Storj技术详解

1.Storj介绍

1.1 Storj简介

Storj，点对点云存储网络(A Peer-to-Peer Cloud Storage Network)，它本质上是一个基于区块链的端对端加密的分布式云存储平台，用户通过私钥来管理数据。比传统的云存储平台更快，更便宜，更可靠。

- Storj是一个协议
- Storj是一个去中心化的分布式云存储平台
- Storj是一个区块链界的"百度云盘"

1.2 中心化云存储的不足

- 基于信任模式(严重依赖第三方)
- 端到端加密标准缺乏
- 安全威胁:中间人攻击、恶意软件、黑客
- 维护代价高

1.3 Storj特点分析

Storj是一个去中心化的云存储平台，主要解决了两个问题：

- 数据的安全问题，完整性，有效性

  端到端加密机制

- 可拓展性问题

  经济驱动

Storj通过一种带奖励的哈希挑战/质询算法/心跳(a challenge-response verification system coupled with direct payments)，定期检查数据的完整性，并且对提供存储能力的矿工(framer)给予奖励。

2.Storj架构

2.1 Storj用户

Storj有两类用户：

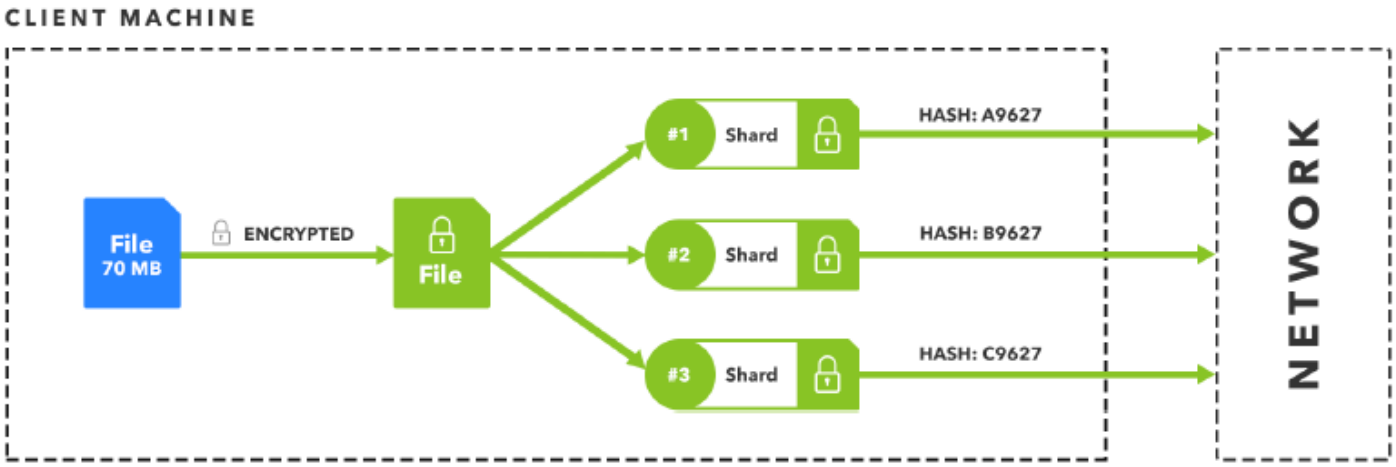
- Clients:存放文件，支付代币，同时文件拥有者的Client也是文件的审计者Verifier。
- Farmers:提供存储空间，获取代币。

2.2 Storj文件处理

- 分片Shards

  Storj确定一个加密后的分片作为存储在网络上的文件的一部分。能更好的保证数据的安全性，分片可以确保没有farmer拥有一个完整的文件副本。 定义标准分片大小为字节的倍数如8MB或者32MB。分片能更好的管理大文件，使大文件被分散到整个网络中。

一个文件加密后分片散发到网络的示意图如下：



- 冗余备份

  传统分布式存储备份方案:RAID,独立硬盘冗余阵列(Redundant Array of Independent Disks)

  Storj:EC,纠错码(Erasure coding）大部分区块链存储项目都是这样处理的。

  n-k纠错码技术:n份原始数据，增加k份校验数据，通过 n+k份数据中的任意n份数据来恢复原始数据。可容忍的最大失效的数据数量为k。

  例如，如果想容错10个盘，采用n+10模式。

  传统的 RAID6 允许两个盘失效，对应 EC就是 n+2模式。EC目前在分布式存储上的应用越来越广。

  如果想数据存储更加安全，可以采用增加k的方式，当然，这样做所付的费用也就相应的增加。

  假定每个碎片在线的概率为p，k-of-n纠删码技术失败的概率计算方法如下：

$$\Pr_{failure}(n;k,p) = \sum_{i=0}^{k-1} p^i (1-p)^{n-i} \binom{n}{i}$$

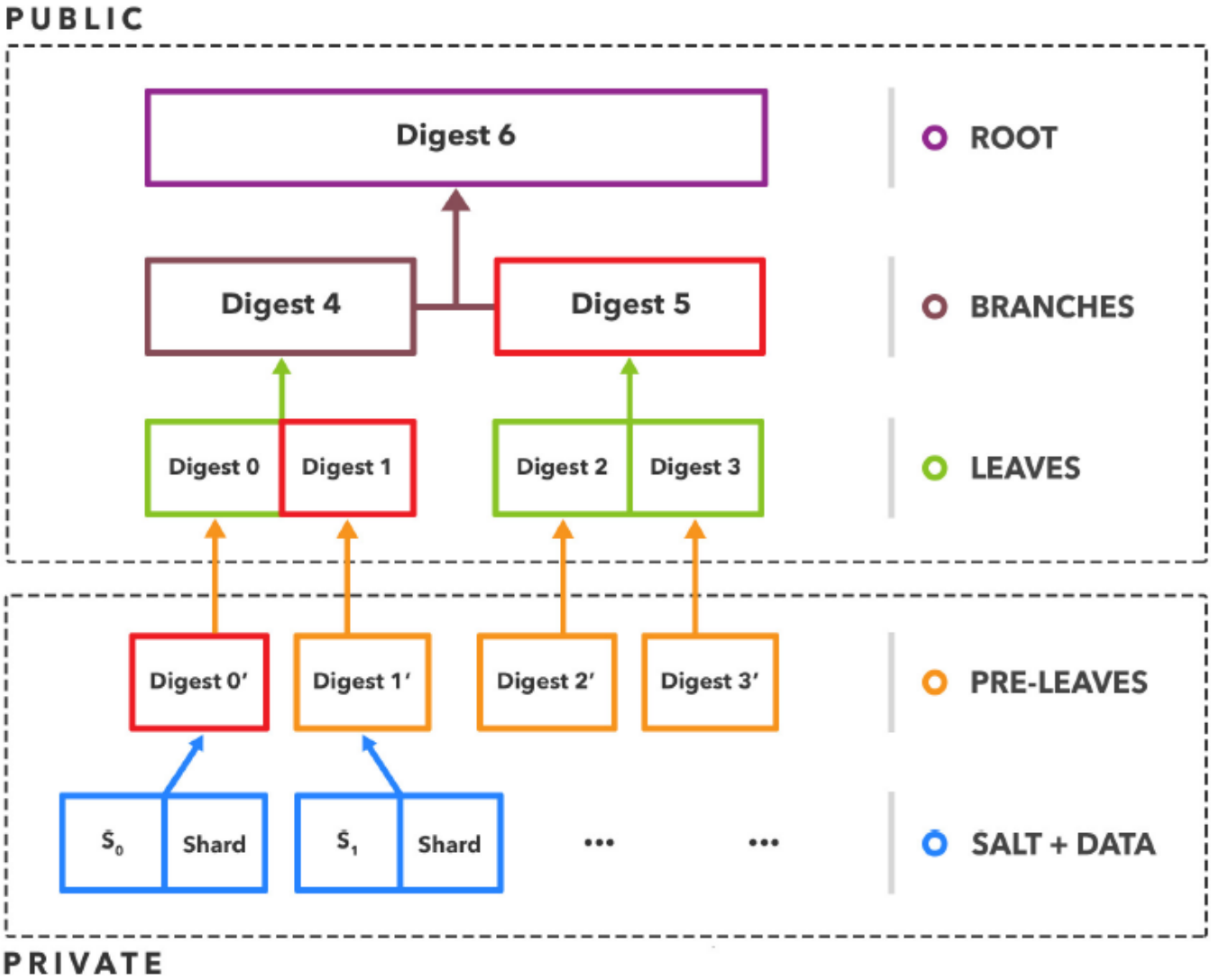
n	k	p	$\Pr_{failure}\, n,k,p$
18	6	0.5	4.812e-02
18	6	0.75	3.424e-05
18	6	0.9	5.266e-10
18	6	0.98	6.391e-19
36	12	0.5	1.440e-02
36	12	0.75	2.615e-08
36	12	0.9	1.977e-17
36	12	0.98	1.628e-34

2.3 Storj存储证明

Farmer 必须能够加密地证明他拥有这个碎片，并且没有以任何形式修改过。

- Proof-of-Retrievability

为了实现一个去信任的数据存储网络，Storj给clients提供一个审计方法， 证明clients存储在网络上的数据可用并且没有被修改。通过使用Merkle树和Merkle证明来做到这一点。 采集数据的集合并生成Merkle树，如下图所示，自下而上， clients(数据所有者)会生成一串随机数，随机数个数与文件分片shard个数相等，称为种子salts，用s表示，s(s0, s1, …… sn-1)。



Step1，将随机生成的salt和文件分片shard一一拼接。

Step2，将SALT+DATA进行哈希，得到merkle tree的准叶子结点PRE_LEAVES。

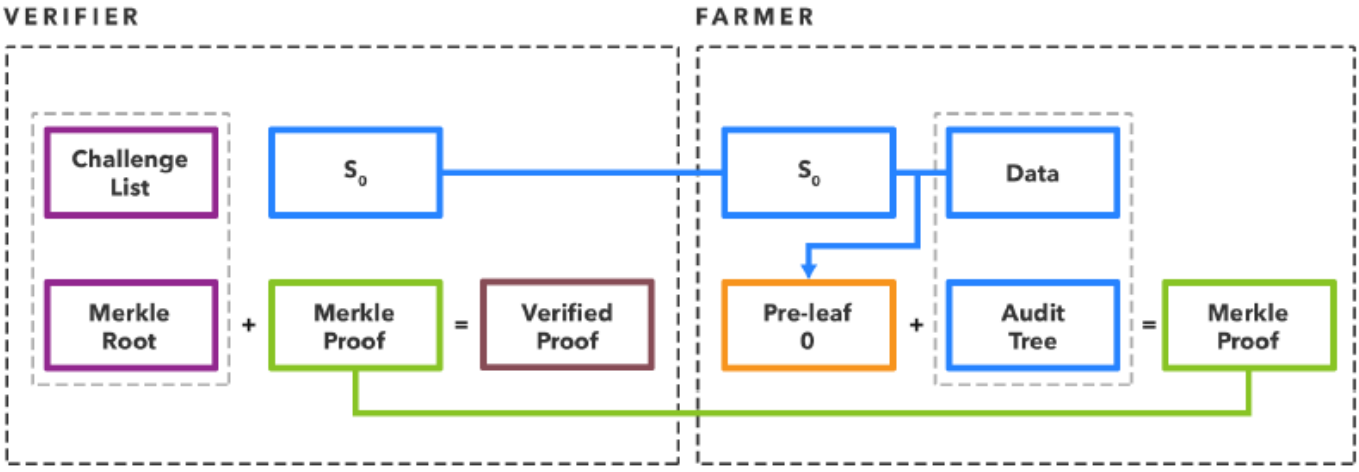
Step3，将pre-leaves再进行一次哈希得到merkle tree的叶子结点LEAVES。

Step4, 依次向上构建，得到merkle root。

构建出的merkle root 会作为metadata存储在transaction中，因此merkle root是公开且不可篡改的。

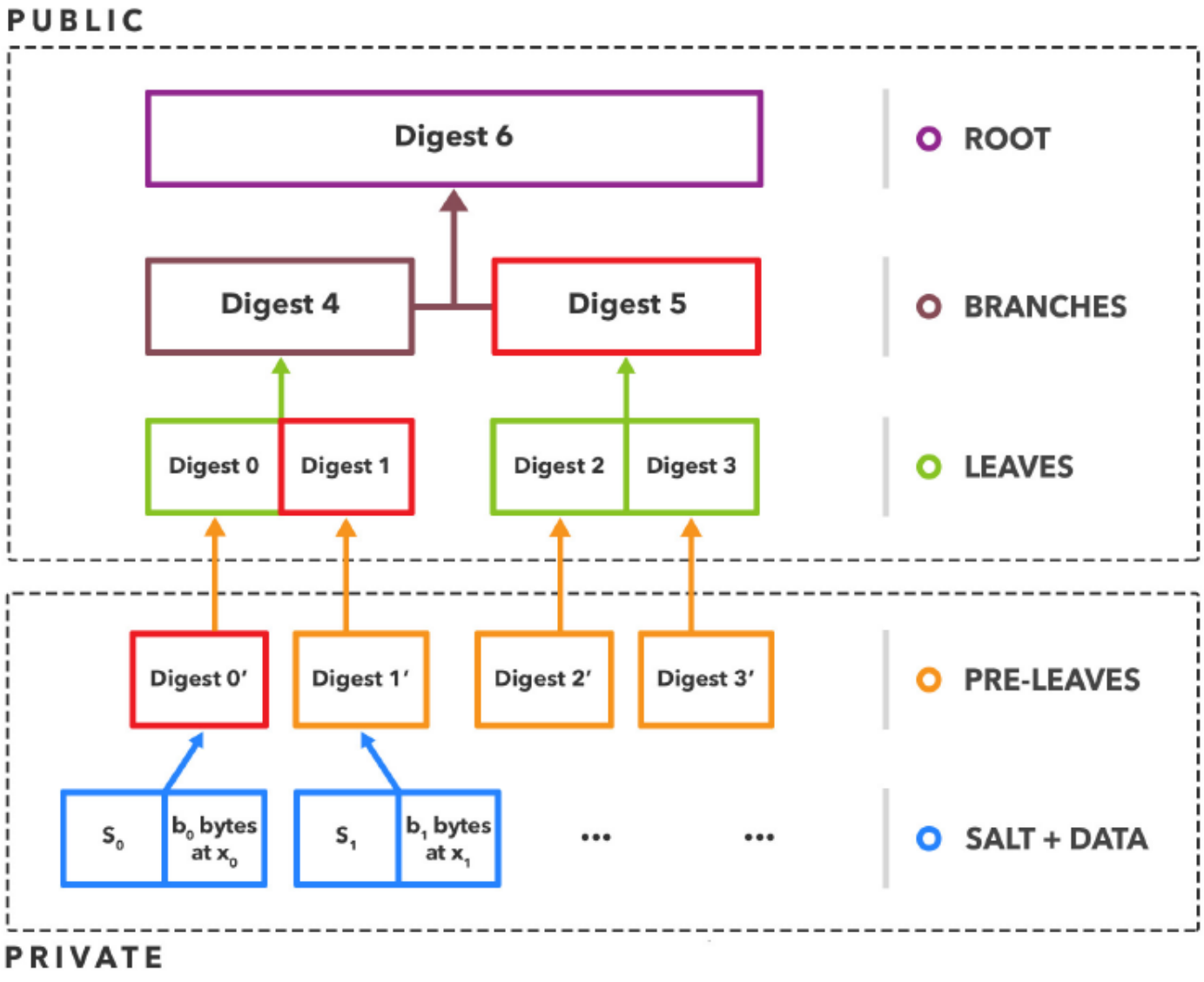
文件所有者Client，审计某个Farmer是否存储分片的流程: Step1,文件所有者Client,(审计者Verifier)，向Farmer发送某个种子salts。 Step2,Farmer计算并返回哈希值。 Step3,Client接收哈希值，并检查这个哈希值是否在merkle树中。 通过哈希挑战（质询算法）来做到这一点，在客户端产生一系列种子（由根种子确定），可以添加到文件并且进行哈希，产生唯一的哈希值。 这个过程称为心跳。

审计流程如下图：



- 改进 上述方法可以用种子和整个碎片产生哈希返回值。这验证了整个碎片的完整性，但是I/O效率特别差，并且导致查询时间长。

一种改进的方法是将分片分成N个离散的条，部分审计。审计流程如下：



Storj采用了3种不同的机制产生哈希挑战：

Full Heartbeat

用种子和整个分片产生哈希返回值，验证了整个分片的完整性。

Cycle Heartbeat

分片被分成N个离散的条，按照顺序验证每个小条直到完成周期。(Filecoin中首次提到) 这种方法效率好，并且允许验证整个分片的完整性，完成了N次心跳。不幸的是，它给系统留下了潜在的攻击可能，攻击者只需要有完整数据分片的1/N就能完成心跳。

Deterministic Heartbeat

使用客户端传过来的根种子读取分片时确定地产生心跳。使用Feistel序列，最多进行N+2√N次挑战就能验证数据。把纠删码技术加到心跳方案中，以致于分片任何微小的变化都能通过心跳检测到，或者通过分片检索进行恢复。这是最有效的方法，并且通过调整参数，能平衡I/O效率和验证文件完整性。

Mixed Methodology

每种方法都有自己的优点和缺点。Storj建议使用这些方法的组合。建议当文件第一次上传到网络时，使用一个完整的心跳。周期性心跳应当在定义的时间表上定期使用。确定性心跳应当在其他的挑战中使用。

3.Storj挖矿

Storj的用户从Storj处付费租用空间，矿工共享自己的磁盘获取代币SJCX。

Storj的共享硬盘代币每月支付一次。

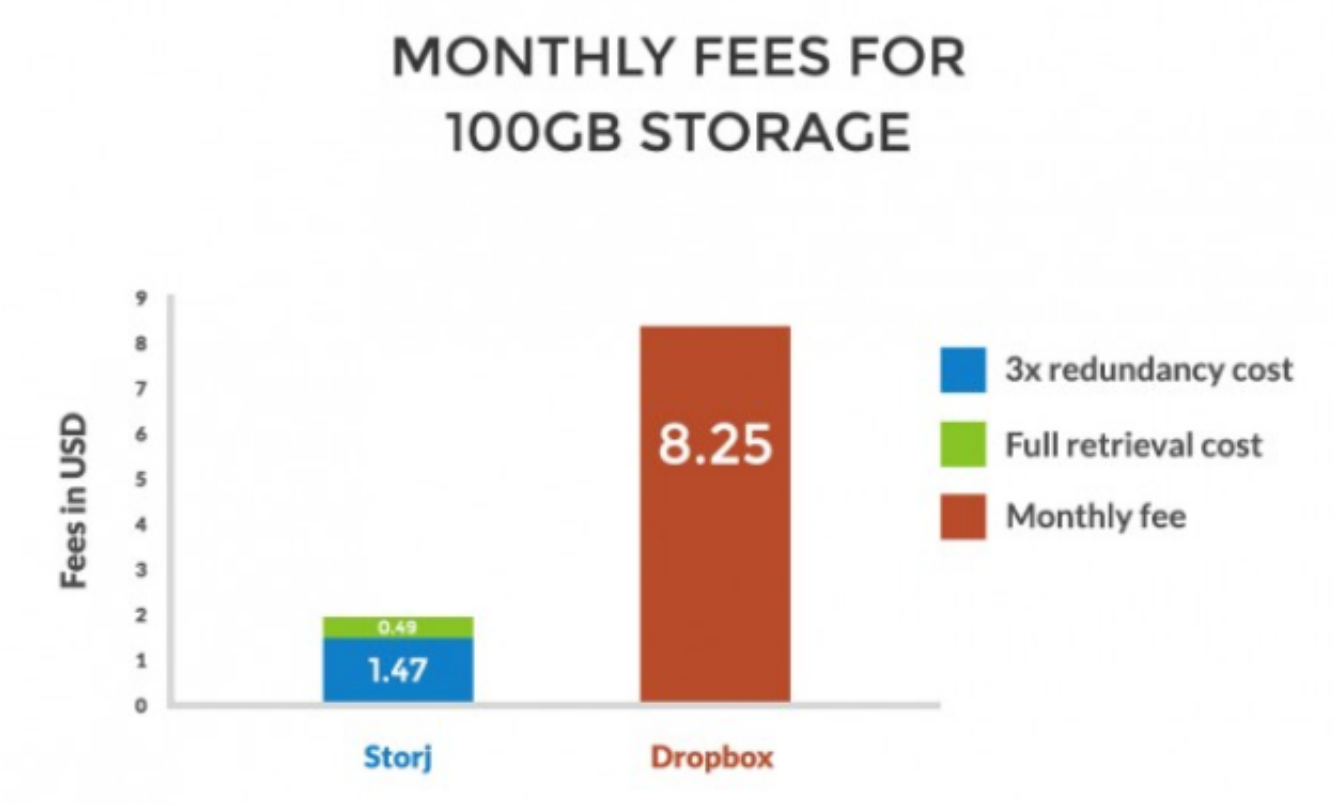
Storj的挖矿称为Drive Farming，利用用户提供的硬盘空间大小完成存储证明（与IPFS类似）。

影响挖矿的因素有：

- 共享的硬盘空间大小
- 上传和下载的速度

- Storj节点的可靠性和可用性（别老掉线）
- 空间的需求量

分布式存储的价格优势，Storj跟传统中心化分布式存储系统-多宝箱Dropbox的100G存储空间每月费用对比如下图：



Storj声称在自动网络上购买和出售硬盘空间将会极大地降低云计算的成本，去中心化存储成本只有中心化存储的1/100~1/10。

参考：

```
1. [https://storj.io/storj.pdf](Storj白皮书)
```