The Filecoin Protocol:

protocol overview, the important results, and new open problems
Filmosia: A Decentralized Storage Network

Prepared by

The Context

Open Problems

Engineering

The Protocol

Questions
Open Problems

The Context

Engineering

The Protocol

Questions
honest - follows the protocol
malicious - tries to cheat, break, attack
rational - follows or defects as convenient
The Mission of Filecoin

to create a decentralized, efficient, and robust foundation for humanity’s information.
Filecoin: A Decentralized Storage Network
Peyman Lale
August 11, 2017

Abstract

The Internet is the middle of a revolution, where file storage is no longer in the hands of a few. Filecoin aims to solve this problem by providing a distributed storage network that allows anyone to store their files. The protocol is designed to incentivize nodes to store data by offering rewards for doing so. The Filecoin network is built on top of the IPFS file system, which provides a scalable and decentralized way to store and retrieve files. The goal is to create a more resilient and usable Internet by leveraging the distributed nature of the blockchain.

This work:
(1) Introduces the Filecoin Network, giving an overview of the protocol, and uses smart contracts to store.
(2) Describes the filecoin storage network.
(3) Analyzes the economics and storage cost of file storage.
(4) Evaluates the feasibility of using Filecoin.

Further work:
(1) Compare the economics of storing data using Filecoin.
(2) Evaluate the performance of the Filecoin network.

Note: Filecoin is a work in progress. The concepts and implementation are subject to change.
IPFS: Distributed Web Protocol

IPLD: authenticated data model & formats

libp2p: modular p2p networking library

Filecoin: Decentralized Storage Network
Filecoin: Decentralized Storage Network

IPFS: Distributed Web Protocol

IPLD: authenticated data model & formats

libp2p: modular p2p networking library
Filecoin

- **Useful Proof-of-Work Consensus**
  - Through useful storage

- **Optimize world’s storage**
  - Put all latent storage to work
  - Minimize bandwidth waste
  - Automatic balancing

- **Decentralize world’s storage**
  - Content Addressed (IPFS)
  - Stop relying on <10 providers
• MUST be a market
• MUST allow price undercutting
• MUST scale to ZB and beyond
• MUST provide all tiers of service
• MUST have excellent performance
• MUST obviate business frontend
Requirements: Security, Scalability, Usability

- Secure
  - BFT safety
  - BAR/IC-BFT safety

- Provide a highly reliable service
  - Collateral commitments
  - Rational SLA

- Fast & easy to PUT
  - Easy to understand
  - 1-few block times
  - Don’t melt miners

- Fast & easy to GET
  - Easy to understand
  - sub-second (for frequent reqs)
  - well specified in Filecoin

- Be Web Scale
  - 2017: 10 EB
  - 2018: 100 EB
  - 2019: 1 ZB
  - 2020: 2 ZB

- Variable size inputs
  - Allow 1KB or 1PB PUTs

- Keep replication overhead low
  - Use IDA / erasure codes

- Mining possible by regular conns
  - Per-miner/user bandwidth <= 10 Mb/s

- Don’t melt the 20-teens internet
  - Total control bandwidth <= 10 Tb/s
Requirements: Systems, Ecosystem, Markets

- Don’t waste resources
  - Don’t do wasteful hashing
  - Don’t do wasteful Proofs-of-space
  - Keep storage overhead to a minimum
  - Keep bandwidth overhead to a minimum

- Consensus
  - Build consensus within
  - Avoid punting consensus
  - Avoid vectors of attack
  -Avoid inter-chain dependence

- Markets
  - Drive storage pricing down
  - Promote pricing competition
  - Races and pricing bloodbaths

- Macro Miners
  - Reward massive storage arrays
  - Easy to on-board massive miners
  - Easy to unite against top 3

- Micro Miners
  - Reward well-placed nodes
  - Reward low latency & high bw

- Ethereum
  - Native connection to Ethereum
  - Allow Eth contracts to use Filecoin

- Formal Verification
  - Verified protocol
  - Verified implementation
  - Verified contracts
The Feynman Algorithm

1. Write down the problem
2. Think hard
3. Write down the solution
The Feynman Algorithm

1. Write down the problem
2. Think hard
3. Write down the solution
The Context

Engineering

The Protocol

Open Problems

Questions
Conceptual Prerequisites:

- Cryptographic Hash Functions
- Digital Signatures
- Verifiable Computation
  - SNARKs
- Byzantine Consensus (BFT)
- Blockchains
- Sequential Computation
- Bitcoin
- Ethereum
Filecoin: A Decentralized Storage Network

Protocol Labs
August 14, 2017

Abstract

The internet is in the middle of a revolution: centralized proprietary services are being replaced with decentralized open ones; trusted parties replaced with verifiable computation; brittle location addresses replaced with resilient content addresses; inefficient monolithic services replaced with peer-to-peer algorithms背后的哈希值。

Filecoin is a decentralized storage network that turns cloud storage into an algorithmic market. The network runs on a blockchain with a native protocol token (also called "Filecoin"), which miners earn by providing storage to clients. Conversely, clients spend Filecoin to store, retrieve, or distribute data. As with Bitcoin, Filecoin miners compete to mine blocks with similar rewards, but Filecoin mining power is proportional to active storage, which directly provides a useful service to clients (unlike Bitcoin mining, whose usefulness is limited to maintaining blockchain consensus). This creates a powerful incentive for miners to store as much storage as they can, and rent it out to clients. The protocol ensures these unused resources into a self-healing storage network that nobody in the world can rely on. The network achieves robustness by replicating and dispersing content, while automatically detecting and repairing replica failures. Clients can select replication parameters to protect against different threat models. The protocol's cloud storage network also provides security, as content is encrypted end-to-end at the client, while storage providers do not have access to decryption keys. Filecoin works as an incentive layer on top of IPFS [1], which can provide storage infrastructure for any data. It is especially useful for decentralizing data, building and running distributed applications, and implementing smart contracts.

This work:
(a) Introduces the Filecoin Network, gives an overview of the protocol, and walks through several components in detail.
(b) Formulates decentralized storage network (DSN) science and their properties, then constructs Filecoin as a DSN.
(c) Introduces a novel class of proof-of-storage schemes called proof-of-replication, which allows proving that any replica of data is stored in physically independent storage.
(d) Introduces a novel self-healing consensus based on sequential proof-of-replication and storage as a measure of power.
(e) Formulates verifiable markets and constructs two markets, a Storage Market and a Retrieval Market, which govern how data is written to and read from Filecoin, respectively.
(f) Discusses use cases, connections to other systems, and how to use the protocol.

Note: Filecoin is a work in progress. Active research is underway, and new versions of this paper will appear at http://filecoin.io. For comments and suggestions, contact us at research@filecoin.io.
## Contents

1 **Introduction**  
   1.1 Elementary Components  
   1.2 Protocol Overview  
   1.3 Paper organization  

2 **Definition of a Decentralized Storage Network**  
   2.1 Fault tolerance  
   2.2 Properties  

3 **Proof-of-Replication and Proof-of-Spacetime**  
   3.1 Motivation  
   3.2 Proof-of-Replication  
   3.3 Proof-of-Spacetime  
   3.4 Practical PoRep and PoSt  
   3.5 Usage in Filecoin  

4 **Filecoin: a DSN Construction**  
   4.1 Setting  
   4.2 Data Structures  
   4.3 Protocol  
   4.4 Guarantees and Requirements  

5 **Filecoin Storage and Retrieval Markets**  
   5.1 Verifiable Markets  
   5.2 Storage Market  
   5.3 Retrieval Market  

6 **Useful Work Consensus**  
   6.1 Motivation  
   6.2 Filecoin Consensus  

7 **Smart Contracts**  
   7.1 Contracts in Filecoin  
   7.2 Integration with other systems  

8 **Future Work**  
   8.1 Ongoing Work  
   8.2 Open Questions  
   8.3 Proofs and Formal Verification  


## Contents

1 **Introduction**  
   1.1 Elementary Components  ................................................. 4  
   1.2 Protocol Overview .................................................... 4  
   1.3 Paper organization .................................................. 4  

2 **Definition of a Decentralized Storage Network**  
   2.1 Fault tolerance ...................................................... 8  
   2.2 Properties .............................................................. 8  

3 **Proof-of-Replication and Proof-of-Spacetime**  
   3.1 Motivation .............................................................. 10  
   3.2 Proof-of-Replication .................................................. 10  
   3.3 Proof-of-Spacetime .................................................... 11  
   3.4 Practical PoRep and PoSt .............................................. 11  
   3.5 Usage in Filecoin .................................................... 14  

4 **Filecoin: a DSN Construction**  
   4.1 Setting ................................................................. 16  
   4.2 Data Structures ........................................................ 17  
   4.3 Protocol ............................................................... 17  
   4.4 Guarantees and Requirements ........................................ 21  

5 **Filecoin Storage and Retrieval Markets**  
   5.1 Verifiable Markets ..................................................... 24  
   5.2 Storage Market ........................................................ 24  
   5.3 Retrieval Market ...................................................... 27  

6 **Useful Work Consensus**  
   6.1 Motivation .............................................................. 30  
   6.2 Filecoin Consensus .................................................... 30  

7 **Smart Contracts**  
   7.1 Contracts in Filecoin .................................................. 33  
   7.2 Integration with other systems ...................................... 33  

8 **Future Work**  
   8.1 On-going Work ......................................................... 34  
   8.2 Open Questions ....................................................... 34  
   8.3 Proofs and Formal Verification ..................................... 35
Novel Components

Decentralized Storage Network (DSN)

Useful formalization of this kind of protocols

Verifiable Markets
Storage & Retrieval

Structures to create competition and optimization

Proofs-of-Replication (PoRep)
&
Proofs-of-Spacetime (PoSt)

Novel classes of proofs that verify storage

Useful Proof of Work for Consensus

How to use useful storage for a PoW consensus
**Cast of Characters**

**Client:** hires the Network to store data. pays for storage with Filecoin

**Miner:** stores data for the network, and its clients. collects Filecoin as a reward

**Network:** organizes all work, verifies and repairs storage. rewards miners with Filecoin
The Network acts as an intermediary.
The Network acts as an intermediary.
In reality, the network is a virtual construction, implemented by the users.
In reality, the network is a virtual construction, implemented by the users.
The network checks miners are storing data over time
The network checks miners are storing data over time.
The network can tolerate <1/2 failures
Miners and Clients could be \{Honest, Rational, or Malicious\}
The network must behave correctly regardless (up to a fraction)
Miners and Clients could be {Honest, Rational, or Malicious}

The network must behave correctly regardless (up to a fraction)

The network must verify all work is being done correctly, must reward correct behavior, and must punish malicious behavior.
The protocol uses a **blockchain** to record and agree on:

- **Currency** - account balances, payments, collaterals, etc
- **Market Orders** - asks, bids, and deals on the markets
- **Allocations** - which miners store which pieces
- **Proofs** - a verifiable record of correct behaviors
- **Contracts** - state and execution of smart contracts
A Miner’s **power** in the blockchain consensus is based on the **useful storage** they provide to the network.
Clients hire Miners through markets, where pricing can be optimized.

Miners place Asks. Clients place Bids. These are matched to form Deals.
The **Storage Market** handles storing data (PUT).
The **Retrieval Market** handles getting data (GET).

Both markets can happen off-chain.
This is especially important for sub-second latency retrieval.
<table>
<thead>
<tr>
<th>Client</th>
<th>Network</th>
<th>Miner</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>SendPieces(P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddOrder(deal)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td></td>
</tr>
<tr>
<td>RecvPieces(P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddOrder(deal)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AssignOrders(OB)</td>
<td>PledgeSector(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SealSector(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProveSector(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RepairOrders(OB)</td>
</tr>
<tr>
<td></td>
<td>CheckSectors(S)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td></td>
<td>AddOrder(ask)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FindPieces(P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AddOrder(deal)</td>
</tr>
</tbody>
</table>
# Contents

1 **Introduction**  
1.1 Elementary Components ........................................... 4  
1.2 Protocol Overview .................................................. 4  
1.3 Paper organization .................................................. 4  

2 **Definition of a Decentralized Storage Network** ............... 8  
2.1 Fault tolerance ..................................................... 8  
2.2 Properties .......................................................... 8  

3 **Proof-of-Replication and Proof-of-Spacetime** ................. 10  
3.1 Motivation .......................................................... 10  
3.2 Proof-of-Replication ............................................... 10  
3.3 Proof-of-Spacetime ................................................ 11  
3.4 Practical PoRep and PoSt ......................................... 11  
3.5 Usage in Filecoin .................................................. 14  

4 **Filecoin: a DSN Construction** .................................. 16  
4.1 Setting ............................................................... 16  
4.2 Data Structures ................................................... 17  
4.3 Protocol ............................................................. 17  
4.4 Guarantees and Requirements .................................... 21  

5 **Filecoin Storage and Retrieval Markets** ....................... 24  
5.1 Verifiable Markets ................................................ 24  
5.2 Storage Market .................................................... 24  
5.3 Retrieval Market .................................................. 27  

6 **Useful Work Consensus** .......................................... 30  
6.1 Motivation .......................................................... 30  
6.2 Filecoin Consensus ................................................ 30  

7 **Smart Contracts** .................................................. 33  
7.1 Contracts in Filecoin .............................................. 33  
7.2 Integration with other systems ................................... 33  

8 **Future Work** ....................................................... 34  
8.1 Ongoing Work ...................................................... 34  
8.2 Open Questions .................................................... 34  
8.3 Proofs and Formal Verification .................................. 35
2 Definition of a Decentralized Storage Network

We introduce the notion of a Decentralized Storage Network (DSN) scheme. DSNs aggregate storage offered by multiple independent storage providers and self-coordinate to provide data storage and data retrieval to clients. Coordination is decentralized and does not require trusted parties: the secure operation of these systems is achieved through protocols that coordinate and verify operations carried out by individual parties. DSNs can employ different strategies for coordination, including Byzantine Agreement, gossip protocols, or CRDTs, depending on the requirements of the system. Later, in Section 4, we provide a construction for the Filecoin DSN.

Definition 2.1. A DSN scheme II is a tuple of protocols run by storage providers and clients:

\[ (\text{Put, Get, Manage}) \]

- Put(data) \rightarrow key: Clients execute the Put protocol to store data under a unique identifier key.
- Get(key) \rightarrow data: Clients execute the Get protocol to retrieve data that is currently stored using key.
- Manage(): The network of participants coordinates via the Manage protocol to: control the available storage, audit the service offered by providers and repair possible faults. The Manage protocol is run by storage providers often in conjunction with clients or a network of auditors.

A DSN scheme II must guarantee data integrity and retrievability as well as tolerate management and storage faults defined in the following sections.
Decentralized Storage Network (DSN)

**DSN Interface**

type DSN interface {
    Put(Data) Key
    Get(Key) Data
    Manage()
}
Decentralized Storage Network (DSN)

**DSN Properties**

- **Complete** if - client in all-honest network can **Put** data to miners and **Get** it back
- **Secure** if:
  - Guarantees *data integrity* - A cannot convince clients to accept altered data
  - Guarantees *retrievability* - given FT assumptions, clients can **Put** data and eventually **Get** it back.

- Tolerates *management faults* - byzantine faults in Manage
  - 
    - 
      - $(f, n)$-tolerant if network has $n$ nodes and can tolerate up to $f$ faults.

- Tolerates *storage faults* - byzantine faults in Put & Get
  - 
    - 
      - $(f, m)$-tolerant if Put stores in $m$ independent storage providers and can tolerate $f$ faults ($m <= n$)
In addition, a DSN MAY:

- Be *publicly verifiable* - if it produces publicly verifiable proofs that the data is stored.
- Be *auditible* - if it generates a verifiable trace of operation that can be checked in the future to confirm storage and storage duration correctness.
- Be *incentive-compatible* - if it rewards and punishes for storing or failing to store data such that a dominant strategy stores data correctly.
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Novel classes of proofs that verify storage

- Needed to solve a fundamental problem to get Filecoin to work well.

- This yielded a new class of **Proofs-of-Storage: Proofs-of-Replication**.

- Also yielded a way to aggregate **Proofs-of-Space** over a period of time: **Proofs-of-Spacetime**.

- We extracted this into its own work
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Novel classes of proofs that verify storage

**Contributions:**

- Classifying the types of **Proofs-of-Storage**
- Concept & a defn of **Proofs-of-Replication**
  - Time-bounded PoReps
  - Construction using CBC (v. expensive)
- Concept & a defn of **Proofs-of-Spacetime**
  - Construction using proof-chains
  - Compact construction using SNARKs
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Novel classes of proofs that verify storage

Proof of Replication
Technical report

Proof of Replication
new work
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Novel classes of proofs that verify storage

- Work w/ Ben Fisch, Nicola Greco
- Ben came up with a much better construction, using Depth Robust Graphs
- Much better definition of Proofs-of-Replication.
- See next talk!
Novel classes of proofs that verify storage
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

A proving scheme where a prover $P$ can prove to a verifier $V$ that $P$ is indeed storing data $D$. 
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover

Setup (data)

Verifier

Proofs-of-Storage

data
Prover Verifier

Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover Setup (data)

data

Verifier
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover

Verifier

Setup (data)

data

challenge
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

data

Setup (data)

challenge

proof ← Prove(challenge, data)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

Setup \(\text{(data)}\)

\[\text{data} \quad \text{←} \quad \text{Prove}(\text{challenge, data})\]

proof
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover

Verifier

Setup (data)

data

challenge

proof ← Prove(challenge, data)

proof

{0, 1} ← Verify(challenge, proof)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover

Verifier

Setup (data)

proof ← Prove(challenge, data)

proof

{0,1} ← Verify(challenge, proof)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Retrievability**

Prover

Verifier

Setup (data)

data

proof ← Prove(challenge, data)

proof

{0, 1} ← Verify(challenge, proof)

challenge

Can retrieve data from the proofs.
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

What if we want P to store 2 copies of data?
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

What if we want P to store 2 copies of data?

P could pretend to store 2 copies, but only store 1.
What if we want P to store 2 copies of data?

P could pretend to store 2 copies, but only store 1. Simple fixes: verifier pre-encrypts different copies. But these preclude prover from repairing copies...
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

What if we want P1 ... Pn to store 1 copy of data each?
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

What if we want P1 ... Pn to store 1 copy of data each?

P1 ... Pn may be sybil identities or a coalition. They could pretend to store 1 copy each, but deduplicate to store only 1. (Sybil Attack)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Prover

Verifier

Proofs-of-Storage

What if P could choose or determine data?
Proofs-of-Replication (PoRep)
&
Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

What if P could choose or determine data?

P could choose data such that P can generate data on demand. P then pretends to store data, but re-generates it just-in-time to Prove challenges.

(Generation Attack)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proofs-of-Storage

Prover

Verifier

What if P could choose or determine data?

This is very easy to do in the DSN setting. Solutions until now (Filecoin v1) aligned incentives, and had to avoid participation rewards to ensure the Generation Attack would be irrational.
Proof-of-Replication

A proof that a prover is dedicating unique physical resources to storing each copy of data, even if they can generate it on the fly.
Proof-of-Replication

A proving scheme where a prover $P$ can prove to a verifier $V$ that $P$ is indeed storing a particular replica $R^D$ of data $D$. No two replicas $R^D_i$ and $R^D_k$ can be deduplicated into the same physical storage.
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Replication

Prover

Verifier

Setup \((\text{data, ek})\)
Proof-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Replication

Prover

Verifier

replica ← Encode(data, ek)

Setup (data, ek)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Replication**

**Prover**

\[ \text{replica} \leftarrow \text{Encode}(\text{data}, \text{ek}) \]

**Verifier**

Setup \((\text{data}, \text{ek})\)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Replication**

Prover

Verifier

Setup \((\text{data, ek})\)

\(\text{replica} \leftarrow \text{Encode}(\text{data, ek})\)

challenge
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Replication**

Prover

\[ \text{replica} \leftarrow \text{Encode}(\text{data}, \text{ek}) \]

Verifier

\[ \text{setup} (\text{data}, \text{ek}) \]

\[ \text{challenge} \leftarrow \text{Prove}(\text{challenge}, \text{replica}) \]

\[ \text{proof} \leftarrow \text{Prover} \]
Proofs-of-Replication (PoRep) &
Proofs-of-Spacetime (PoSt)

**Proof-of-Replication**

Prover

\[ \text{replica} \leftarrow \text{Encode}(\text{data}, \text{ek}) \]

Verifier

\[ \text{Setup} (\text{data}, \text{ek}) \]

\[ \text{challenge} \]

\[ \text{proof} \leftarrow \text{Prove}(\text{challenge}, \text{replica}) \]

\[ \{0, 1\} \leftarrow \text{Verify}(\text{challenge}, \text{proof}) \]
Proof-of-Replication

\[ r_1 \leftarrow \text{Encode}(\text{data}, e_{k1}) \]
\[ r_2 \leftarrow \text{Encode}(\text{data}, e_{k2}) \]

\[ p_1 \leftarrow \text{Prove}(c_1, r_1) \]
\[ p_2 \leftarrow \text{Prove}(c_2, r_2) \]

\[ \{0,1\} \leftarrow \text{Verify}(c_1, p_1) \]
\[ \{0,1\} \leftarrow \text{Verify}(c_2, p_2) \]
Proofs-of-Replication (PoRep) &
Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier

How do we reduce *frequency of verifications*?

Proofs-of-Storage require the verifier $V$ to do frequent polling to ensure the prover $P$ does indeed continue to store data $D$. 
Proof-of-Spacetime

How do we reduce frequency of verifications?

Intuitions: We can aggregate proofs and present them all at once. We can use chaining to ensure proofs were produced sequentially, covering a span of time.
Proof-of-Spacetime

A proving scheme that allows a prover $P$ to aggregate **Proofs-of-Space** (or **Proofs-of-Storage**) over time into an auditable record $\pi$ that **proves** to a verifier $V$ that $P$ was indeed wasting **space** $S$ (or storing **data** $D$) for a certain duration of time $T$. 
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier

Setup \((\text{data}, \text{r0})\)

data

c_0 \leftarrow \text{NextChallenge}(\text{r0})

p_0 \leftarrow \text{Prove}(c_0, \text{data})
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Spacetime**

**Prover**

\[ \text{Setup} \left( \text{data, } r_0 \right) \]

\[ \text{data} \]

\[ c_0 \leftarrow \text{NextChallenge}(r_0) \]

\[ p_0 \leftarrow \text{Prove}(c_0, \text{data}) \]

\[ c_1 \leftarrow \text{NextChallenge}(p_0) \]

\[ p_1 \leftarrow \text{Prove}(c_1, \text{data}) \]

**Verifier**

Setup (data, r0)
Proof-of-Replication (PoRep) & Proof-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

data

\[ c_0 \leftarrow \text{NextChallenge}(r_0) \]
\[ p_0 \leftarrow \text{Prove}(c_0, \text{data}) \]
\[ c_1 \leftarrow \text{NextChallenge}(p_0) \]
\[ p_1 \leftarrow \text{Prove}(c_1, \text{data}) \]
\[ c_2 \leftarrow \text{NextChallenge}(p_1) \]
\[ p_2 \leftarrow \text{Prove}(c_2, \text{data}) \]

Verifier

Setup \((\text{data, } r_0)\)
Proofs-of-Replication (PoRep) &
Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

data
c0 ← NextChallenge(r0)
p0 ← Prove(c0, data)
c1 ← NextChallenge(p0)
p1 ← Prove(c1, data)
c2 ← NextChallenge(p1)
p2 ← Prove(c2, data)

Verifier

Setup (data, r0)

re-challenge r1
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

data
c0 ← NextChallenge(r0)
p0 ← Prove(c0, data)
c1 ← NextChallenge(p0)
p1 ← Prove(c1, data)
c2 ← NextChallenge(p1)
p2 ← Prove(c2, data)
c3 ← NextChallenge(p2, r1)
p3 ← Prove(c3, data)

Verifier

Setup (data, r0)
poll & re-challenge r1
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

**Proof-of-Spacetime**

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup ((data, r_0))</td>
<td>((p_0, c_0), \ldots, (p_3, c_3))</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
data \\
c_0 &\leftarrow \text{NextChallenge}(r_0) \\
p_0 &\leftarrow \text{Prove}(c_0, data) \\
c_1 &\leftarrow \text{NextChallenge}(p_0) \\
p_1 &\leftarrow \text{Prove}(c_1, data) \\
c_2 &\leftarrow \text{NextChallenge}(p_1) \\
p_2 &\leftarrow \text{Prove}(c_2, data) \\
c_3 &\leftarrow \text{NextChallenge}(p_2, r_1) \\
p_3 &\leftarrow \text{Prove}(c_3, data)
\end{align*}
\]
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Setup \((\text{data}, r_0)\)

data

\(c_0 \leftarrow \text{NextChallenge}(r_0)\)
\(p_0 \leftarrow \text{Prove}(c_0, \text{data})\)
\(c_1 \leftarrow \text{NextChallenge}(p_0)\)
\(p_1 \leftarrow \text{Prove}(c_1, \text{data})\)
\(c_2 \leftarrow \text{NextChallenge}(p_1)\)
\(p_2 \leftarrow \text{Prove}(c_2, \text{data})\)
\(c_3 \leftarrow \text{NextChallenge}(p_2, r_1)\)
\(p_3 \leftarrow \text{Prove}(c_3, \text{data})\)
\(c_4 \leftarrow \text{NextChallenge}(p_3)\)
\(p_4 \leftarrow \text{Prove}(c_4, \text{data})\)
\(c_5 \leftarrow \text{NextChallenge}(p_4)\)
\(p_5 \leftarrow \text{Prove}(c_5, \text{data})\)

\(\text{poll} \ & \text{re-challenge} \ r_1\)

\((p_0, c_0), \ldots, (p_3, c_3)\)

Verifier

\(\{0,1\} \leftarrow \text{Verify}(c_0 \ldots c_3, p_0 \ldots p_3)\)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier

Setup \((\text{data, } r)\)

produce auditable proof chain

Verify proof chain
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier

Setup \((\text{data}, \mathbf{r})\)

Produce auditable proof chain

\[
\begin{array}{c|c}
  c0 & p0 \\
  c1 & p1 \\
  c2 & p2 \\
  c3 & p3 \\
  \vdots \\
  cn & pn \\
\end{array}
\]

Verify proof chain

\(c0-n\) \(p0-n\)
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

<table>
<thead>
<tr>
<th>c0</th>
<th>p0</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p1</td>
</tr>
<tr>
<td>c2</td>
<td>p2</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

| cn | pn |

Poll & add randomness to sequential comp.

Verifier

Setup (data, r1)

Poll (r2)

Verify proof chain

produce auditable proof chain
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)

Proof-of-Spacetime

Prover

Verifier

Aggregate w/ a SNARK/STARK.

\[ sp0 \leftarrow \text{Prove}(c0 \ldots cn, \text{data } 0 \ldots \text{data } cn) \]

\[ \{0,1\} \leftarrow \text{Verify}(sp0) \]
Proofs-of-Replication (PoRep) & Proofs-of-Spacetime (PoSt)
The Storage Market handles storing data (PUT). The Retrieval Market handles getting data (GET).

Both markets can happen off-chain. This is especially important for sub-second latency retrieval.
<table>
<thead>
<tr>
<th>Client</th>
<th>Network</th>
<th>Miner</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>SendPieces(P)</td>
<td></td>
<td>RecvPieces(P)</td>
</tr>
<tr>
<td>AddOrder(deal)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>RecvPieces(P)</td>
<td></td>
<td>SendPieces(P)</td>
</tr>
<tr>
<td>AddOrder(deal)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>AssignOrders(OB)</td>
<td></td>
<td>PledgeSector(s)</td>
</tr>
<tr>
<td>RepairOrders(OB)</td>
<td></td>
<td>SealSector(s)</td>
</tr>
<tr>
<td>CheckSectors(S)</td>
<td></td>
<td>ProveSector(s)</td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>AddOrder(ask)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
</tbody>
</table>
Storage Market

MatchOrders(OB)
Storage Market

SendPieces(P)

RecvPieces(P)

DEAL
Storage Market

AddOrder(deal)
Storage Market

BID

ASK

DEAL

DEAL

ASK

BID
<table>
<thead>
<tr>
<th>Client</th>
<th>Network</th>
<th>Miner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AddOrder(ask)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>MatchOrders(OB)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AddOrder(deal)</td>
<td>AddOrder(deal)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>AddOrder(ask)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>MatchOrders(OB)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AddOrder(deal)</td>
<td>AddOrder(deal)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>AddOrder(ask)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>AssignOrders(OB)</td>
<td>PledgeSector(s)</td>
</tr>
<tr>
<td>3</td>
<td>MatchOrders(OB)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>AddOrder(deal)</td>
<td>AddOrder(deal)</td>
</tr>
<tr>
<td>5</td>
<td>RepairOrders(OB)</td>
<td>SealSector(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProveSector(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CheckSectors(S)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>AddOrder(ask)</td>
<td>FindPieces(P)</td>
</tr>
<tr>
<td>3</td>
<td>AddOrder(bid)</td>
<td>AddOrder(ask)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>AddOrder(deal)</td>
</tr>
</tbody>
</table>
Retrieval Market

Off-chain State or Payment Channel Network
Retrieval Market

Off-chain State or Payment Channel Network

AddOrder(ask)

AddOrder(ask)
Retrieval Market

Off-chain State or Payment Channel Network
Retrieval Market

Off-chain State or Payment Channel Network

MatchOrders(OB)
Retrieval Market

Off-chain State or Payment Channel Network
## Contents

1. **Introduction** ................................................................. 4  
   1.1 Elementary Components .............................................. 4  
   1.2 Protocol Overview ................................................... 4  
   1.3 Paper organization ................................................... 4  

2. **Definition of a Decentralized Storage Network** ................. 8  
   2.1 Fault tolerance ....................................................... 8  
   2.2 Properties ............................................................ 8  

3. **Proof-of-Replication and Proof-of-Spacetime** ................... 10  
   3.1 Motivation ............................................................ 10  
   3.2 Proof-of-Replication ............................................... 10  
   3.3 Proof-of-Spacetime ................................................ 11  
   3.4 Practical PoRep and PoSt ......................................... 11  
   3.5 Usage in Filecoin .................................................. 14  

4. **Filecoin: a DSN Construction** ....................................... 16  
   4.1 Setting ............................................................... 16  
   4.2 Data Structures .................................................... 17  
   4.3 Protocol ............................................................. 17  
   4.4 Guarantees and Requirements .................................... 21  

5. **Filecoin Storage and Retrieval Markets** .......................... 24  
   5.1 Verifiable Markets ................................................ 24  
   5.2 Storage Market .................................................... 24  
   5.3 Retrieval Market .................................................. 27  

6. **Useful Work Consensus** ................................................ 30  
   6.1 Motivation ............................................................ 30  
   6.2 Filecoin Consensus ................................................ 30  

7. **Smart Contracts** ...................................................... 33  
   7.1 Contracts in Filecoin ............................................... 33  
   7.2 Integration with other systems ................................... 33  

8. **Future Work** ........................................................... 34  
   8.1 Ongoing Work ....................................................... 34  
   8.2 Open Questions .................................................... 34  
   8.3 Proofs and Formal Verification .................................. 35
Filecoin Consensus

- As we layered the protocol on top of Ethereum we encountered many problems, such as layer 2 censoring issues.

- Separately, the wastefulness of Proof-of-Work has always frustrated us.

- Filecoin v1 attempted to weave storage into consensus but could not do it.

- Since then, the state-of-the-art advanced tremendously. In particular, the new Proof-of-Stake protocols created a model that works for us.
Filecoin Consensus

Contributions:

- One of the main contributions of the Filecoin Protocol is: constructing a “useful work” consensus, based on a useful Proof-of-Storage.

- More specifically, our consensus that uses Proofs-of-Replication as the core work function, aggregated in a Proof-of-Spacetime, with a Secret Leader Election construction (as in Proof-of-Stake protocols).
Contributions:

- Finally, this work motivated us to think deeply about the nature of Byzantine Fault Tolerance (BFT) and using resources — such as hashing or storage — as voting influence. This led us to create a new model for Consensus Protocols, which we named **Power Fault Tolerance (PFT)**.

- The model has since helped us analyze other protocols and highlight weaknesses. It has also yielded interesting new consensus protocols, and (what appears to be) a big result for asynchronous consensus protocols.
Proofs-of-Replication (PoRep)

Proofs-of-Spacetime (PoSt)

Secret Leader Election (SLE)

Expected Consensus (EC)
Secret Leader Election (SLE)

A method of secretly selecting one member of a group, according to a weighted distribution.
Secret Leader Election (SLE)

A method of secretly selecting one member of a group, according to a weighted distribution.
Secret Leader Election (SLE)

A method of secretly selecting one member of a group, according to a weighted distribution.

This is similar to Proof-of-Stake protocols like CoA, Snow White, and Algorand.
Secret Leader Election (SLE)

Fair election - random number

Verifiable - verify both the miner and the block

Secret - avoid bribery/collusion/corruption

\[
\mathcal{H} \left( \langle t || \text{rand}(t) \rangle \mathcal{M}_i \right) / 2^L \leq \frac{p^t_i}{\sum_j p^t_j}
\]

verifiable, personal, random, secret number between 0 and 1, at a known time

resulting in a fair lottery
Secret Leader Election (SLE)

- Problem: this SLE elects a number of leaders sampled from a distribution centered at 1.
- Ideally, we would always elect a single leader.
- This is hard to do in an uncensorable and fair way.

★ This is an open-ish problem!

$$\mathcal{H} \left( \langle t \mid \text{rand}(t) \rangle \mathcal{M}_i \right) / 2^L \leq \frac{p_i^t}{\sum_j p_j^t}$$

verifiable, personal, random, secret number between 0 and 1, at a known time

resulting in a fair lottery
Expected Consensus (EC)

A consensus protocol using a lottery leader election capable of tolerating epochs with 0 or multiple leaders elected.
A consensus protocol using a lottery leader election capable of tolerating epochs with 0 or multiple leaders elected.
Expected Consensus (EC)

A consensus protocol using a lottery leader election capable of tolerating epochs with 0 or multiple leaders elected.

Here only because SLE yields >1 leader. A better SLE (exactly 1) would obviate EC.
**Expected Consensus (EC)**

A consensus protocol using a lottery leader election capable of tolerating epochs with 0 or multiple leaders elected.

Here only because SLE yields >1 leader. A better SLE (exactly 1) would obviate EC.

Other protocols use round-based Byzantine Agreement style protocols between the elected leaders (eg Algorand)
Expected Consensus (EC)

A consensus protocol using a lottery leader election capable of tolerating epochs with 0 or multiple leaders elected.

Here only because SLE yields >1 leader. A better SLE (exactly 1) would obviate EC.

Other protocols use round-based Byzantine Agreement style protocols between the elected leaders (eg Algorand)
Expected Consensus (EC)

Here’s how we narrow down participants into consensus:

Start with many participants (thousands, millions, ...)

SLE yields around 1 leader.

\[ P(|\text{leaders}|=x) \rightarrow 0 \quad \text{as } x \rightarrow \infty \]

EC yields 1 log. Full consensus.
Storage Power Consensus

putting the pieces together
Useful Proof of Work for Consensus

First: pledging.
First: pledging. Then sealing.

- (1) Collect pieces into sectors.
First: pledging. Then sealing.

- (1) Collect pieces into sectors.
- (2) Once full, Seal them.
First: pledging. Then sealing.

1. Collect pieces into sectors.
2. Once full, Seal them.
3. Post proof of the seal.
Useful Proof of Work for Consensus

First: pledging. Then sealing.

- (1) Collect pieces into sectors.
- (2) Once full, Seal them.
- (3) Post proof of the seal.
- (4) Now miner has 2 sealed sectors. Her mining “power” = 2.
Useful Proof of Work for Consensus

First: pledging. Then sealing.

1. Collect pieces into sectors.
2. Once full, seal them.
3. Post proof of the seal.
4. Now miner has 2 sealed sectors. Her mining “power” = 2.
5. Miner can now mine. Probability of winning: $2/\text{count(sectors)}$
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Now: How to mine / extend the chain

c0 = H(head)

The challenge picks a sector randomly, and gives an input to a Proof-of-Replication over that sector.
Now: How to mine / extend the chain

\[ c_0 = H(\text{head}) \]
\[ p_0 = \text{PoRep.Prove}(c_0, \text{sealedSectors}) \]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

\[
c_0 = H(\text{head})
p_0 = \text{PoRep.Prove}(c_0, \text{sealedSectors})
c_1 = H(p_0)
p_1 = \text{PoRep.Prove}(c_1, \text{sealedSectors})
\cdots
\]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

Do $t$ proofs, where $t$ is a difficulty estimator like Bitcoin’s, which tracks the proving speed.

The number of proofs $t$ is chosen by the network such that on expectation $t$ proofs are solved in a single block time (eg 30s).
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

Then, take the last proof as the random sample for the *Secret Leader Election*.

\[
H\left(\langle t||\text{rand}(t)\rangle_{\mathcal{M}_i}\right) / 2^L \leq \frac{p_i^t}{\sum_j p_j^t}
\]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

Then, take the last proof as the random sample for the *Secret Leader Election*.

\[
\mathcal{H}(\langle t || \text{rand}(t) \rangle_{\mathcal{M}_i}) / 2^L \leq \frac{p^t_i}{\sum_j p^t_j}
\]

\[
\mathcal{H}(\langle r \rangle_{\mathcal{M}_i}) / 2^L \leq \text{# of sectors miner has}
\]

\[
\text{# of sectors in network}
\]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

On expectation, only one leader is chosen. Run Expected Consensus.
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine/extend the chain
Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

\[ H(\langle r \rangle_{M_i}) / 2^L \leq \begin{array}{c} 2 \\ 6 \end{array} \]

\[ H(\langle r \rangle_{M_i}) / 2^L \leq \begin{array}{c} 4 \\ 6 \end{array} \]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

\[ H\left(\left\langle r, M_i\right\rangle\right)/2^L \leq \left\langle\begin{array}{c} 2 \\ 6 \\ 4 \\ 6 \end{array}\right\rangle \]
Useful Proof of Work for Consensus

Now: How to mine / extend the chain

Over time, network is creating a publicly verifiable record that all the sectors are being stored. Can aggregate or re-introduce the proof-chains into the blockchain. (Compress w/ SNARKs or other tools)
# Contents

1 Introduction
   1.1 Elementary Components ........................................... 4
   1.2 Protocol Overview .................................................. 4
   1.3 Paper organization .................................................. 4

2 Definition of a Decentralized Storage Network ........................................... 8
   2.1 Fault tolerance ...................................................... 8
   2.2 Properties .......................................................... 8

3 Proof-of-Replication and Proof-of-Spacetime ........................................... 10
   3.1 Motivation .......................................................... 10
   3.2 Proof-of-Replication ............................................... 10
   3.3 Proof-of-Spacetime ................................................ 11
   3.4 Practical PoRep and PoSt ......................................... 11
   3.5 Usage in Filecoin .................................................. 14

4 Filecoin: a DSN Construction ................................................... 16
   4.1 Setting .............................................................. 16
   4.2 Data Structures ..................................................... 17
   4.3 Protocol ............................................................ 17
   4.4 Guarantees and Requirements .................................... 21

5 Filecoin Storage and Retrieval Markets ........................................... 24
   5.1 Verifiable Markets .................................................. 24
   5.2 Storage Market ...................................................... 24
   5.3 Retrieval Market .................................................... 27

6 Useful Work Consensus ....................................................... 30
   6.1 Motivation .......................................................... 30
   6.2 Filecoin Consensus ................................................ 30

7 Smart Contracts ............................................................ 33
   7.1 Contracts in Filecoin ............................................. 33
   7.2 Integration with other systems ................................... 33

8 Future Work ............................................................... 34
   8.1 Ongoing Work ...................................................... 34
   8.2 Open Questions .................................................... 34
   8.3 Proofs and Formal Verification ................................... 35
<table>
<thead>
<tr>
<th></th>
<th>Client</th>
<th>Network</th>
<th>Miner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>SendPieces(P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AddOrder(deal)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>AddOrder(ask)</td>
<td>MatchOrders(OB)</td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RecvPieces(P)</td>
<td></td>
<td>SendPieces(P)</td>
</tr>
<tr>
<td>4</td>
<td>AddOrder(deal)</td>
<td></td>
<td>AddOrder(deal)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>AssignOrders(OB)</td>
<td>PledgeSector(s)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>SealSector(s)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>ProveSector(s)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>RepairOrders(OB)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CheckSectors(S)</td>
<td></td>
<td>AddOrder(bid)</td>
</tr>
<tr>
<td>2</td>
<td>AddOrder(ask)</td>
<td></td>
<td>FindPieces(P)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>AddOrder(deal)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Guarantees & Requirements:

- **Achieving Integrity:** Pieces are named after their cryptographic hash. After a Put request, clients only need to store this hash to retrieve the data via Get and to verify the integrity of the content received.

- **Achieving Retrievability:** In a Put request, clients specify the replication factor and the type of erasure coding desired, specifying in this way the storage to be \((f, m)\)-tolerant.

- **Achieving Public Verifiability and Auditability:** Storage Miners are required to submit their proofs-of-storage \((\pi\text{-SEAL}, \pi\text{-POST})\) to the blockchain. Any user in the network can verify the validity of these proofs, without having access to the outsourced data. Since the proofs are stored on the blockchain, they are a trace of operations that can be audited at any time.

- **Achieving Incentive Compatibility:** Informally, miners are rewarded for the storage they are providing. When miners commit to store some data, then they are required to generate proofs. Miners that skip proofs are penalized (by losing part of their collateral) and not rewarded for their storage.

- **Achieving Confidentiality:** Clients that desire for their data to be stored privately, must encrypt their data before submitting them to the network.
Filaxia: A Decentralized Storage Network

Prepared: 2022

Abstract

The Internet is the public medium of a revolution. These problems are not being solved, with thousands of decentralized storage networks and millions of volunteers working on research, prototypes, and applications. However, most of the work has been done by a few large companies and universities. The environment is ripe for innovation, with blockchains, distributed databases, and encrypted storage. The ecosystem is built on open-source software, with a central organization that dates back to the 1970s. The Internet is a critical technology for the future, a new kind of computing infrastructure. The context is a distributed storage network that scales to a global scale. The network is a distributed storage network that scales to a global scale. The network is a distributed storage network that scales to a global scale. The network is a distributed storage network that scales to a global scale.

This work:
1. Introduces the Filaxia Network, gives an overview of the protocol, and walks through several applications.
2. Designing a decentralized storage network: how distributed storage networks are different, how they differ from traditional storage.
3. The network performs well and has a high performance.
4. Implementing the network and conducting tests on smaller applications and in real-world scenarios.
5. Researching the network and conducting tests on larger applications, through their real-world functional libraries.

Note: Filaxia is a work in progress. Active research is under way, and work presented in this paper will appear in future Filaxia. The research and development are an ongoing project.

Open Problems

Questions

Engineering

The Protocol
# Contents

1 Introduction .......................... 4
   1.1 Elementary Components ................. 4
   1.2 Protocol Overview .................. 4
   1.3 Paper organization ................. 4

2 Definition of a Decentralized Storage Network 8
   2.1 Fault tolerance .................. 8
   2.2 Properties .................. 8

3 Proof-of-Replication and Proof-of-Spacetime 10
   3.1 Motivation .................. 10
   3.2 Proof-of-Replication .......... 10
   3.3 Proof-of-Spacetime .......... 11
   3.4 Practical PoRep and PoSt .......... 11
   3.5 Usage in Filecoin ........... 14

4 Filecoin: a DSN Construction 16
   4.1 Setting .................. 16
   4.2 Data Structures ........... 17
   4.3 Protocol .................. 17
   4.4 Guarantees and Requirements ........ 21

5 Filecoin Storage and Retrieval Markets 24
   5.1 Verifiable Markets ........... 24
   5.2 Storage Market ........... 24
   5.3 Retrieval Market .......... 27

6 Useful Work Consensus 30
   6.1 Motivation ........... 30
   6.2 Filecoin Consensus .......... 30

7 Smart Contracts 33
   7.1 Contracts in Filecoin ........... 33
   7.2 Integration with other systems ........ 33

8 Future Work 34
   8.1 Ongoing Work ........... 34
   8.2 Open Questions .......... 34
   8.3 Proofs and Formal Verification .......... 35
8.1 On-going Work

The following topics represent ongoing work.

- A specification of the Filecoin state tree in every block.
- Detailed performance estimates and benchmarks for Filecoin and its components.
- A full implementable Filecoin protocol specification.
- A sponsored-retrieval ticketing model where any client C1 can sponsor the download of another client C2 by issuing per-piece bearer-spendable tokens.
- A Hierarchical Consensus protocol where Filecoin subnets can partition and continue processing transactions during temporary or permanent partitions.
- Incremental blockchain snapshotting using SNARK/STARK
- Filecoin-in-Ethereum interface contracts and protocols.
- Blockchain archives and inter-blockchain stamping with Braid.
- Only post *Proofs-of-Spacetime* on the blockchain for conflict resolution.
- Formally prove the realizations of the Filecoin DSN and the novel *Proofs-of-Storage*. 
8.2 Open Questions

There are a number of open questions whose answers have the potential to substantially improve the network as a whole, despite the fact that none of them have to be solved before launch.

- A better primitive for the *Proof-of-Replication* Seal function, which ideally is $O(n)$ on decode (not $O(nm)$) and publicly-verifiable without requiring SNARK/STARK.

- A better primitive for the *Proof-of-Replication* Prove function, which is publicly-verifiable and transparent without SNARK/STARK.

- A transparent, publicly-verifiable *Proof-of-Retrievability* or other Proof-of-Storage.

- New strategies for retrieval in the Retrieval Market (e.g. based on probabilistic payments, zero knowledge contingent payments)

- A better secret leader election for the Expected Consensus, which gives exactly one elected leader per epoch.

- A better trusted setup scheme for SNARKs that allows incremental expansion of public parameters (schemes where a sequence of MPCs can be run, where each additional MPC strictly lowers probability of faults and where the output of each MPC is usable for a system).
RFPs and Bounties
Open Problems

Questions

The Protocol

Engineering

The Context
Questions?