Filecoin: from Proof-of-Space Blockchain to Decentralized Storage

Luca Nizzardo

Protocol Labs

October 2021

Protocol Labs

An open-source research, development, and deployment laboratory.
We aim to building the next generation of the internet with the focus on decentralisation!

Founded in 2014

Committed to Web 3

Many Different Projects

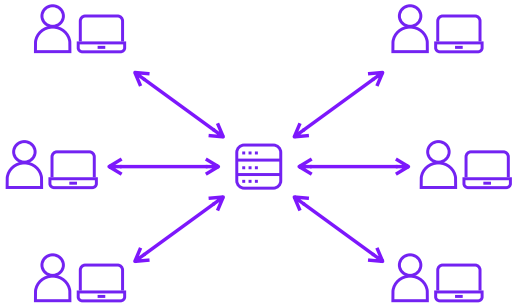
IPFS

libp2p

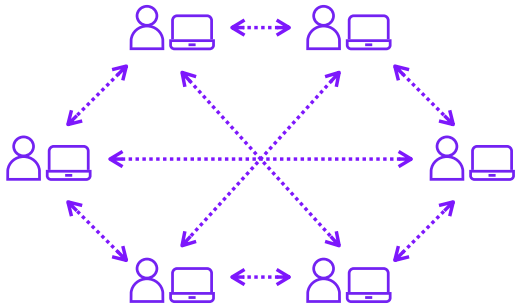
~150 Full Time Collaborators

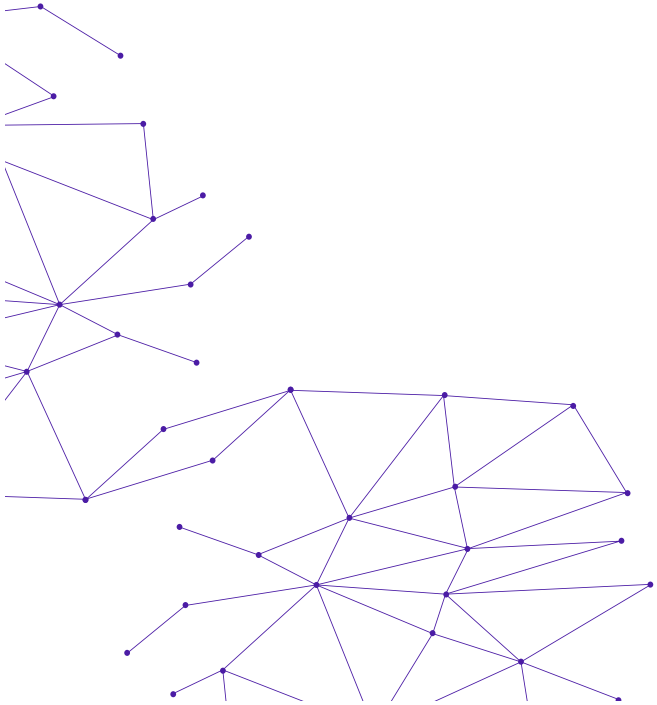
100% Remote Organization

Filecoin: Decentralise the Storage Market



VS

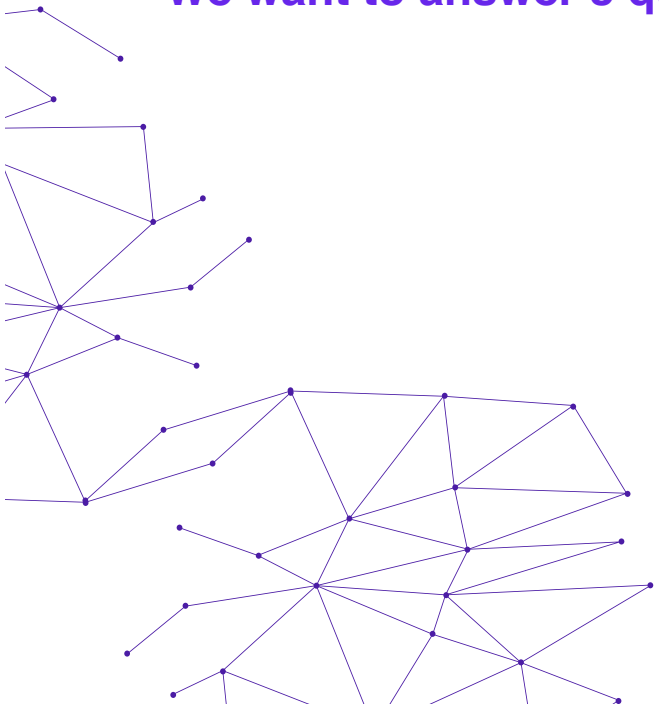




The **Blockchain** replaces the **Trusted Third Party** by keeping track of everything going on in the network

Talk Outline

We want to answer 3 questions:

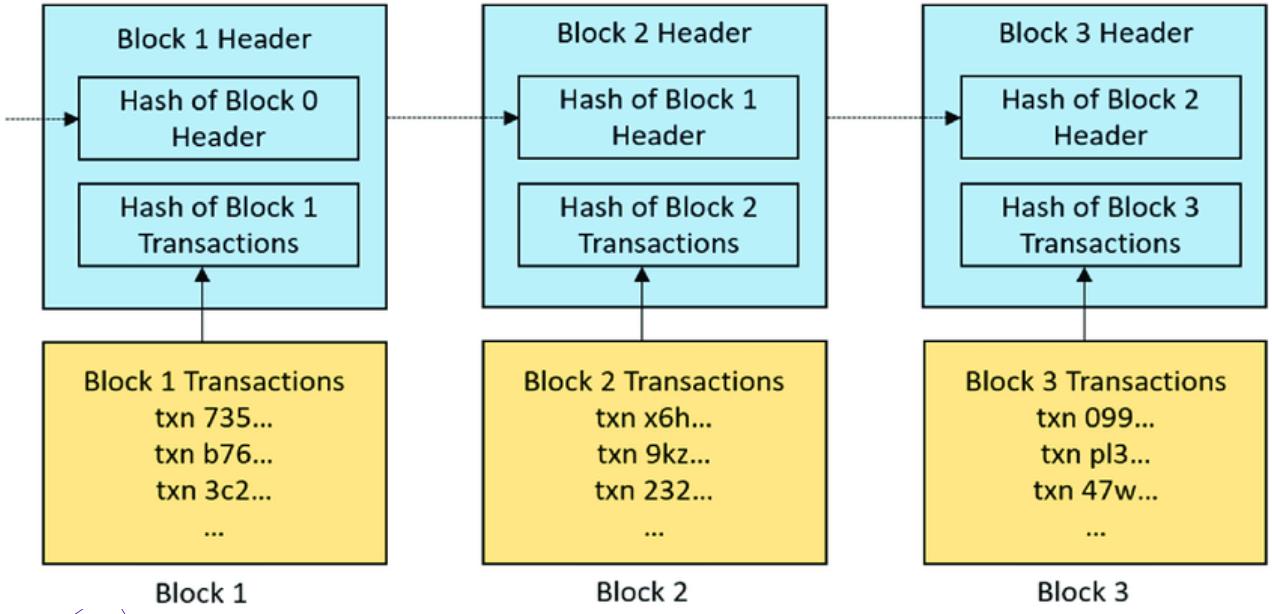


(Preliminaries)

1. What is a Proof of Space?
2. How PoS differs from PoW?
3. How do we build a Proof of Space and how do we use it for proving Persistent and Useful Storage in Filecoin?

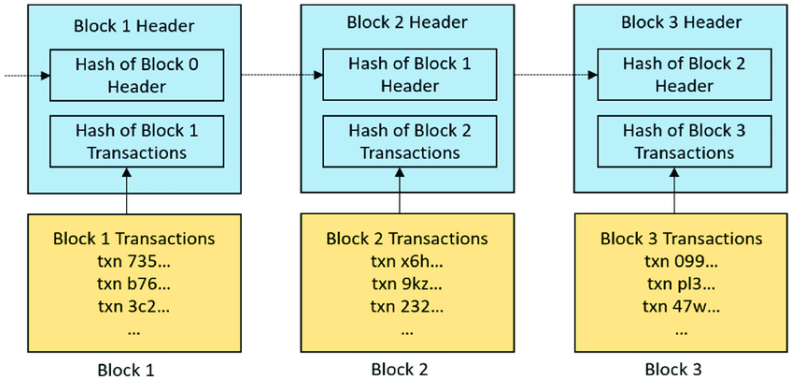
Blockchain Preliminaries

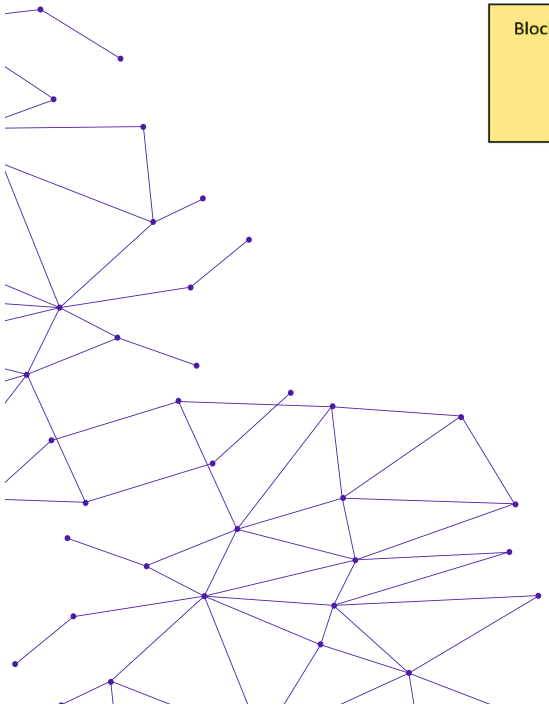
Blockchain = distributed (ie, maintained by a network) ledger organized in blocks



Blockchain preliminaries

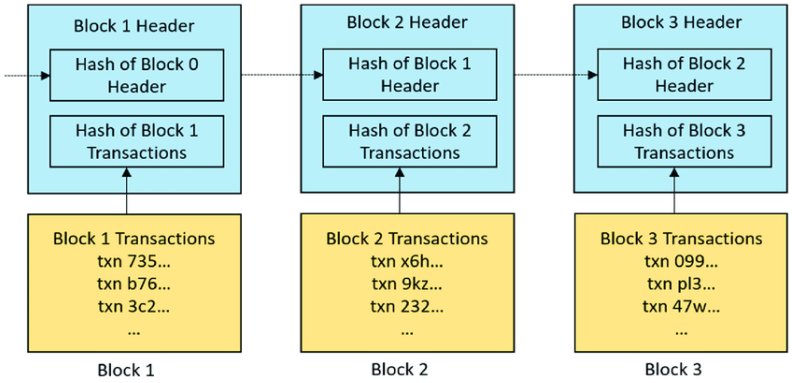
Critical problem: How do we agree on the next block?





Blockchain preliminaries

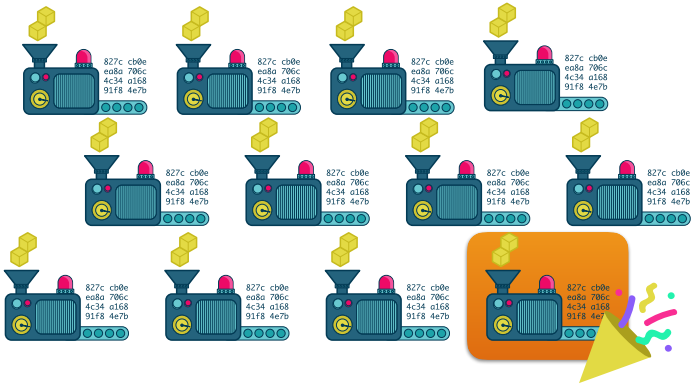
Critical Problem: How do we agree on the next block?



Bitcoin: Proof of Work!

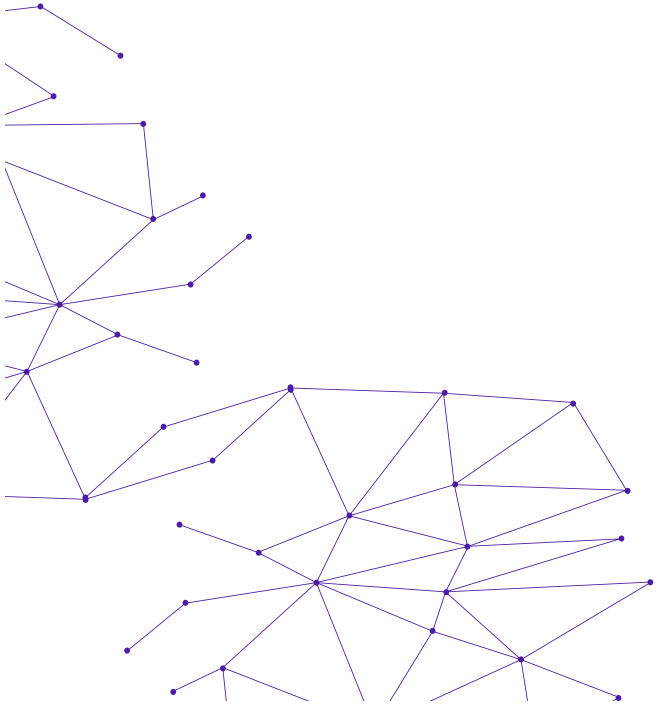
A **lottery** based on computational power chooses one node who creates the next block!

(Nakamoto Consensus Protocol)



Filecoin: What's New?

vs

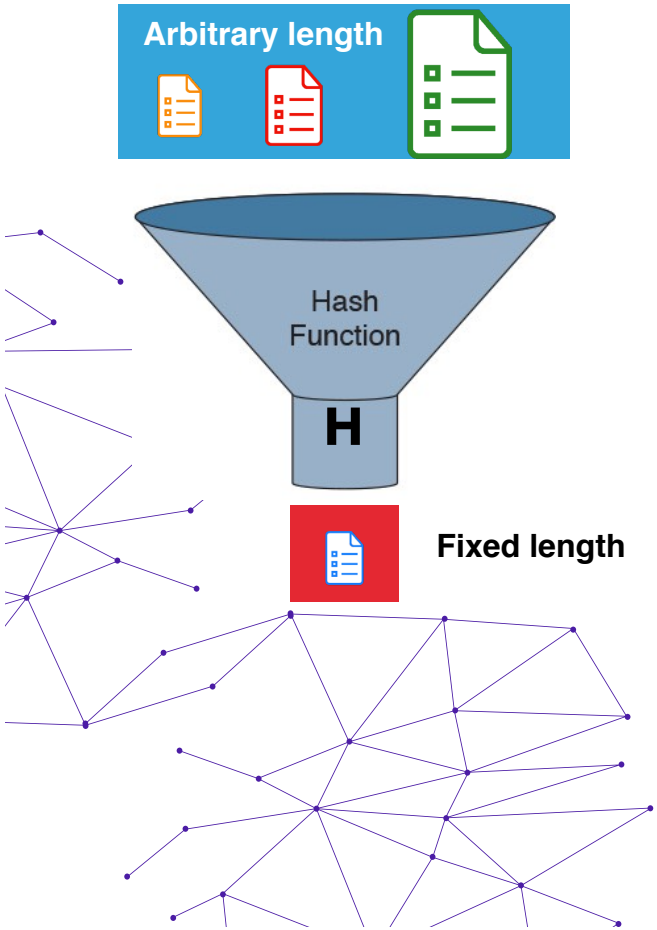


- Filecoin does not use Proof of Work
- Filecoin uses **Proof of Space!**
- We use Proof of Space both for
  - Enforcing Consensus
  - Proving Persistent Storage

The consensus cost is re-used to empower a storage market!

Preliminaries

Hash Functions

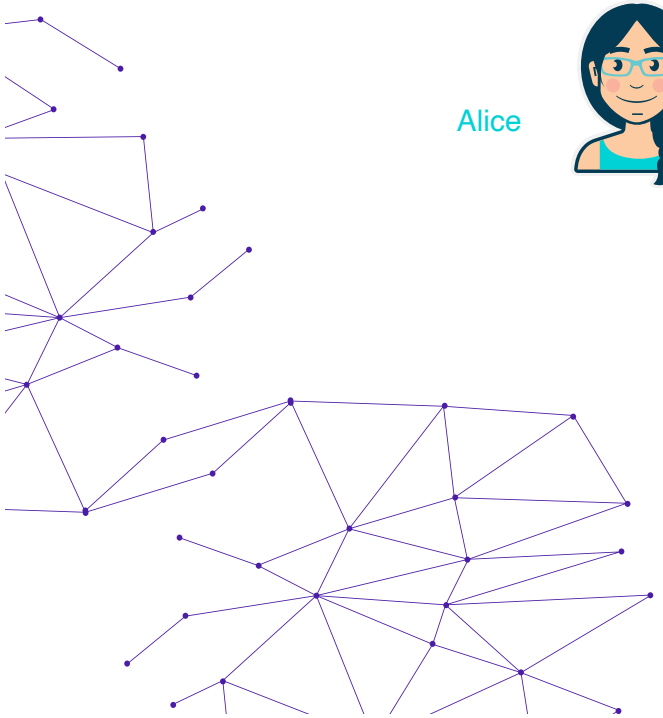


Properties

- Given it is difficult to find such that H( ) =
- It is difficult to find and such that H( ) = H( )
- If one takes and makes a minimal change , then H( ) and H( ) are totally different

Preliminaries

Commitment Schemes



I put my data into an envelope

Show me the data inside the envelope

Alice

Bob

Binding: Alice can not convince Bob that the data inside  is  instead of 

Hiding: Bob can not see inside 

Vector Commitment Scheme

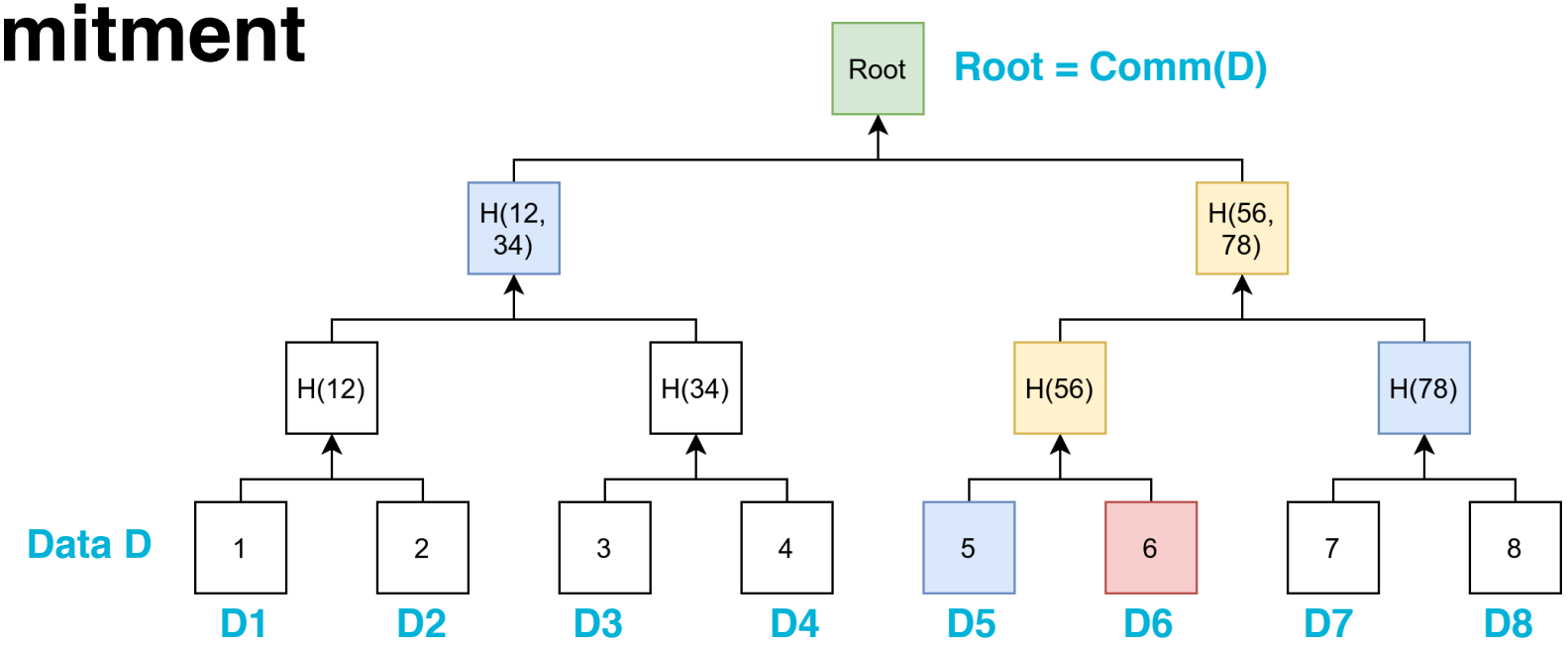
A vector commitment scheme is a triple of algorithms (Comm, Open, Verify)

- Comm(v1, v2, …) —> C
- Open(C, position i) —> Aux, v
- Verify(i, Aux, v, C) —> 0/1

Position Binding: Given C it in unfeasible to find v' ≠ v and Aux', Aux such that

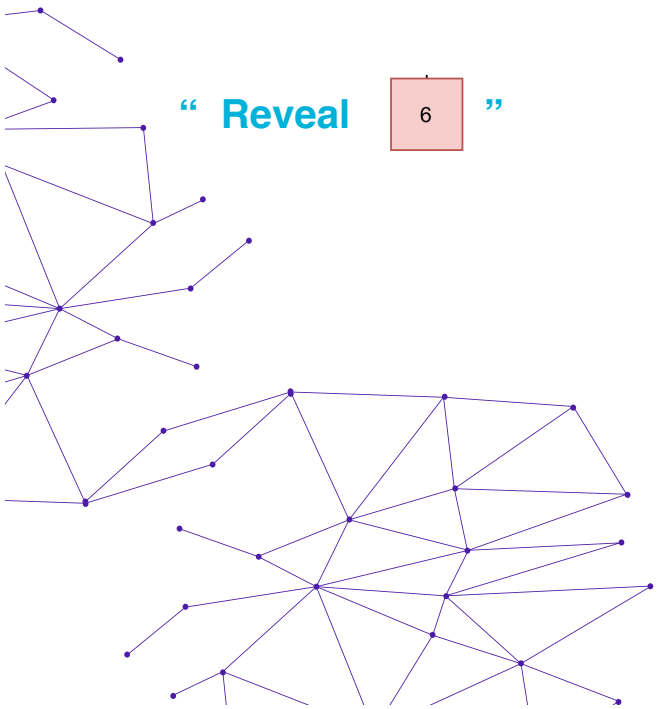
Verify(i,v', Aux', C) = Verify(i,v, Aux, C) = 1

Merkle-Tree (MT) Commitment

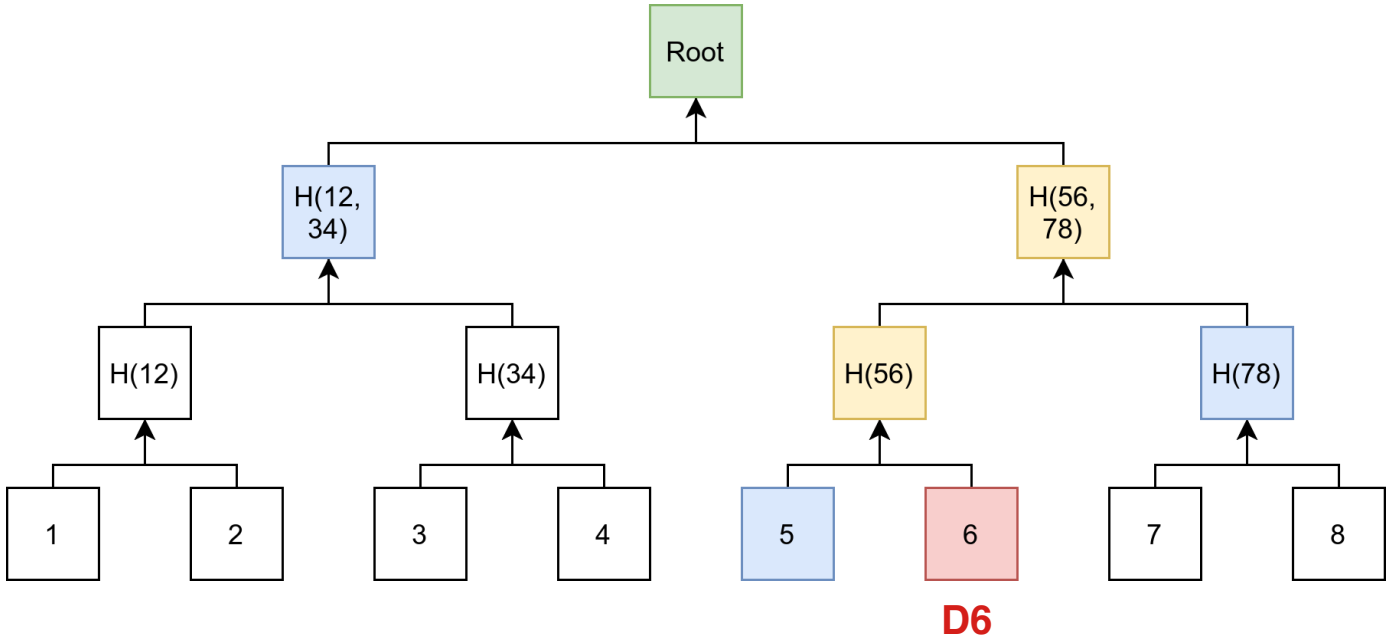


Preliminaries

Merkle-Tree (MT) Commitment

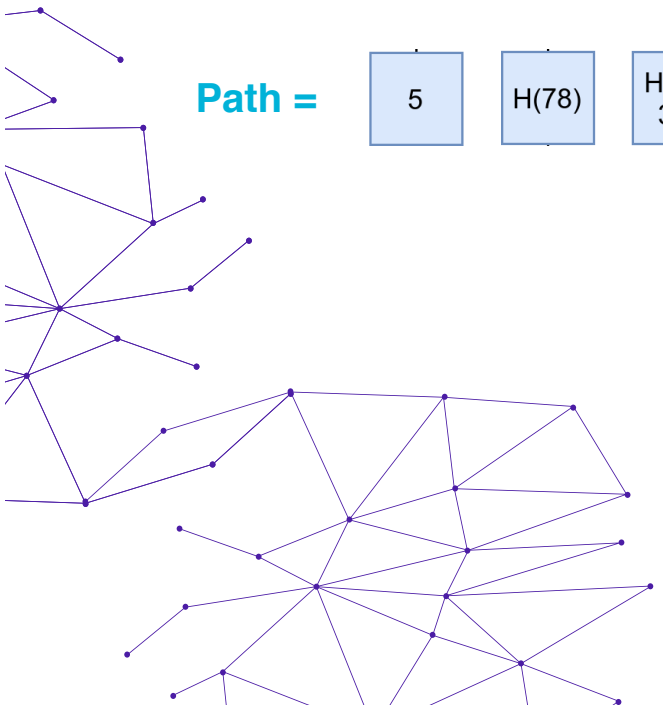


“ Reveal 6 ”

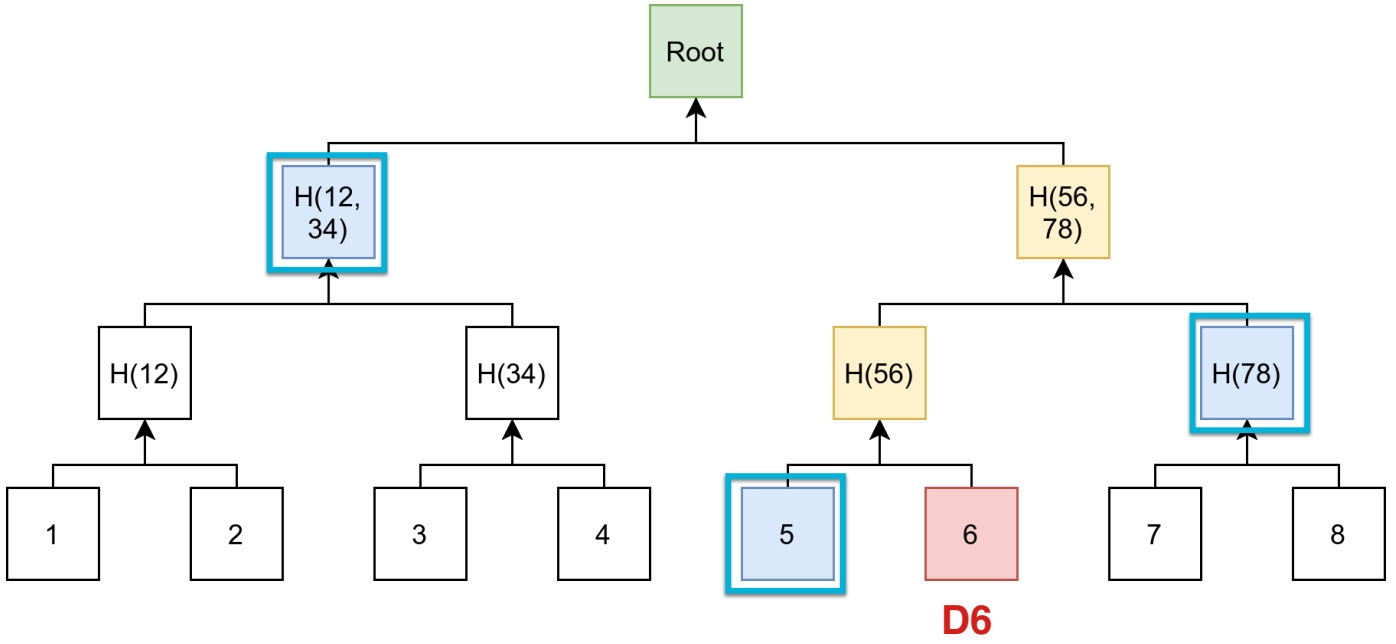


Preliminaries

Merkle-Tree (MT) Commitment



Path = 5 H(78) H(12, 34)

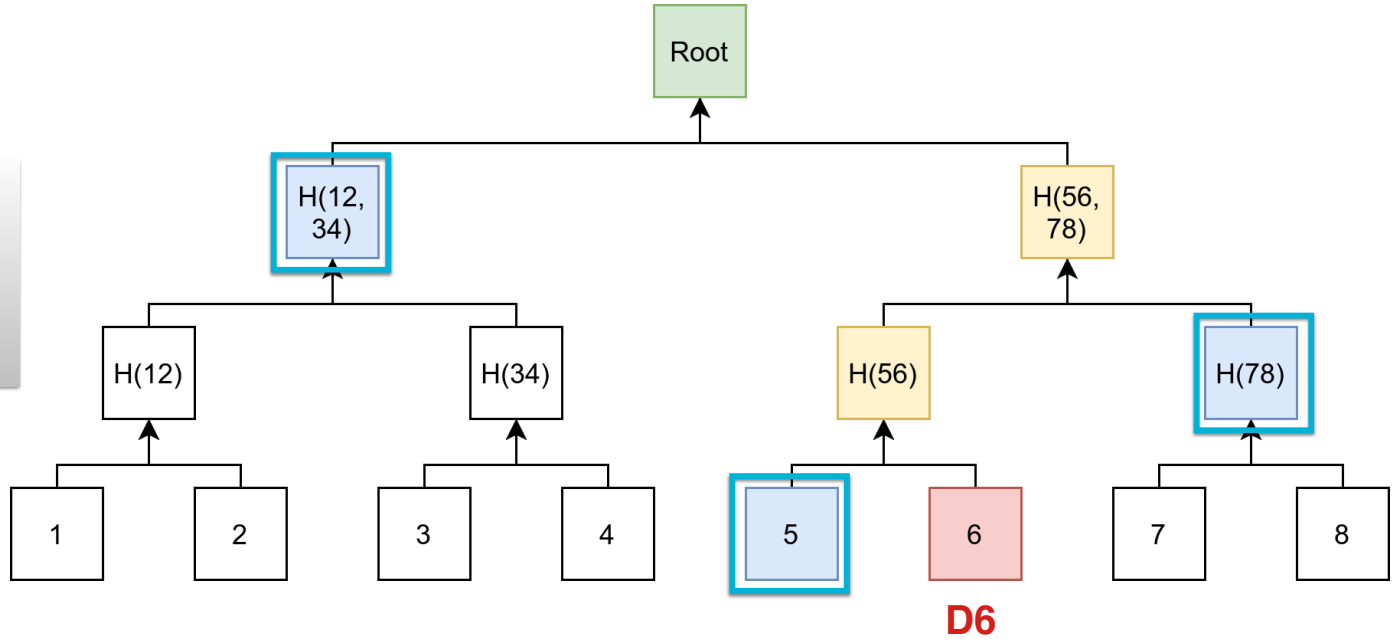


Preliminaries

Merkle-Tree (MT) Commitment

Size of D = 32 GiB = 2^30 x 32 bytes

Take H = SHA-256
1 commitment = 32 bytes
1 path = 30 x 32 bytes ~ 1kbyte



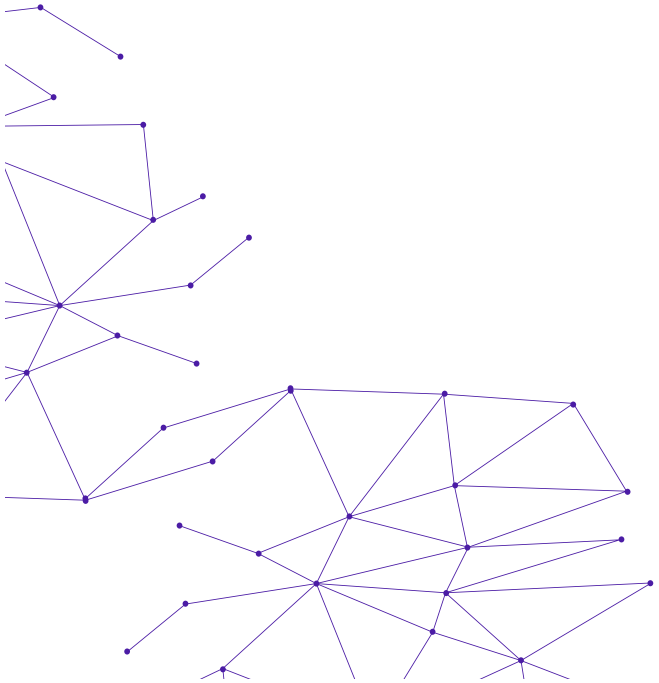
Question 1: What is a Proof of Space?

Proof of Space (PoS)

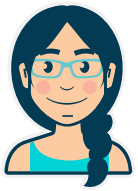
Two phases protocol:

1. **Initialization** (one-time setup)

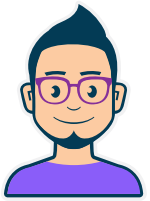
2. **Execution** (repeated audit phase)



Alice



Public Input: N



Bob

Bob checks that Alice stores an incompressible file of size N!

Question 1: What is a Proof of Space?

Proof of Space (PoS)

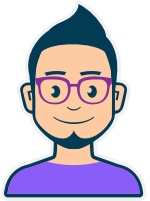
Two phases protocol:

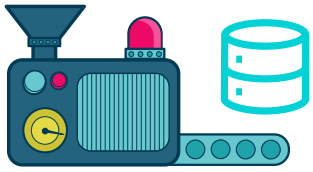
1. **Initialization** (one-time setup)

2. **Execution** (repeated audit phase)

Initialization:

Public Input: N





Advice

Comm( )

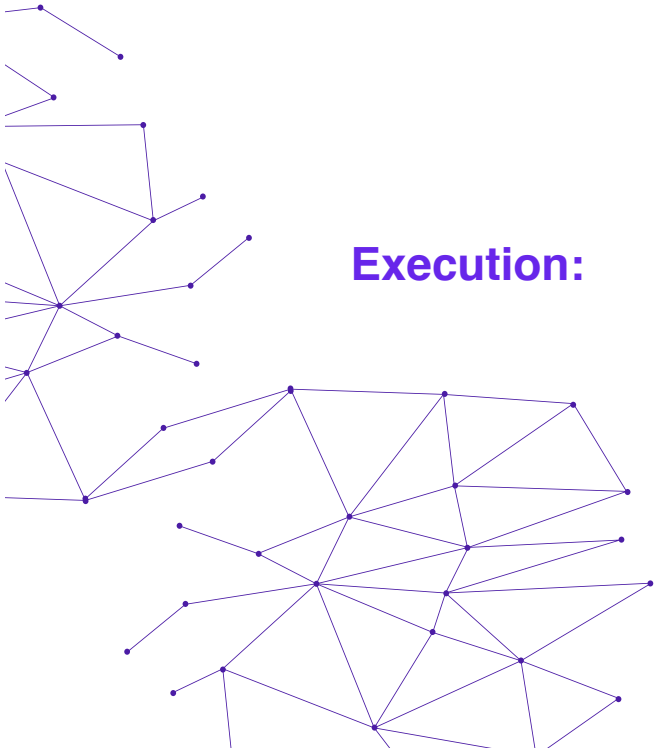
Question 1: What is a Proof of Space?

Proof of Space (PoS)

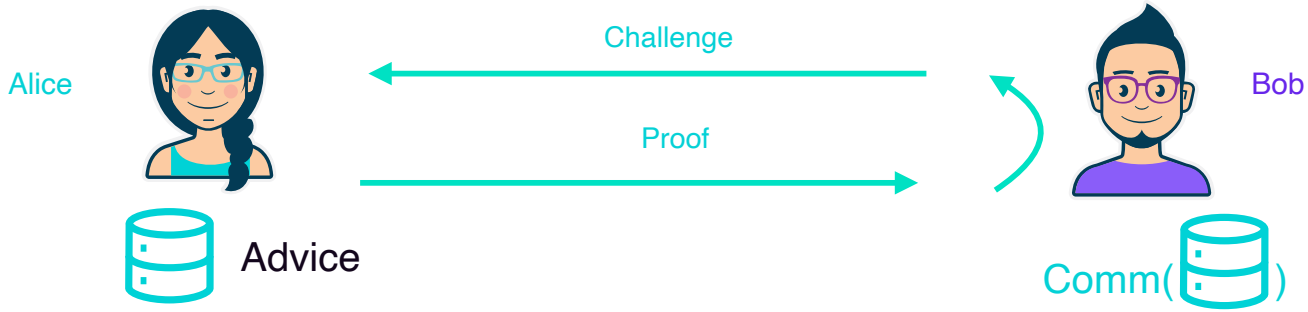
Two phases protocol:

1. **Initialization** (one-time setup)

2. **Execution** (repeated audit phase)



Execution:



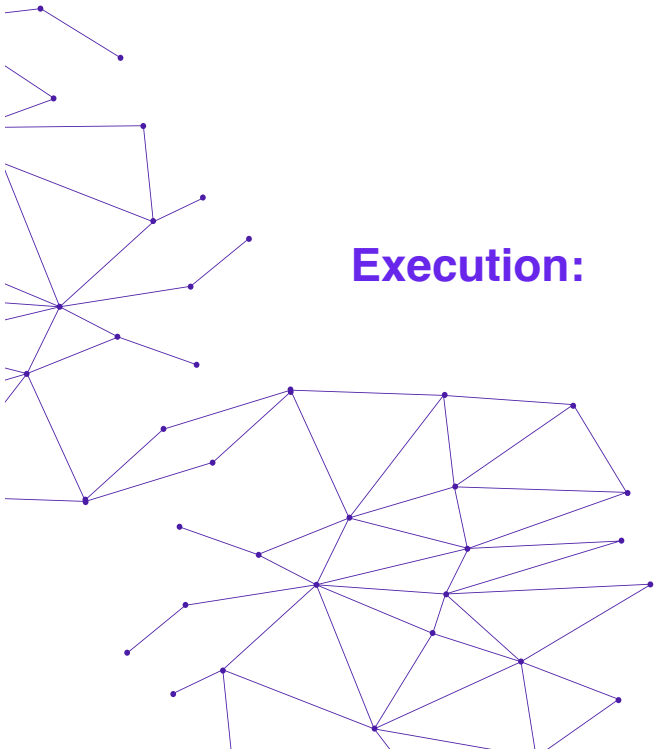
Question 1: What is a Proof of Space?

Proof of Space (PoS)

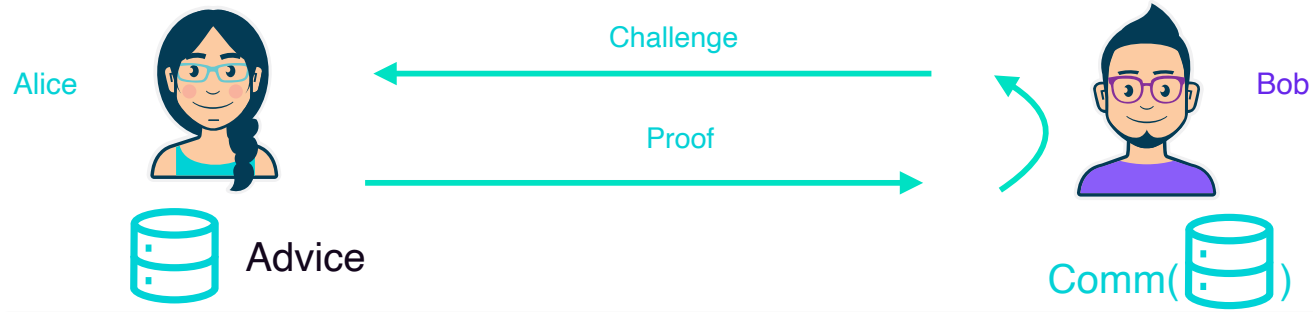
Two phases protocol:

1. **Initialization** (one-time setup)

2. **Execution** (repeated audit phase)



Execution:



Soundness (informal)

"To produce a convincing proof, Alice needs to store (a large part) of the advice or to do a computation of T steps".

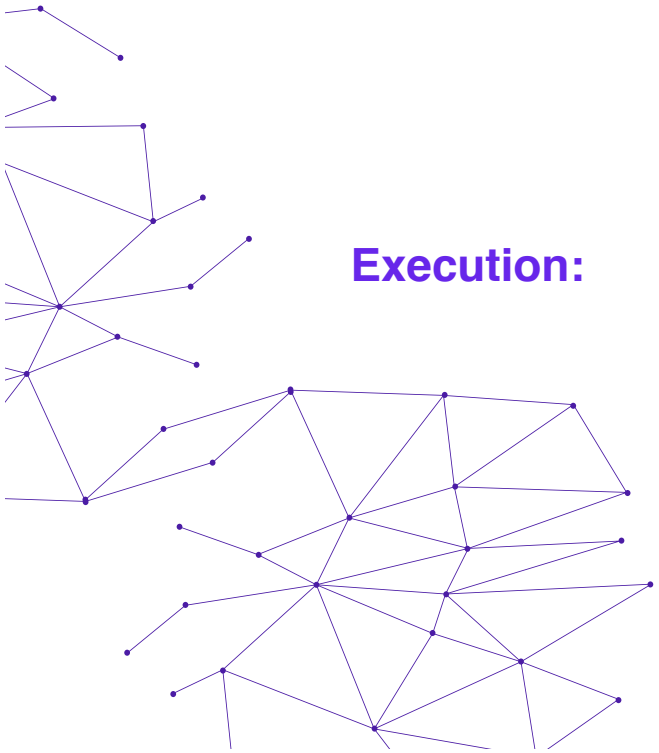
Question 1: What is a Proof of Space?

Proof of Space (PoS)

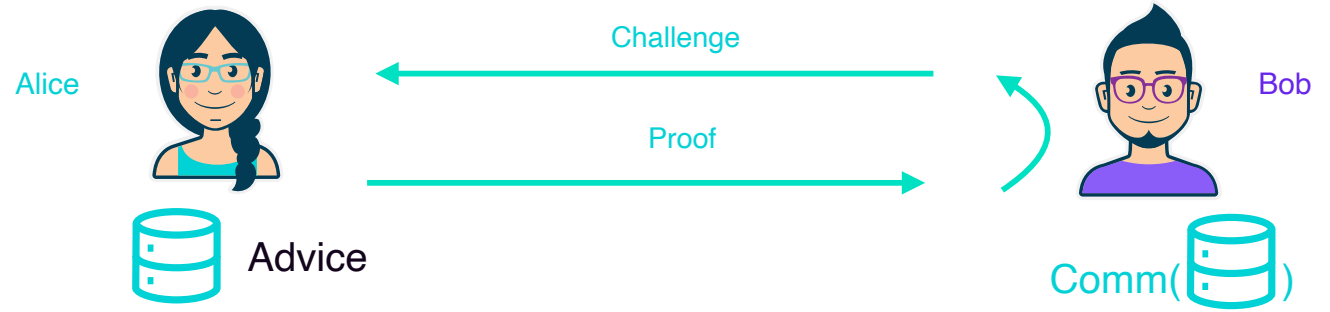
Two phases protocol:

1. **Initialization** (one-time setup)

2. **Execution** (repeated audit phase)



Execution:



Soundness (informal)

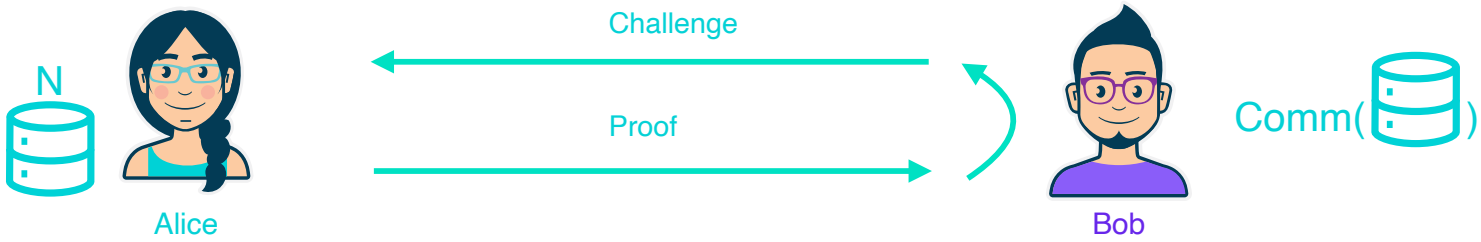
"To produce a convincing proof, Alice needs to store (a large part) of the advice or to do a computation of T steps".

If the *"T-step computation"* is infeasible/irrational
=> Alice is persistently storing the advice

Question 1: What is a Proof of Space?

Proof of Space (PoS)

Execution

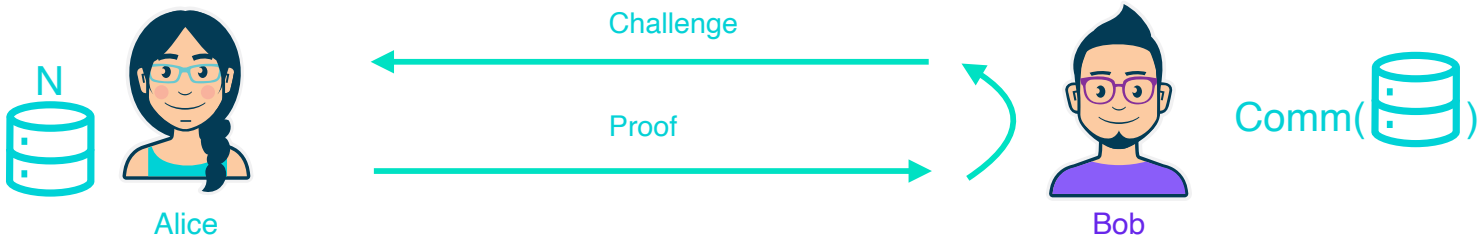


What does "T steps computation is infeasible mean in practice?

Question 1: What is a Proof of Space?

Proof of Space (PoS)

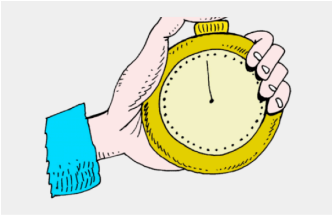
Execution



What does "T steps computation is infeasible mean in practice?

Time model: Alice must answer challenges in a limited window of time for the proof to be valid.

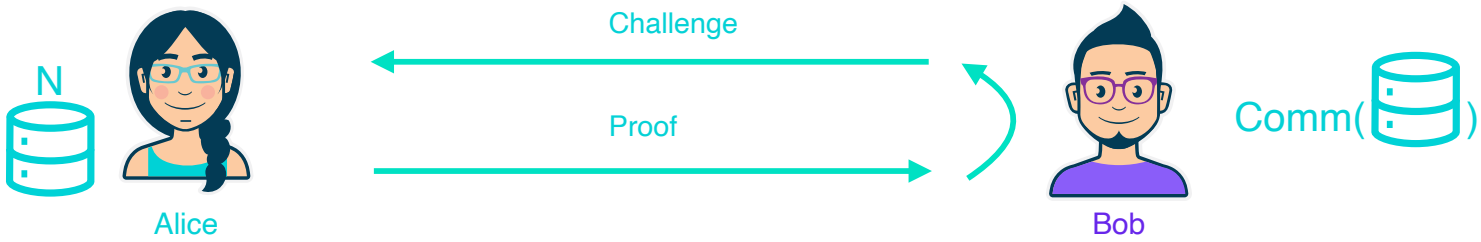
We need a *timing assumption* to translate T-steps in t seconds!



Question 1: What is a Proof of Space?

Proof of Space (PoS)

Execution

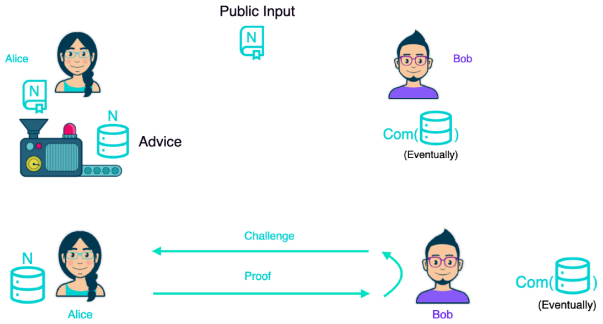


What does "T steps computation is infeasible" mean in practice?

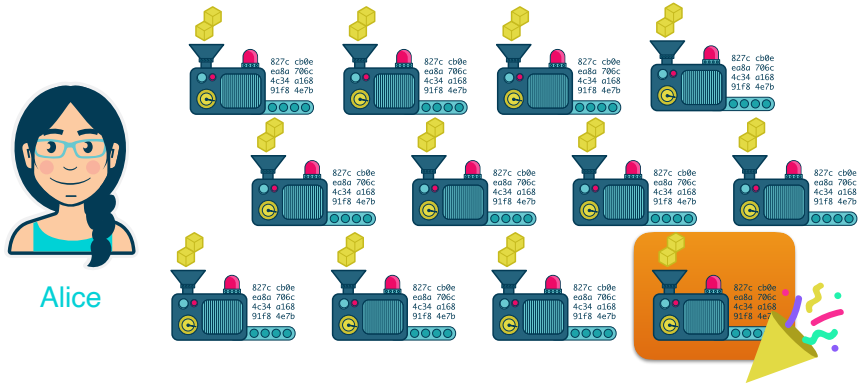
Cost Model: Alice chooses to store because T-steps are more expensive (rational prover). We need a *cost assumption* to translate T-steps in t dollars!

Question 2: How PoS is different from PoW?

Proof of Space (PoS)



Proof of Work (PoW)

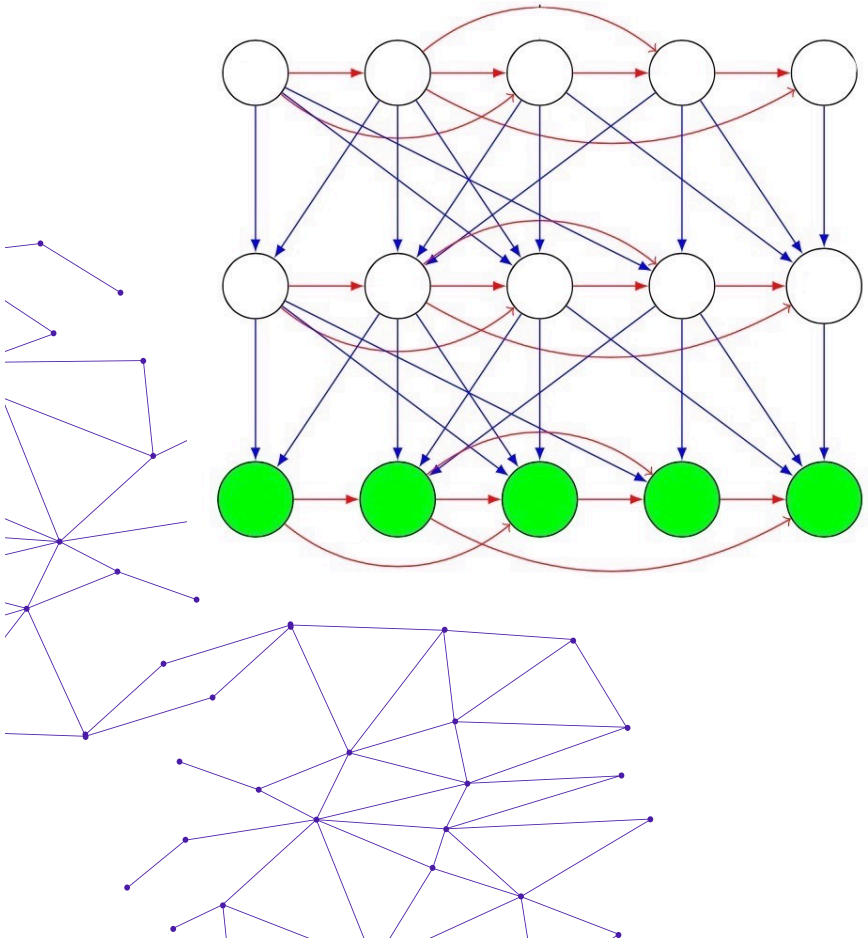


In PoS the underlying resource is (persistent) storage rather than computation.

A more efficient and green alternative to PoW in Nakamoto-Style Consensus Protocols!

Question 3: How do we build a Proof of Space?

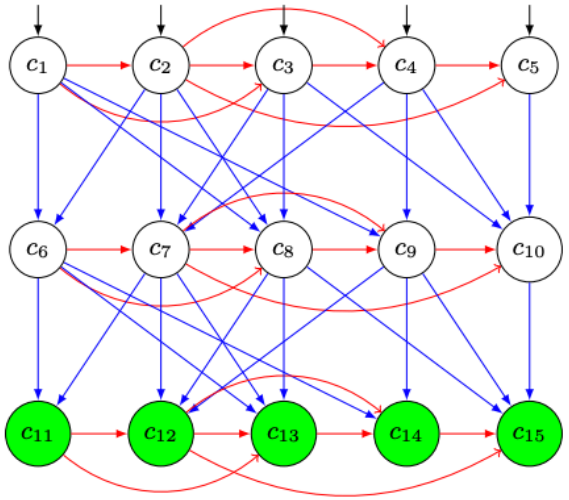
Graph-labelling based PoS



Fix a **graph**: set of nodes and edges among nodes (no cycles)

Question 3: How do we build a Proof of Space?

Graph-labelling based PoS

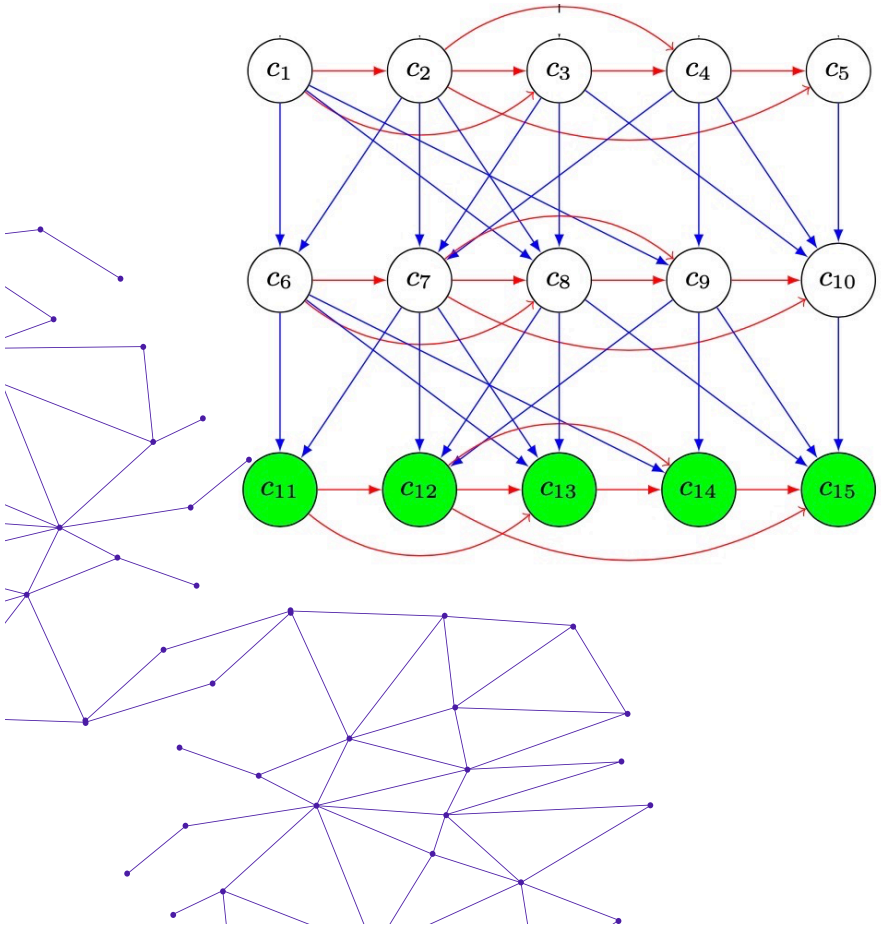


Initialization:

1. Build the labels of the graph nodes using hash of parents, layer by layer
2. Proof that labels are correctly generated by showing random nodes among all the layers are correctly built
3. Keep the last layer

Question 3: How do we build a Proof of Space?

Graph-labelling based PoS



A bit more formally:

Label of a node:
c_i = Hash(i ll Public Inputs ll c_j if j is a parent of i)

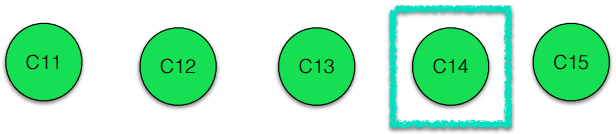
Advice = labels of green nodes

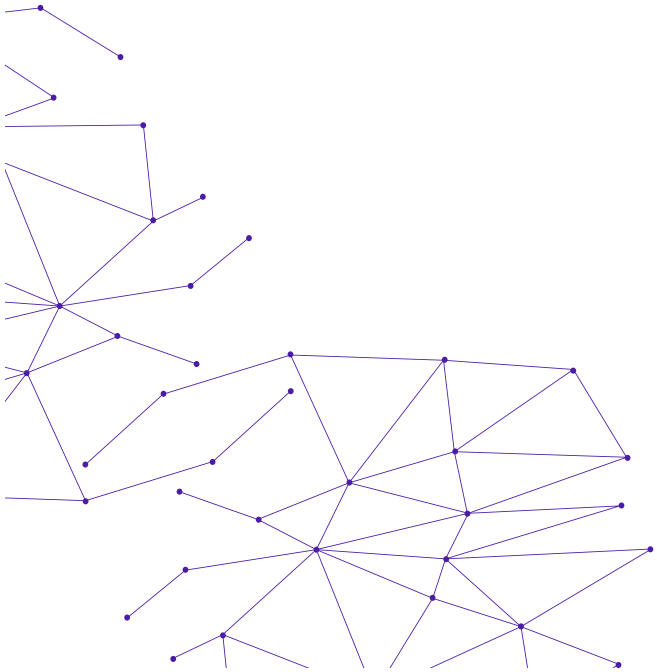
Commit to the advice:
Comm(Advice) = MT(c_11, ... ,c_15)

Alice stores the advice
Bob knows Comm(Advice)

Question 3: How do we build a Proof of Space?

Graph-labelling based PoS





Execution

Bob periodically sends a random challenge

Alice answer with the label and MT path

If she does not store it, must re-compute it!!

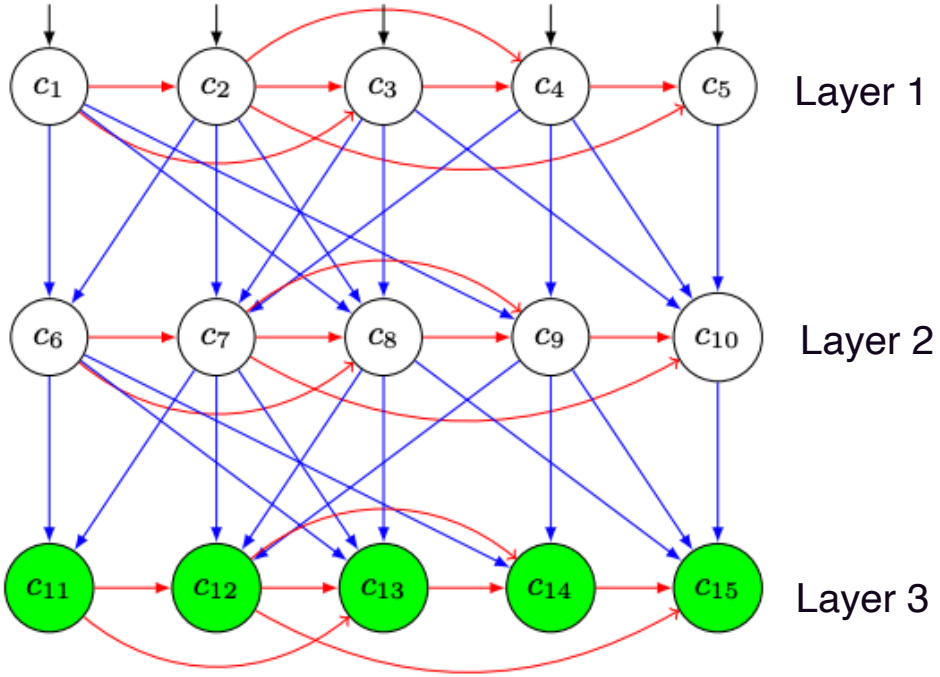
Why this is secure?

Re-computation is slow or expensive:

1. Time Model: If you have to re-compute nodes, you run out of time
2. Cost model: Recomputing labels is more expensive compared to storing.

Question 3: How do we build a Proof of Space?

Stacked-DRGs Graph (aka “SDR")

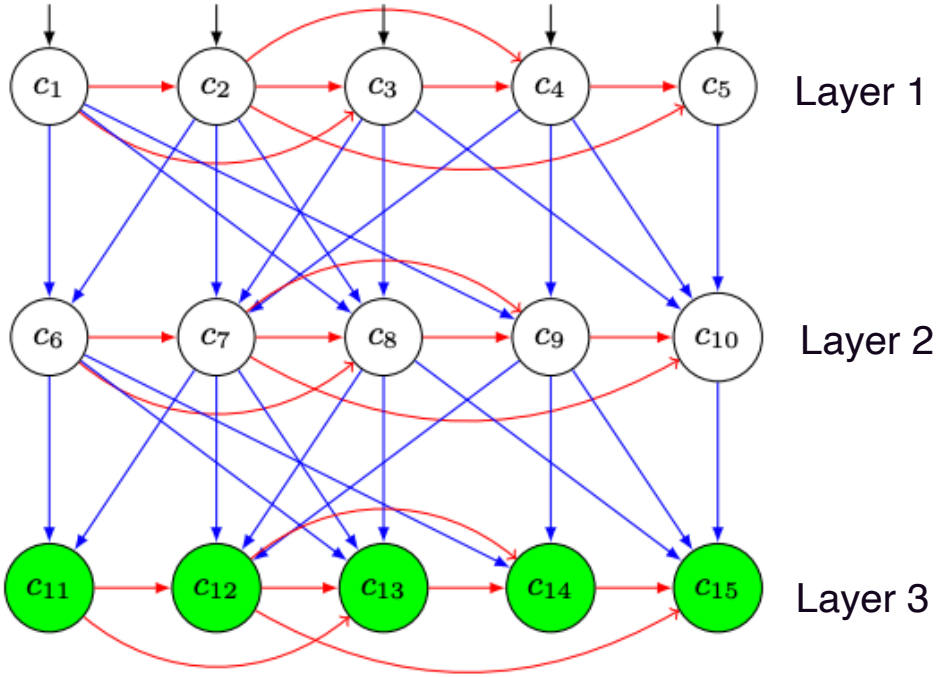


Each layer is a **depth-robust graph** (red edges)**:**
each set of 80% of the nodes has a long path

Each pair of layers is an **expander** (blue edges)**:**
an x fraction of nodes in the lower lever has a 2x of the nodes of the upper layer as parents (x < 1/3)

Question 3: How do we build a Proof of Space?

Stacked-DRGs Graph (aka “SDR")



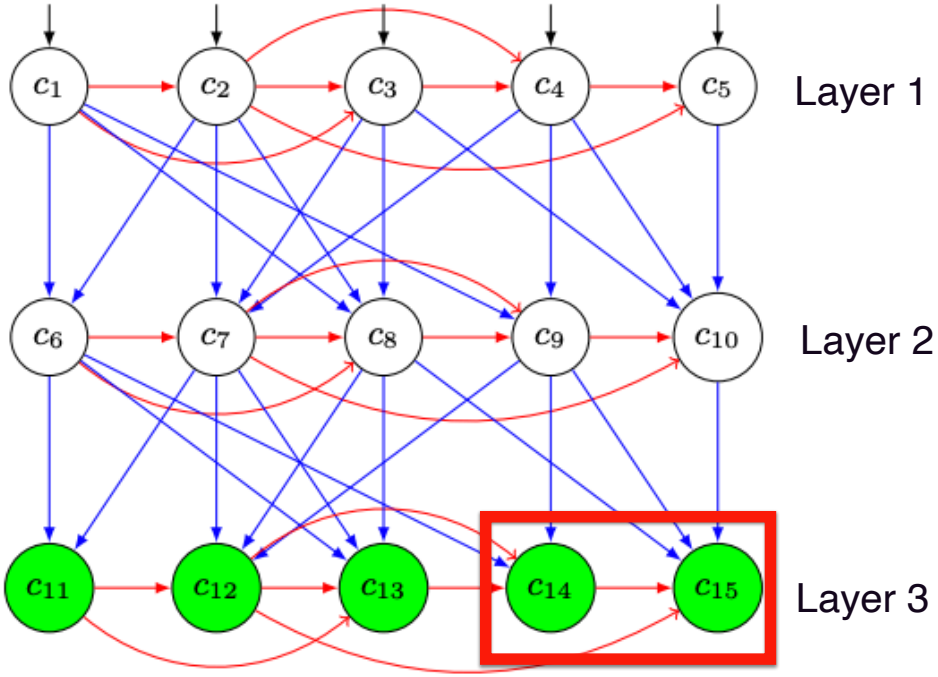
Security intuition:

- Miner does not store, need to **re-compute** it!
- Re-computing from scratch is slow/expensive because:

  a random node has a lot of parents (with high prob.)

Question 3: How do we build a Proof of Space?

Stacked-DRGs Graph (aka “SDR")

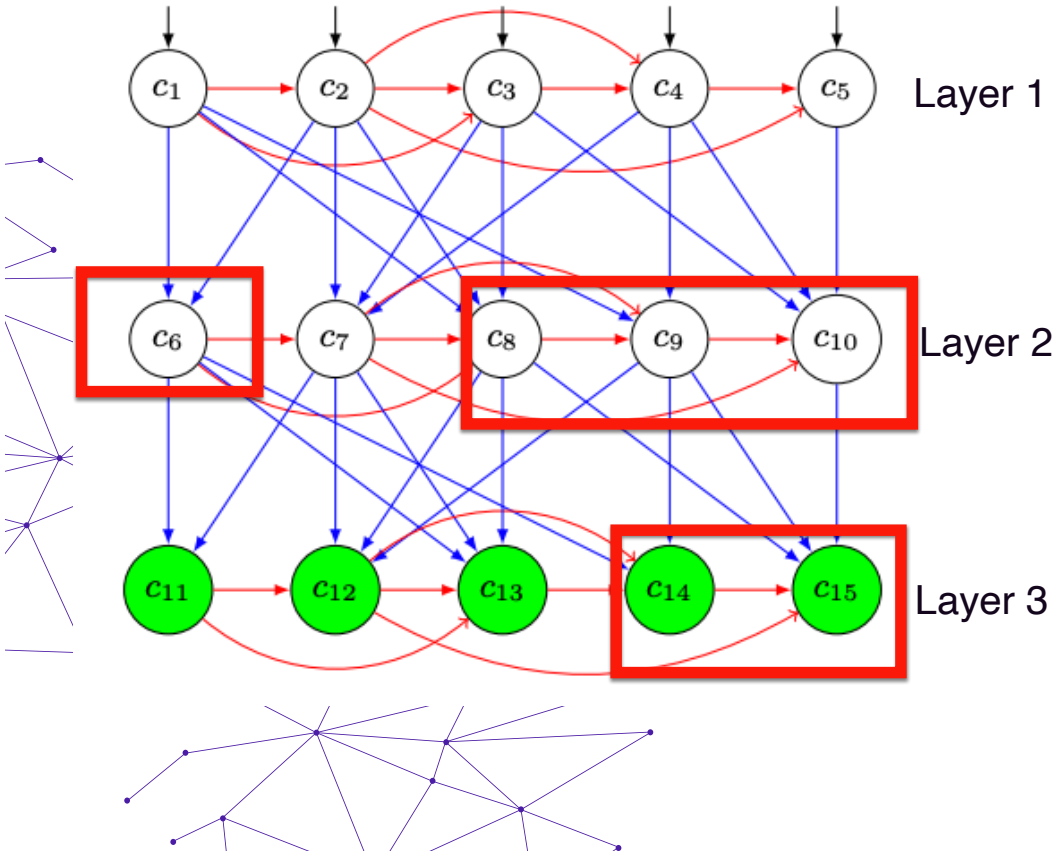


Malicious Alice:

Store only c11, c12, c13
(Delete c14, c15)

Question 3: How do we build a Proof of Space?

Stacked-DRGs Graph (aka “SDR")



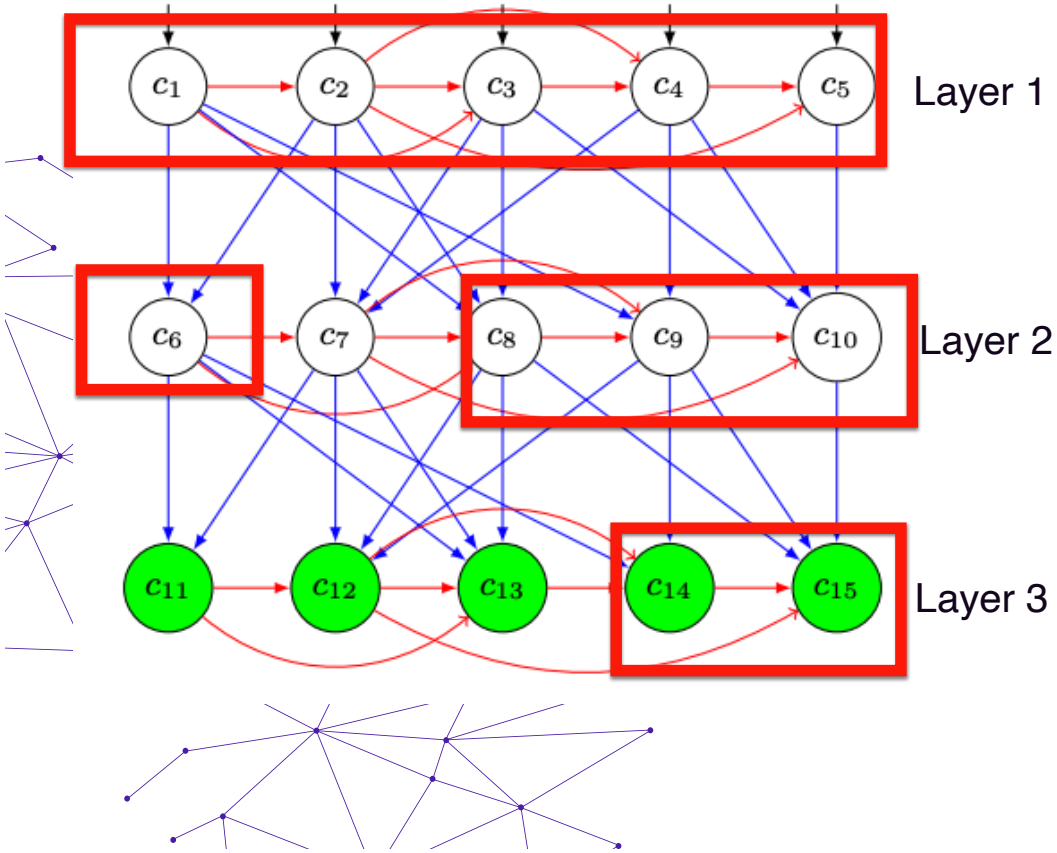
Security argument:

Recomputing c14, c15

- 4 parents in layer 2 (because of the expander graph)

Graph-labelling based PoS

Stacked-DRGs graph (aka “SDR”)



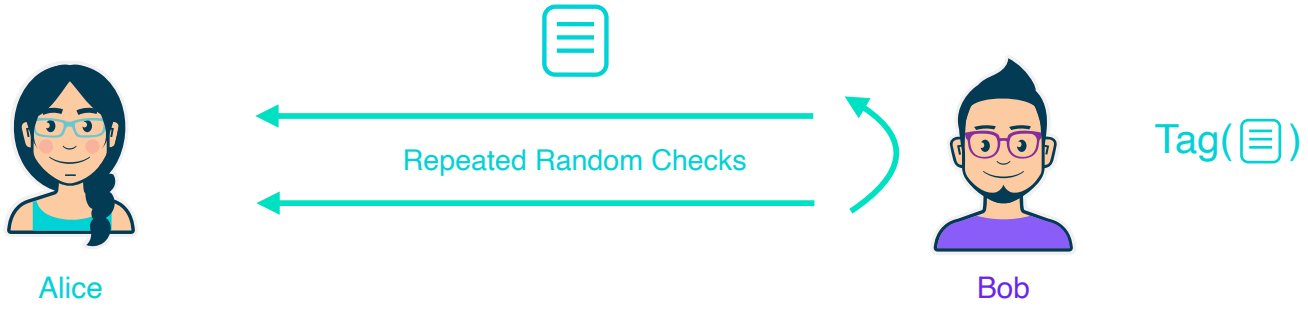
Security argument:

Recomputing c14, c15
- 80% of the final layer => long path!!
(because of the depth-robust graph)

Question 3: How do we use PoS for proving Persistent Storage?

From Persistent Storage to Useful Storage

First Attempt:
Proof of Retrievability



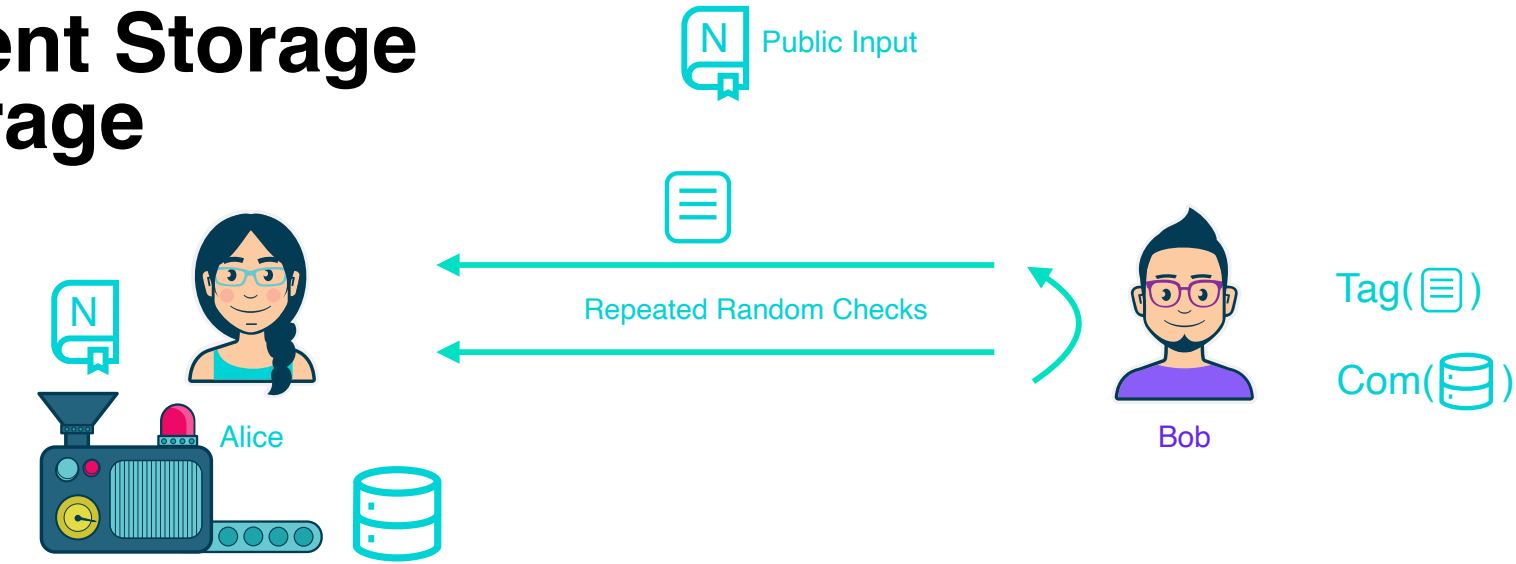
Problem 1: Data can be compressible! Alice can store instead of

Problem 2: We do not know a priori that Alice and Bob are two different persons. What if Alice asks herself to store a string of zeros?

We need something else!

Question 3: How do we use PoS for proving Persistent Storage?

From Persistent Storage to Useful Storage



Second Attempt:
Proof of Space +
Proof of Retrievability

Problem 1: How do we know that the advice and the data are linked together?

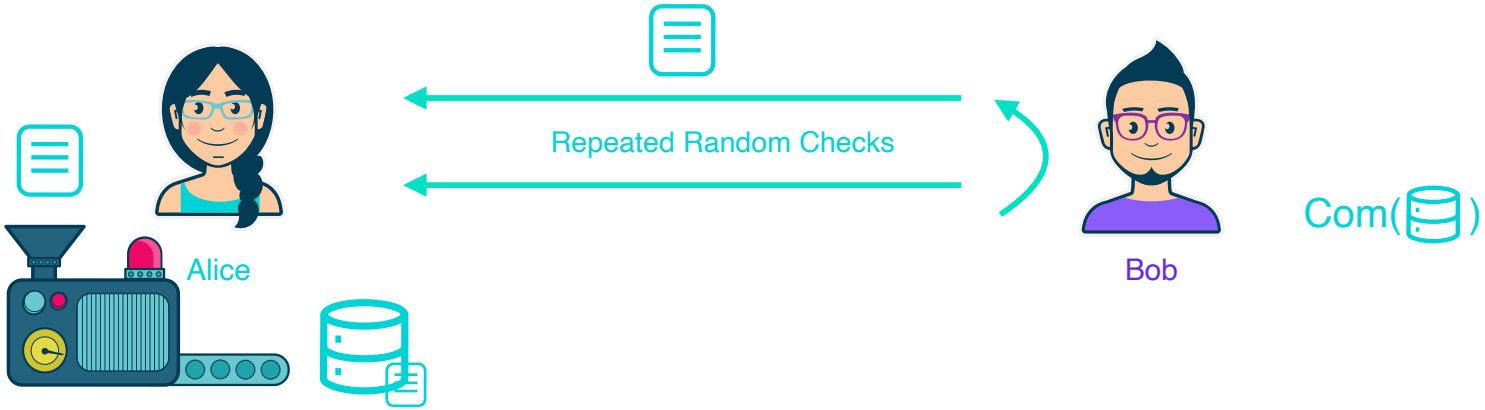
Problem 2: We do not know a priori that Alice and Bob are two different persons.
What if Alice asks herself to store a string of zeros?

We need something else!

Question 3: How do we use PoS for proving Persistent Storage?

From Persistent Storage to Useful Storage

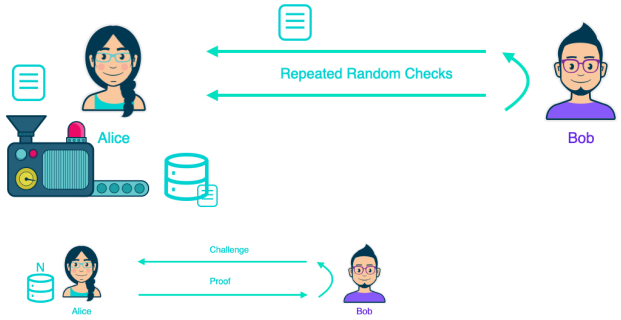
Third attempt
Proof of Space and
Proof of Retrievability



This approach works!

Question 3: How do we use PoS for proving Persistent Storage?

Adding Storage



1. Take some data **D** of size N 

2. Compute **com(D)** and use it as public input for computing the advice **A** through initialization 

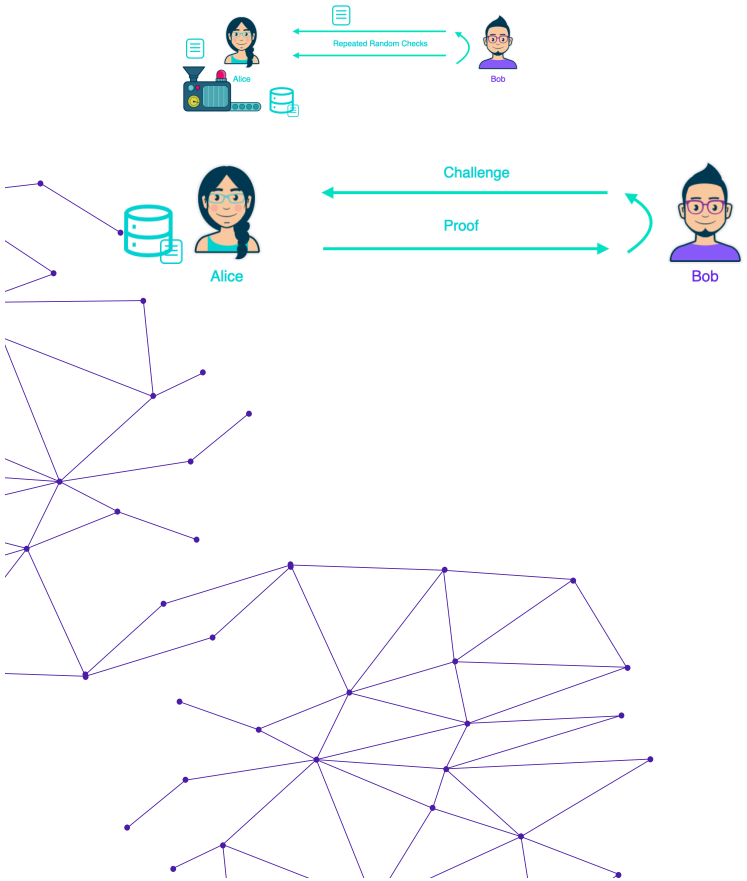
3. Prove that the advice was correctly generated
4. XOR the advice with data **D** 

5. Store **D**xor**A**
6. Commit to **D**xor**A** and **D** onchain 

What do we have? After this step data is encoded in an incompressible way

Question 3: How do we use PoS for proving Persistent Storage?

Auditing Storage



1. **Bob** asks challenges
2. If **Alice** successfully replies with a proof in less than T it means she is storing **D**xor**A** 

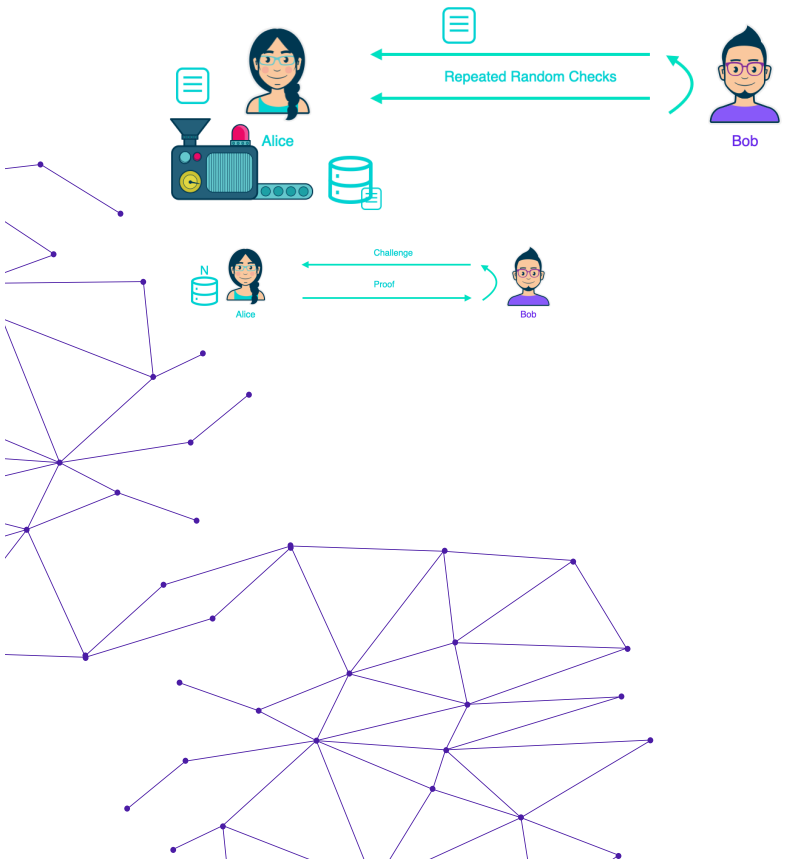
3. Proof goes onchain

What do we have? After each audit we know Alice keeps storing an incompressible encoding of the data

What about data retrieval?

Question 3: How do we use PoS for proving Persistent Storage?

Data Retrieval



1. Use **com(D)** as public input for computing the advice **A** re-running initialization
2. XOR the advice **A** with the incompressible encoding
3. Retireve data **D**

XOR XOR

What do we have? Alice can always retrive data re-running initialization

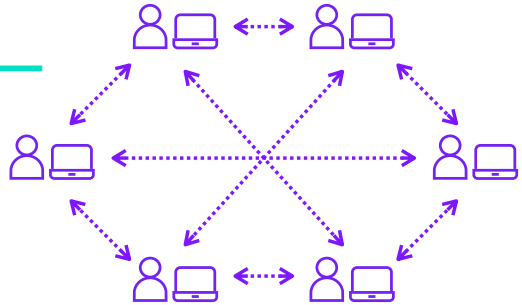
PoS in Filecoin

PoS Initialization = miner onboarding storage (aka “PoRep”)

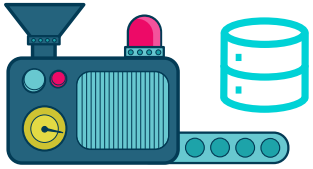
Initialization (1 replica produced) = N bytes of storage committed to Filecoin

MINER

Data D



Blockchain



Advice R
(called **Replica** in
Filecoin)

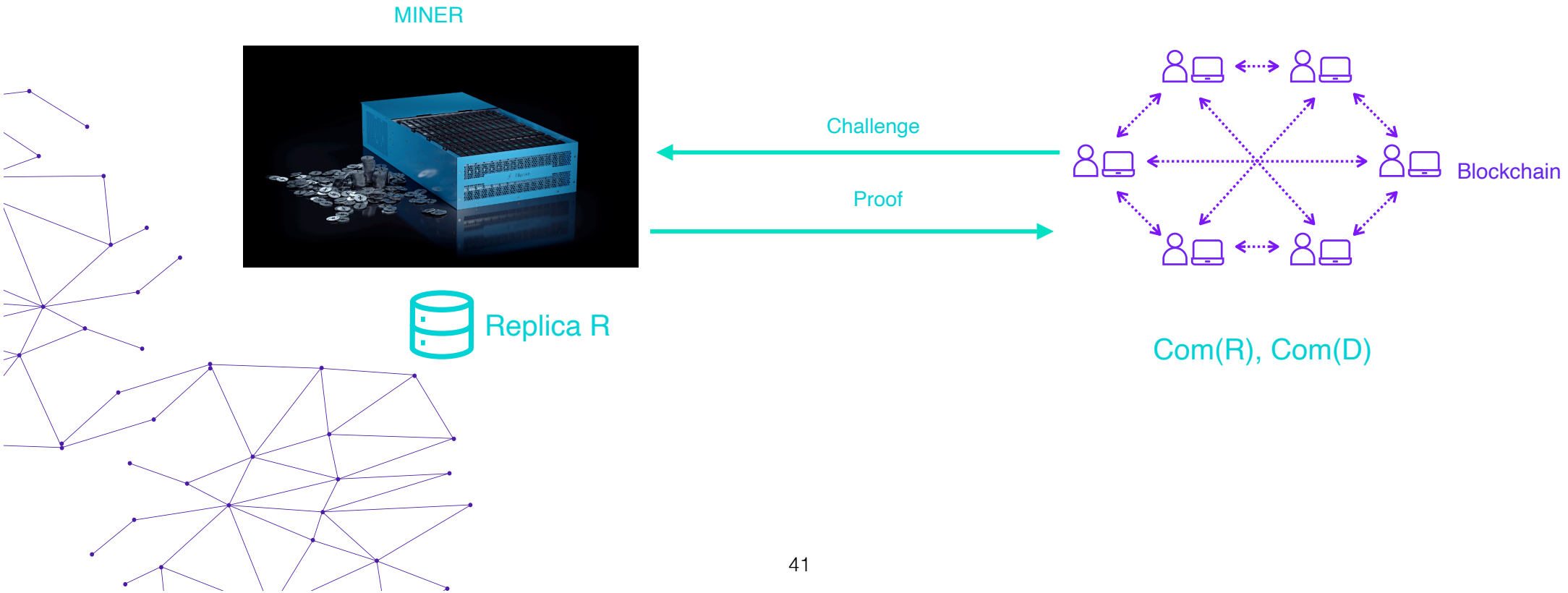
Comm(R) = MT root on R

Comm(D) = MT root on D

PoS in Filecoin

PoS Execution (aka “WindowPoSt”)

The blockchain verifies the committed storage every 24



PoS in Filecoin

Filecoin is live! Mainnet launched in Oct 2020

PoS in Filecoin

Some numbers (Oct 2021):

~ 3358 miners

~ 360M replicas

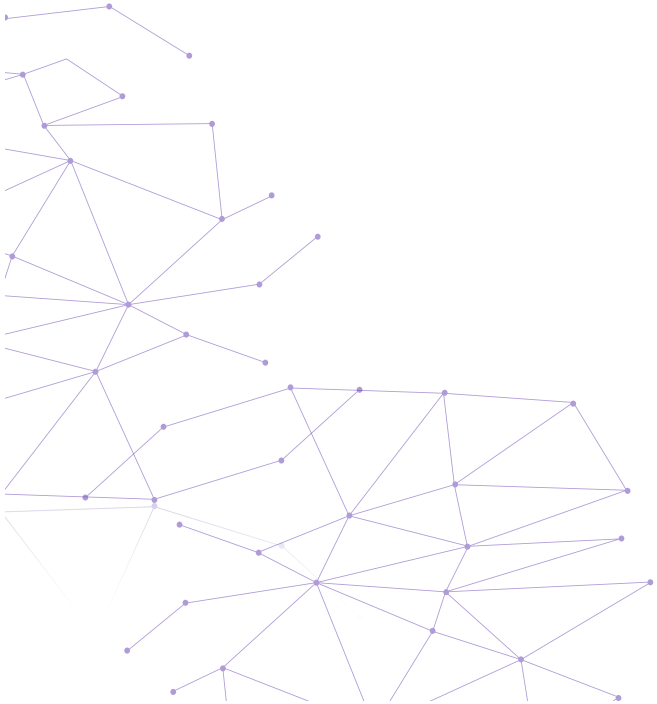
Each replica = 32 GiB (or 64GiB)
Total of ~ 11.9 EiB (~ 8 * 2^(60) bytes)

https://filfox.info/en

How does Filecoin use PoS?

Is that all? Nope….

- Each challenge requires a proof to Com(R) (ie, a Merkle Tree path) on chain
- Number of proofs that need to go on chain is massive….*not feasible!*
- How can we reduce on-chain footprint?



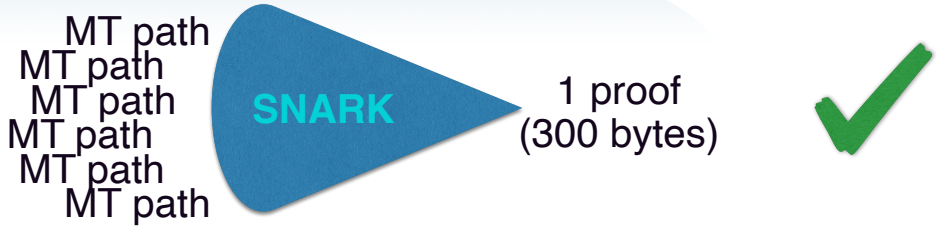
How does Filecoin use PoS?

Is that all? Nope….

- Each challenge requires a proof to Com(R) (ie, a Merkle Tree path) on chain
- Number of proofs that need to go on chain is massive….*not feasible!*
- How can we reduce on-chain footprint?

We can use **SNARKs** (*Succinct Non interactive Arguments of Knowledge)*

A cryptographic tools that gives
a **constant size proof** of complex statements.



Thank you!

Questions?