Filecoin 存储证明 浅析

胡飞瞳
游走在区块链技术和市场的边缘；IPFS和DWeb研究和推广者

概要

2. PoRep = PoS + PoR
3. 算法的设计目标是，复制证明必须在足够长的时间内才能完成，而时空证明要足够快，从而达到防止作弊的目的
4. 证明算法具体实现在 rust-fil-proofs 项目中，Filecoin实现通过CGo调用 Rust 相关函数

1. 为什么需要存储证明

信息产业发展至今，存储已经成为一个最为基础的产业。整个存储市场处于高速发展之中，基本上是每两年存储容量就翻一番。然而，作为信息的承载体，存储市场起步晚于网络市场，因此，其发展也较晚，当通用网络协议和实现都已经标准化的今天，存储市场却完全被巨头把控。在存储越来越重要的今天，通用的统一的存储网络呼声也就越来越高。

Filecoin就是致力于建立这样一个通用的人人可以参与的存储网络，在这个网络之中，没有寡头控制，也无需巨头背书。一个基本的目标就是，网络（代码）本身就可以形成一个良性的生态，奖优罚劣，形成一个高质量的网络。其首要任务就是防攻击（防欺骗）。在Filecoin的白皮书中提到了三种攻击方式：女巫攻击、生成攻击和外包攻击。对这些攻击的定义这里不展开，需要提到的是，所谓的防攻击，也就是要让欺骗者现行，让诚实者获得利益。

从存储的角度来理解，很简单，这个网络需要防止存储提供者作假，也就是要保证：1）不能虚报存储空间；2）不能虚报存储的数据或存储份数；3）不能虚报存储相应数据的时间周期。如果网络（代码）本身能够防止这些欺骗，那么，就不需要一个企业或机构来背书或仲裁。网络的生态形成了。

1. **Proof of Replication**: 复制证明，证明数据的的一个单独的拷贝已经在一个特定的扇区内创建成功。复制证明由**封印（****Seal****）**操作完成，封印操作创建一份数据的拷贝，并产生相应的**复制证明**
2. **Proof of Space-Time**：时空证明， 证明一定数量的已封印的扇区在一定的时间范围内存在于指定的存储空间之中 — 而不是证明者临时生成的数据（这被视为攻击）
3. **Piece Inclusion Proof**：数据片段包含证明，证明一个给定的数据片段存在于一个已封印的扇区之中
4. **Proof of Retrievability**：可检索证明，一种默克树证明，用来表明一个给定的叶节点存在于一个已封印的扇区内

Filecoin的时空证明是与复制证明算法紧密相关的。从实现的角度而言，实际上是一起实现的。协议实验室在前人研究的基础上进行了创新，组合了两种（图）算法，实现了更严格的证明和更高效的验证，这种算法就叫做** Zigzag，目前最高效，最严格的存储证明算法**。

> Zigzag: 当前最高效，最严格的存储证明算法
> 是两种（图）算法的组合 - 实现了更严格的证明和更高效的解封
> - Depth Robust Graph
> - Expander Graph
> 仅使用其中任何一种，都不能达到此目的

3. 存储证明和验证流程

原（当有检索的时候进行），挑战随机数生成（需要验证时进行），复制证明，复制验证。

理论上讲，就是对应以下7个步骤：

1. PoRep.Setup(λ,T) → pp ： 初始化阶段，每个节点执行，产生公共参数pp（每个节点一样），security parameter λ, time parameter T (挑战响应时间)
2. PoRep.Preproc(sk, D) → D˜, τD ： 预处理过程，节点自己完成。sk, 秘钥，D要处理的数据；产生，处理后的数据D˜和数据标识τD ： 数据标识需要发送给验证方，也即网络。
3. PoRep.Replicate(id, τD , D˜) → R, aux ： 产生复制数据。id为复制数据序号，在节点上执行，产生复制数据R，aux是附加信息，在做证明和验证的时候要用到
4. PoRep.Extract(pp, id, R) → D˜ ：从复制数据还原成原数据（预处理过的原数据）
5. PoRep.Prove(R, aux, id, r) → π ： 复制证明，输入分别为，复制数据及其附加信息，序号，加上挑战信息（一个由验证者产生的随机数），生成证明 π
6. PoRep.Poll(aux) → r ： 产生挑战信息，输入：复制数据的附加信息。
7. PoRep.Verify(id, τD, r, aux, π) → {0,1} ： 验证证明是否有效，输入包括：复制数据序号，原数据标识，挑战随机数，复制附加信息，验证信息。所有这些都需要传送给网络进行验证。验证只有通过或不通过两种状态。

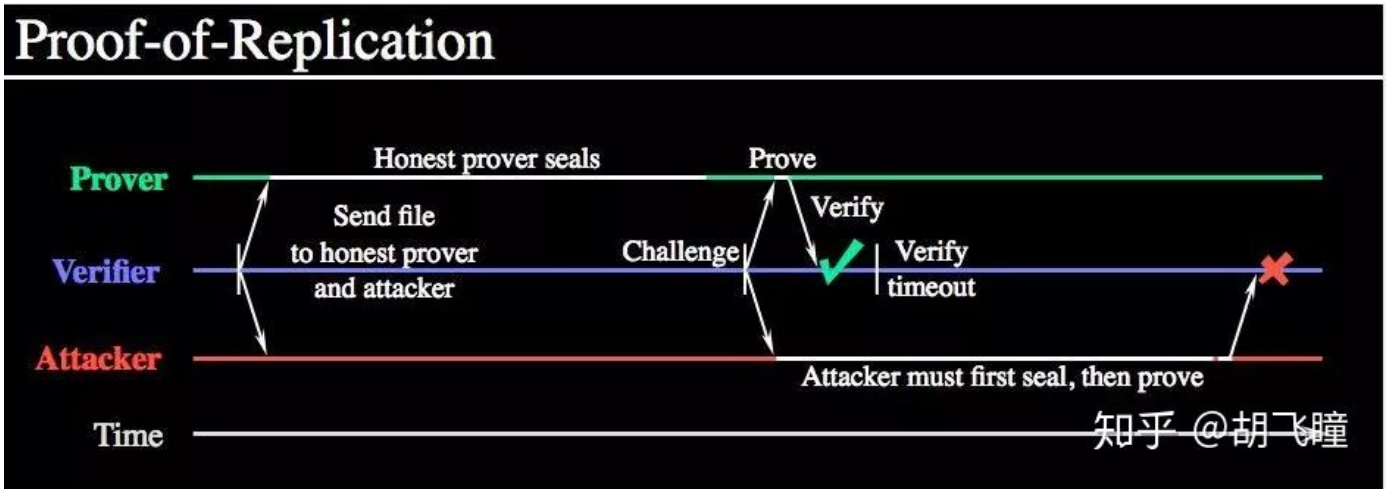
注意：这里似乎并没有提到时空证明，其实，时空证明包含着复制证明和验证之中。这是因为系统周期性地发送挑战信息给每个节点，而每个节点都需要证明在一定的期间之内能够完成数据存储的验证，通过持续的证明和验证，就完成了时空证明。

4. 如何防攻击

知乎 首发于 ProtoSchool协议学院

以大家都熟知的比特币为例，为防止双花（对网络的攻击），PoW提高了其难度，但不能抵御51%攻击。也就是说，机制的设计是让攻击者付出极高的成本，太不划算而不去做。

Filecoin的存储证明也是一样，没有绝对的安全，所设计的算法只是提高欺骗（攻击）的成本。那Filecoin存储证明的一个基本原则就是：让欺骗者付出比诚实者高得多的成本，那还不如做一个诚实节点，赚更多的钱。根据前文所述，欺骗者的一个主要手段就是在没有存储数据的情况谎报存储而获取利益。存储证明不能避免这种情况发生。但是整个协议的设计是：上节的7个步骤中的第3步（生成复制数据）比较复杂，需要较长的时间和运算，而第5步要求很快。那么当一个存储提供方，在没有保存第3步产生的数据的情况下，要证明自己（第5步），就必须临时做第3步，者需要消耗大量的计算，需要非常高的系统配置才来得及，否则就会超时被判存储证明失败。去大大提高系统的配置，还不如加一点存储老老实实做简单得多。



作弊者需要首先seal，如果在规定的时间完成，则需要极高的成本

配和管理方面强大得多。

rust-fil-proofs 代码请参见：github.com/filecoin-pro.... 其与Filecoin证明相关的 API 定义在 filecoin-proofs/src/api/ 目录之下。其中：

> 验证相关的接口，就定义在 filecoin-proofs/src/api/ 目录中；而与复制证明相关的接口，则定义在 filecoin-proofs/src/api/sector_builder/helper/ 目录中，这些接口包括：
> - add_piece: 往一个sector 里面填充内容
> - seal: 封印，即通过密码运算产生一个可证明的复制，当产生之后，原数据可删除
> - retrieve_piece: 获取用户数据，包括unseal

rust-fil-proofs 是一个独立的项目，Filecoin 的Go语言实现（目前唯一的一种实现）通过 CGo 来调用 rust-fil-proofs函数。其中接口定义在 github.com/filecoin-pro...;

更多信息，参见rust-fil-proofs Readme文件，以下列出了与Filecoin API相关部分。

- Sector Base API: …/rust-fil-proofs/target/doc/sector_base/api/index.html
- Filecoin Proofs API: …/rust-fil-proofs/target/doc/filecoin_proofs/api/index.html
- filecoin-proofs 扇区构建相关API的Go语言实现
  -- 相关的接口结构定义
- filecoin-proofs 验证相关 API的 Go语言实现
  -- 相关的接口结果定义

- Tight Proofs of Space and Replication , Ben Fisch
- github.com/filecoin-pro...
- github.com/filecoin-pro...

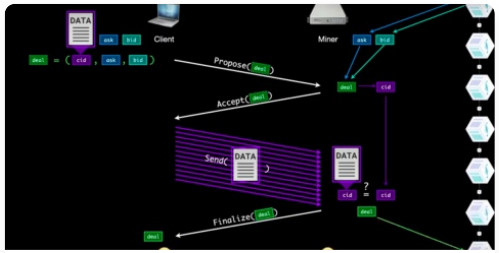
发布于 2019-03-18

Filecoin IPFS 区块链(Blockchain)

知乎 首发于 ProtoSchool协议学院

ProtoSchool协议学院 进入专栏

推荐阅读



Filecoin 存储封印和证明初步解析

胡飞瞳 发表于Proto...

hyperledger fabric 代码分析之 gossip 协议

gossip功能：1、管理组织内节点和通道信息，以及检测哪些节点是否在线活着离线2、广播数据使组织内相同channel的节点同步到相同的数据3、管理新加入入节点，并同步数据到新的节点 其它问题...

同甫陈

IPFS:Filecoin和复制证明

飞向未来(... 发表于IPFS指...

Filecoin Discover后落地的主网又进一步

怪盗KID...

还没有评论

写下你的评论...

▲ 赞同 1 ▼ 添加评论 分享 收藏 ...