、发展规划讨论、提出改进意见、贡献办法等。对于积极参与社区的用户我们会给予豆匣币的奖励。

**奖励规则** 豆匣理事会总基金为 10，000,000 豆匣币 ，理事会对社区用户申请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：S=N%×M+50×N,S 为豆匣币空投数量，N 是整数因素（0<N≦5）,M 是空投地址原持币数量（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。

**社区远景** 我们认为豆匣协议的发展，既需要区块链技术爱好者和豆匣团队一起来参与代码编写、审核、测试工作，也需要更多的公链、企业、个人部署豆匣协议。和豆匣团队一起讨论协议存在的问题和可能的发展，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。最终形成自由、开放的豆匣存储服务市场。详细的运营细则和奖励规范请参见“豆匣社区运营白皮书”。

## 7、 应用领域

纵观国内目前第一梯队的互联网科技公司，是大量的长尾用户、大量的交易、高频地使用推动了它们的快速发展和技术革新。我们坚信将豆匣协议和豆匣链商业化，有了更多的商户和用户使用，更多网络部署豆匣协议，才能有长远地发展。

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内的长足发展。我们正在和合作伙伴一起研发以下落地的商业应用。

## 7.1 云存（存证和保全）

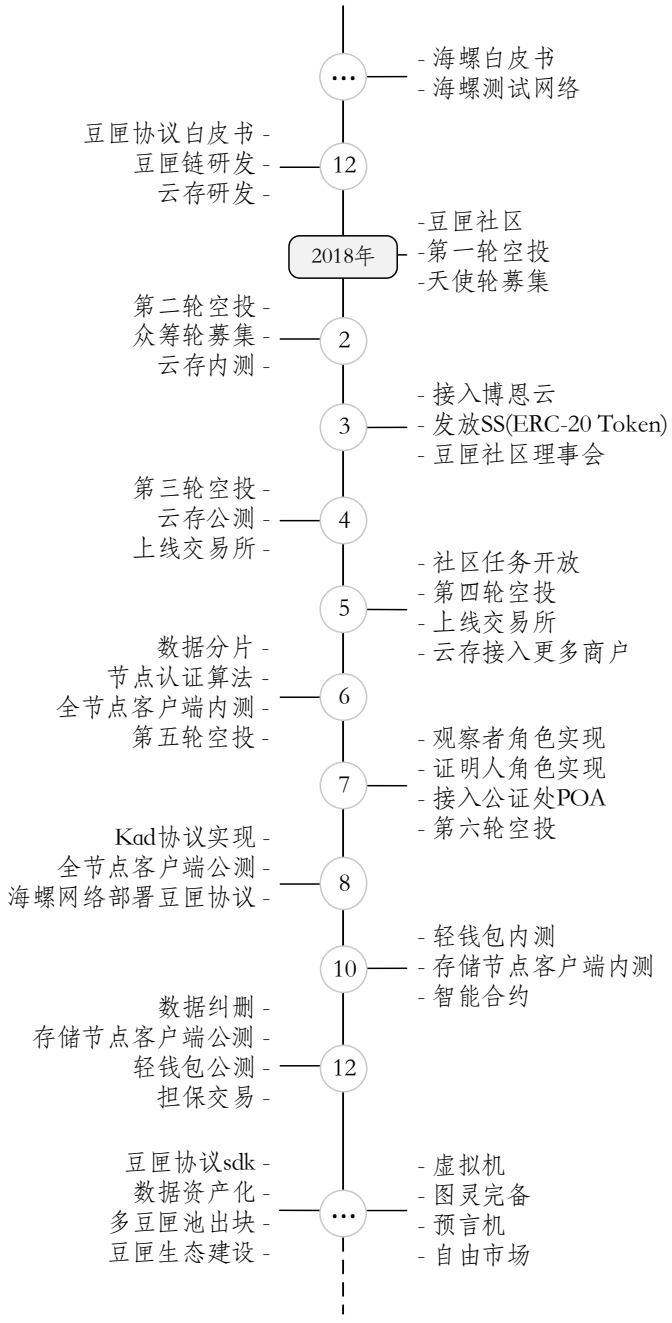
面向企业用户的数据存证和保全平台。主要服务于 P2P 网贷、小贷、消费金融、电商、ERP 系统，可以将电子合同、支付凭证、投资记录等电子数据保存于豆匣网络中，利用区块链可溯源不可篡改的特性对已上链和保存于豆匣网络中的数据出具保全证书和司法证明。

## 7.2 One Fair（原创作品版权）

基于豆匣链构建的原创作品交易平台。不仅能帮助民间创作家将原创作品安全地存储于豆匣网络，还提供版权证明和保护。最后还能将原创的数字作品资产化依托于豆匣链的担保交易方便快捷地完成交易。

# 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有无限可能和遐想的领域，在这个领域里有太多值得去探索的，基于之上的商业应用部分更加广袤。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，豆匣基金会也会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<br>云存 | 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

感谢成文过程中张明镜、左欣荣等提出的修订意见。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。