Exploring Proof of Space with Hard-to-Pebble Graphs

Vivek Bhupatiraju, John Kuszmaul, Vinjai Vale
Mentors: Ling Ren, Albert Kwon

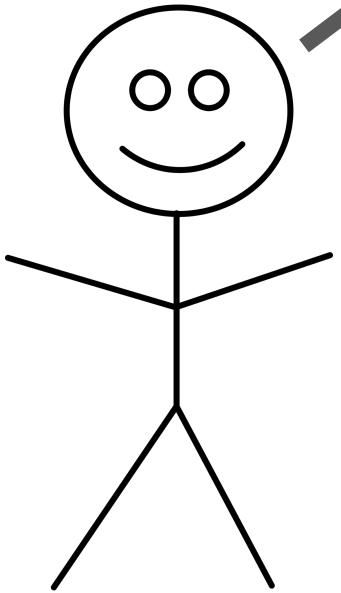
Proof of Work

- Involves a prover and a verifier.
- A puzzle that is hard to solve. Lots of computational power is necessary.
- Easy to check.
- Originated in the 90s.

Proof of Work

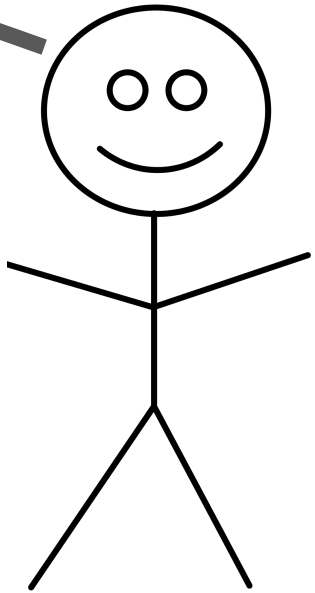
"Please solve this problem: *problem"*

Verifier



"Will do!"

Prover



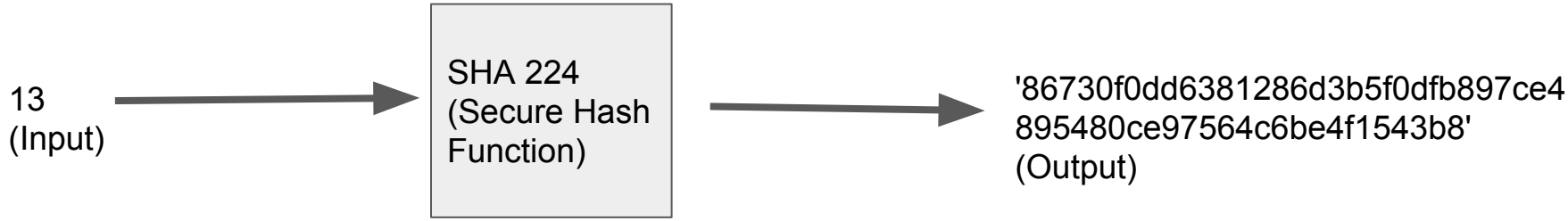
Uses a lot of computing power to solve *problem*

Uses a tiny amount of computing power to check answer

"I got it! Here's the answer: *answer"*

"Nice job! You got it right!"

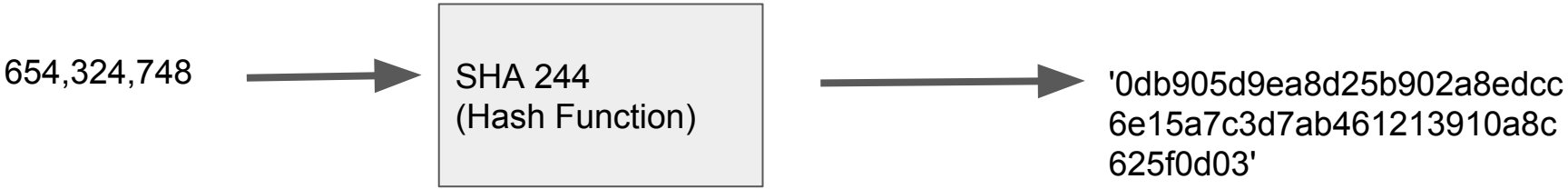
What are Secure Hashes?



- Most hashes are fast to compute.
- Hard to find the input given the output. The only known method is guess and check.

How are Hashes useful for Proof of Work?

- Alice tells Bob that she hashed a natural number less than or equal to 10^9.



- Alice tells Bob the output.
- Bob will have to calculate an expected 500,000,000 hashes to find Alice's input.
- Alice can check the Bob's result almost instantly.

Proof of Work

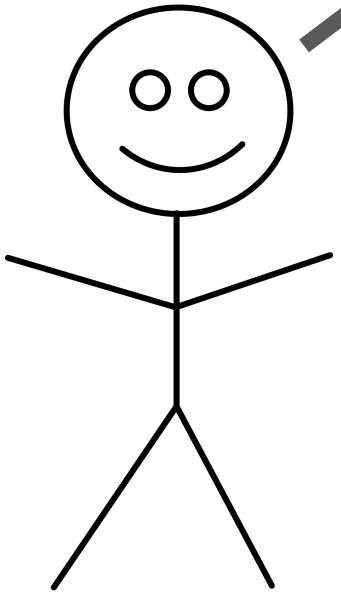
- Uses
  - Preventing spamming on email
  - Stopping double spending on Bitcoin

Proof of Space

- Involves a prover and a verifier.
- Proving you have devoted space instead of computation time.
- Proofs of Space are only a few years old.
- Easy to check.

Proof of Space

Verifier



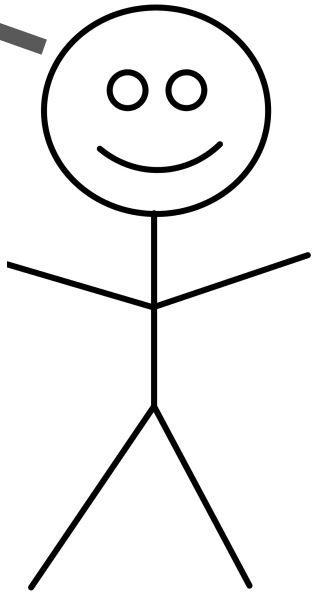
"Please store this data: *data*"

Checks the proof from the prover using very little space

"Good job!"

"Will do!"

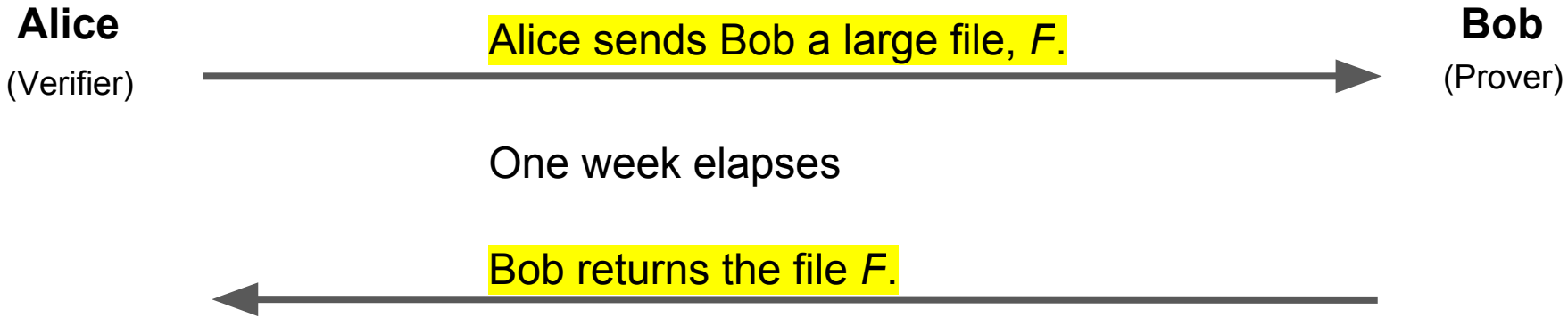
Prover



Uses a lot of disk space to store *data*

"Here is proof that I stored the data: *proof*"

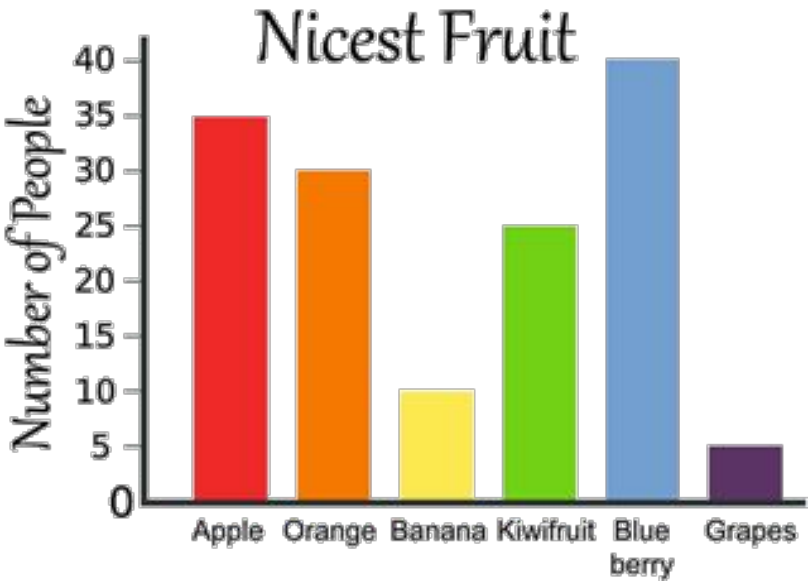
Trivial Proof of Space



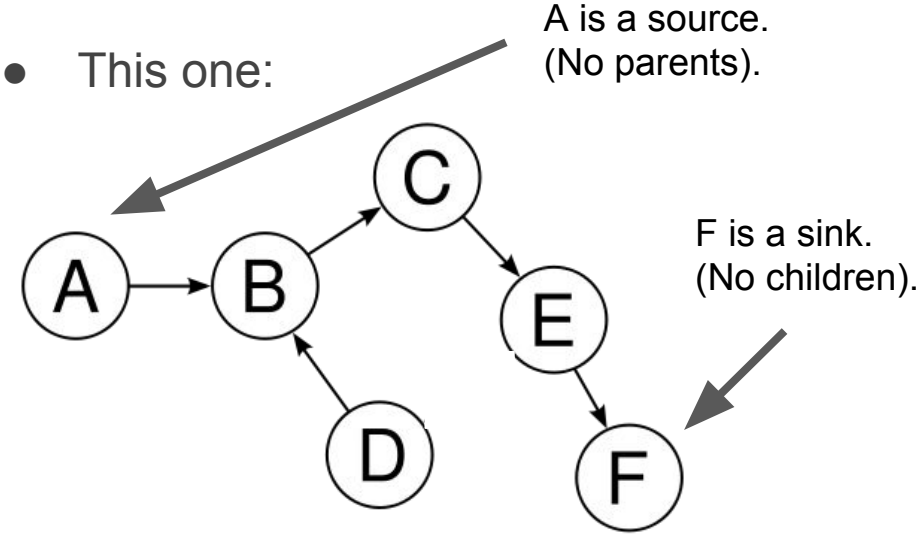
- Alice uses too much space
- Sending the file is impractical

Exploring Proof of Space with Hard-to-Pebble Graphs

- Not this Kind of Graph:



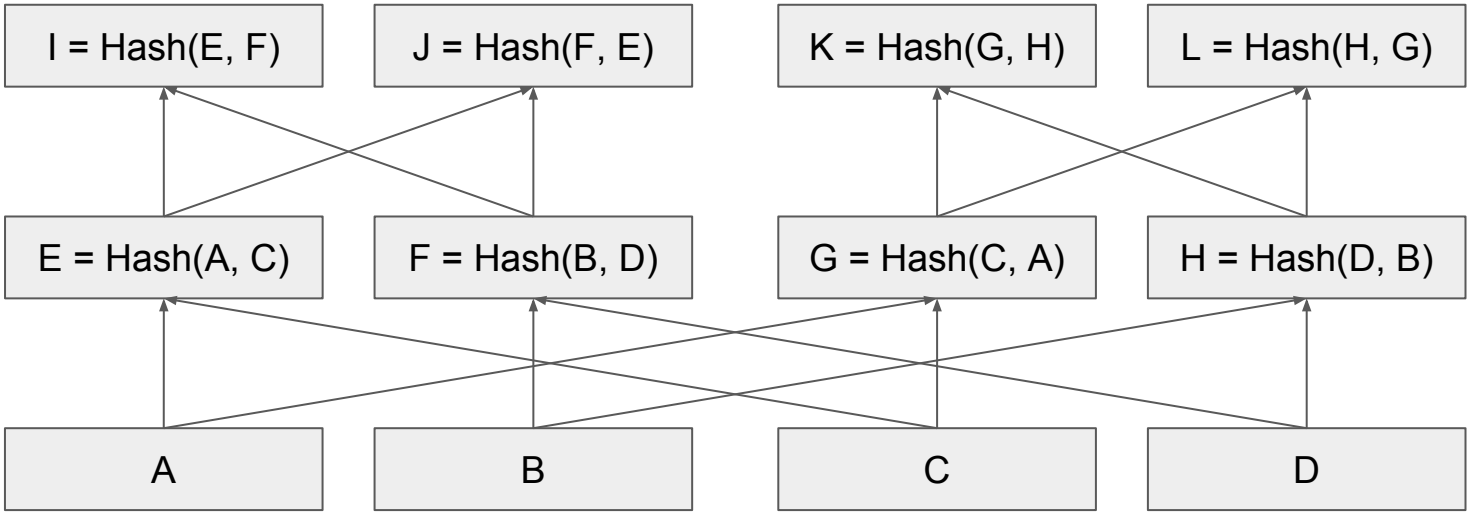
- This one:



- Graphs have vertices and edges.
- We're interested in directed acyclic graphs.

Computing Hashes on Graphs

- We compute and store the value of a vertex by hashing its parents

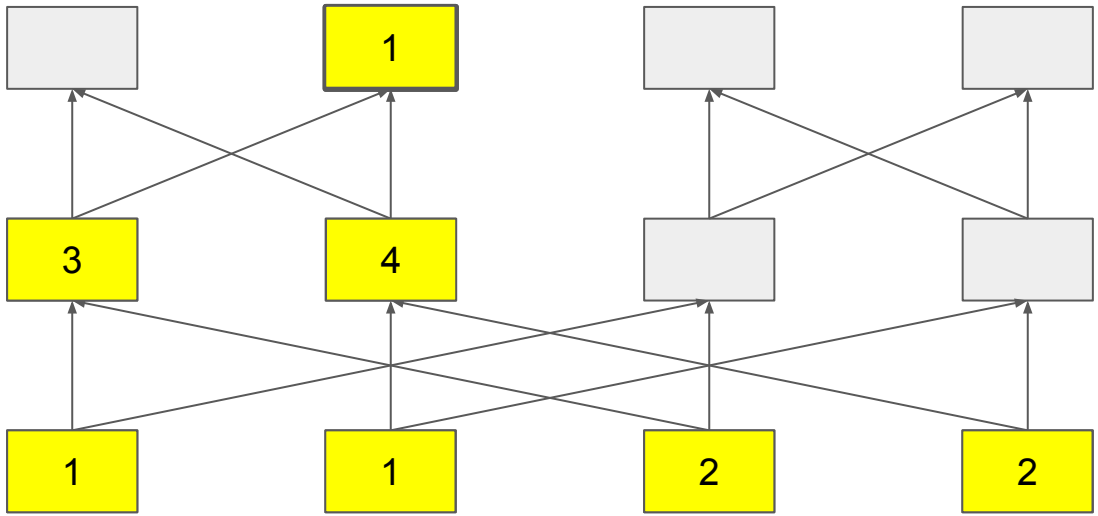


The Pebbling Game

- You can pebble a vertex only if all of its parents have been pebbled
- This means that the sources can be pebbled at any time
- Pebbling a vertex is storing the hash of its parents
- Removing a pebble is freeing that memory

Playing the Game

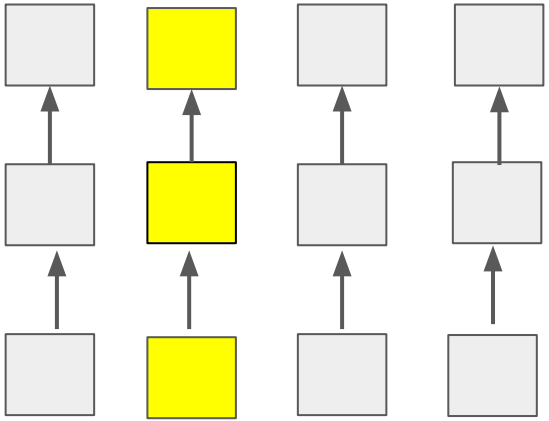
Suppose we want to pebble this vertex here.



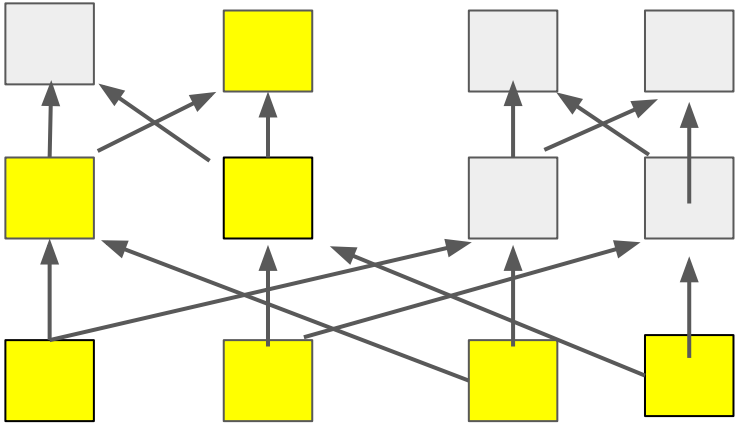
We can do it with four pebbles:

Hard-to-Pebble Graphs

Graphs are hard to pebble if you need many pebbles to pebble a vertex (high interconnectedness)



Not Very Hard to Pebble;
Low interconnectedness
(Few edges)



Harder to Pebble;
High interconnectedness
(Many edges)

A PTC Graph

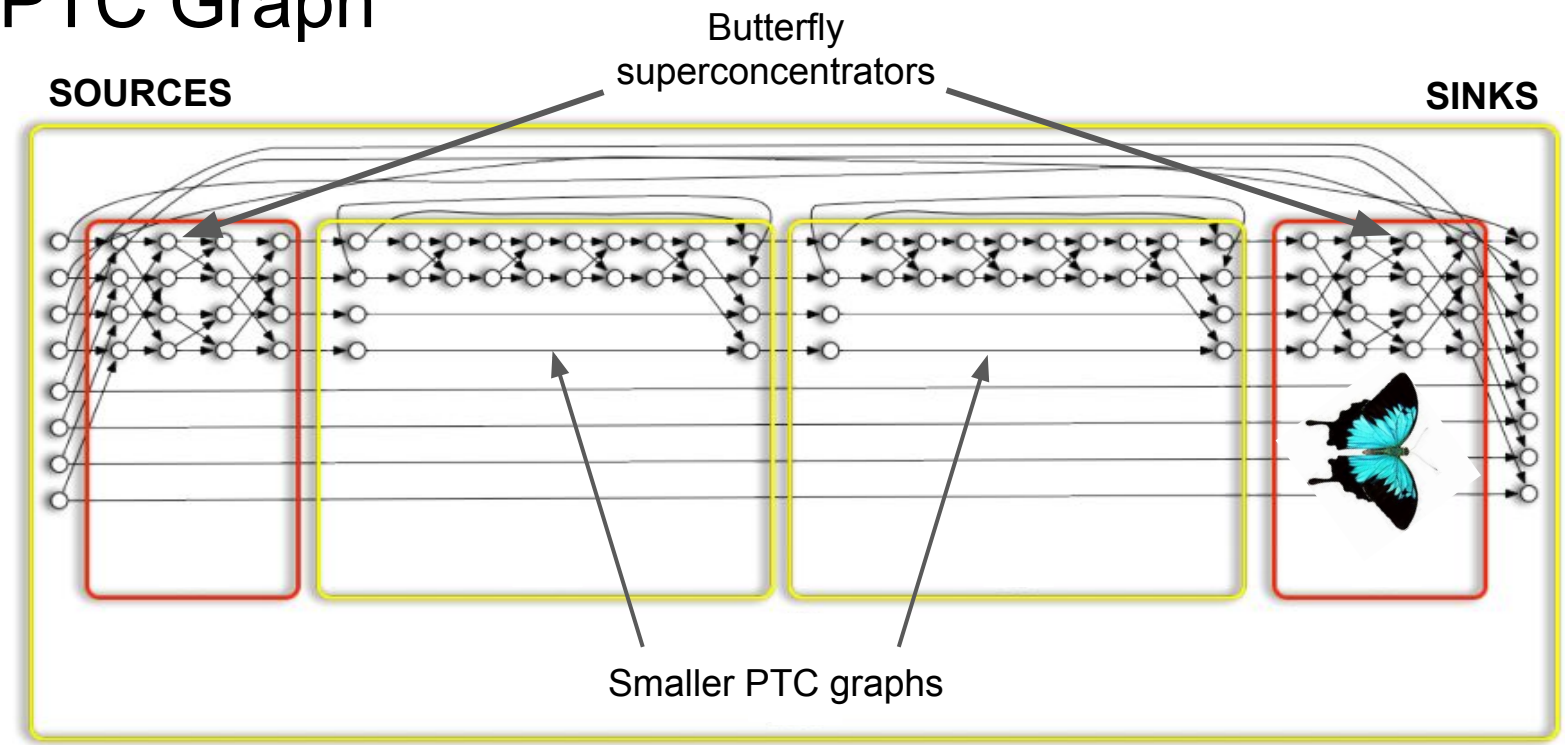


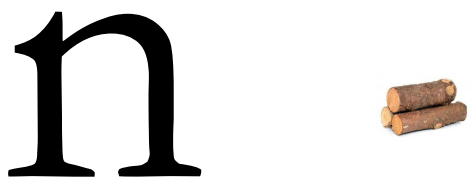
Image adapted from Nicolaos P. Karvelas' master thesis, "Proofs of Secure Erasure"

The PTC Graph

Paul, Tarjan, and Celoni proved that any pebbling strategy needs at least

$$c \cdot \frac{n}{\log n}$$

pebbles, where c is some constant factor and n is the number of vertices in the graph.



Source: Wolfgang J. Paul, Robert E. Tarjan, James R. Celoni, “Space Bounds for a Game on Graphs.” In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, STOC, pages 149–160. ACM, 1976.

Naive Prover/Verifier Interface

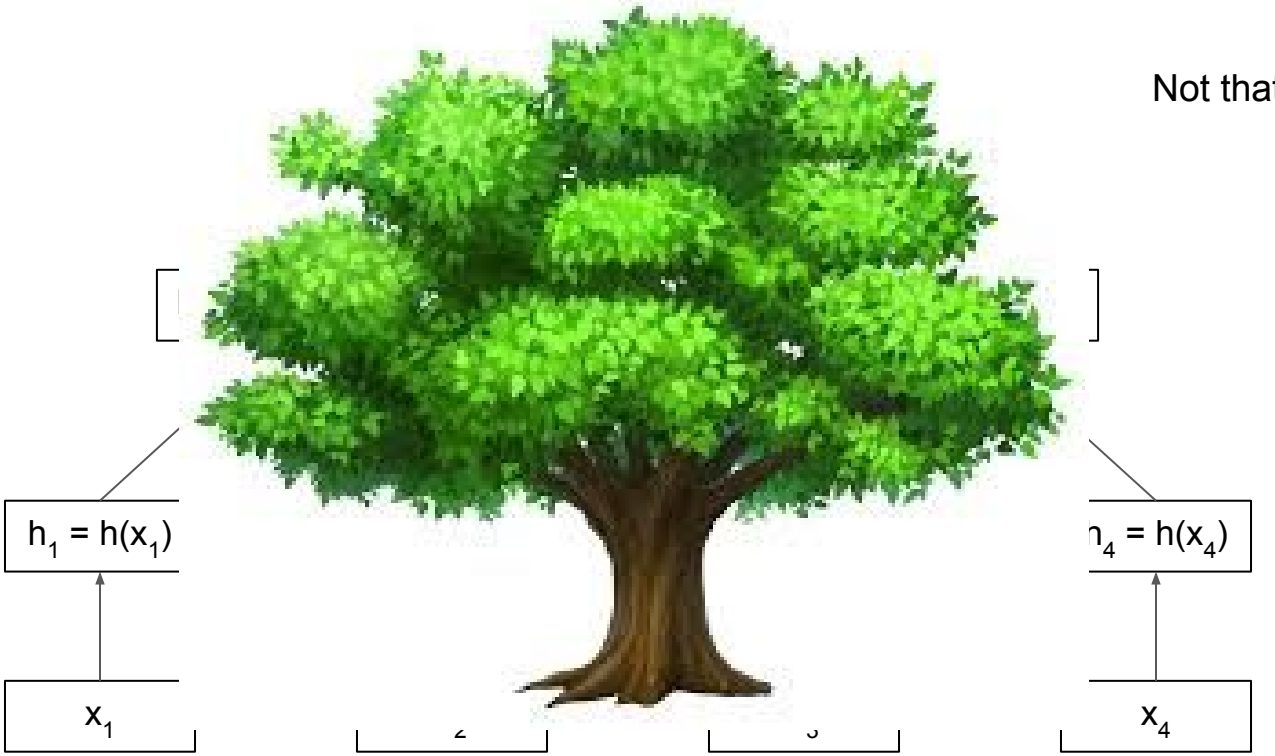
- V gives P:
  - Parameters for generating the PTC graph
  - Values for the sources of the graph
- Both P and V compute the values of all the vertices in the graph
- V asks P for her values for each vertex and checks them against his own
- If they are all good, then P passes, otherwise P fails

Downside to Naive Interface

- Sending over every single vertex is very inefficient
- Many applications require efficient and quick verification (similar to Proof of Work)

How can we construct an efficient verifying scheme?

Merkle Tree



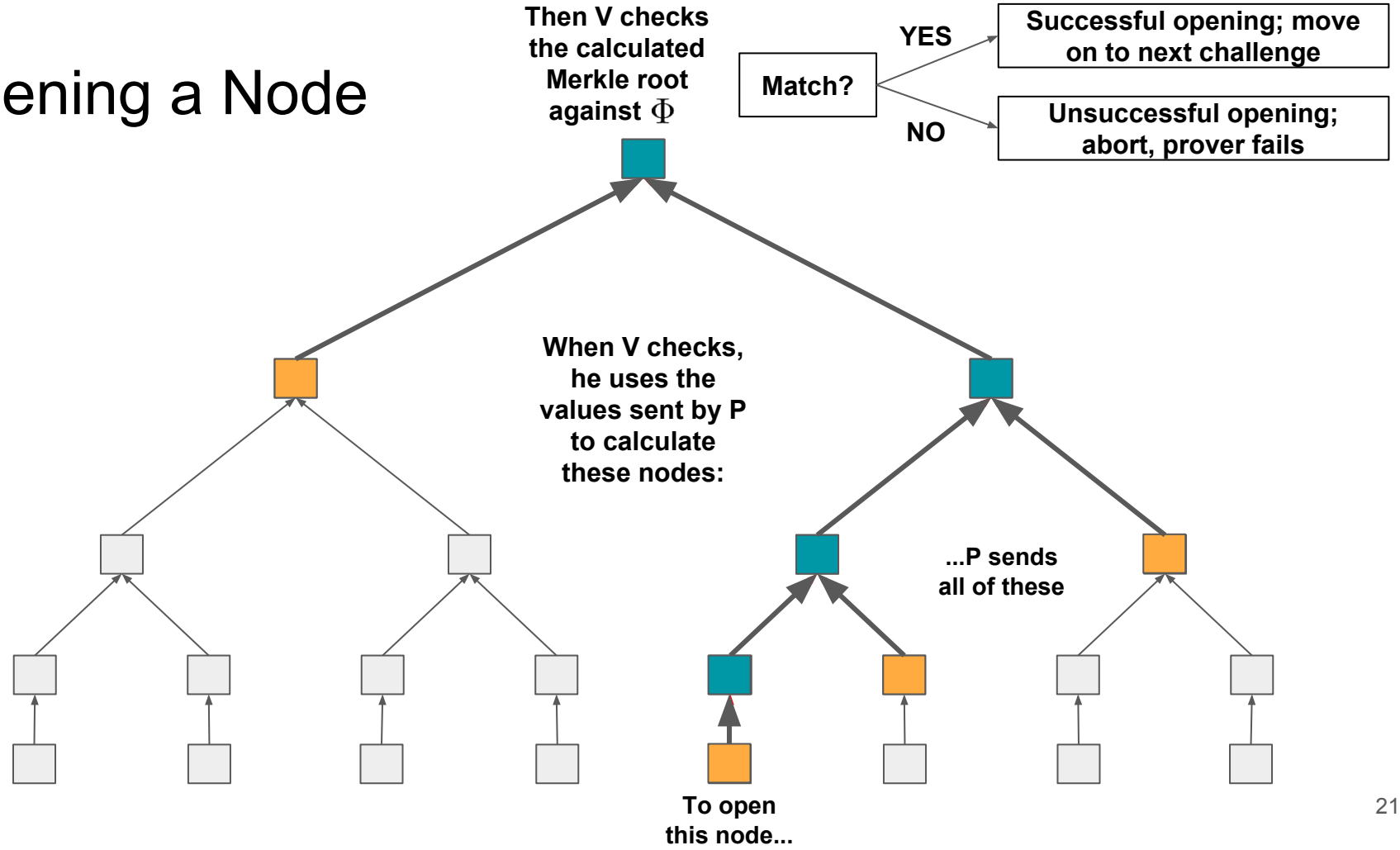
Not that kind of tree!

Using Merkle Trees

P can now distill the values of all of her nodes down to the Merkle root Φ

To check the value of a particular node, V will ask **P** to “open” the node under the Merkle root

Opening a Node



Opening a Node

- But how does V verify the root itself?
- V will open sources, and make sure that they match the small number of initial values that he sent over
- It turns out that you can prove that P only needs to open a small number of nodes in order for V to be convinced that P is sufficiently honest

Efficient Pebbling

Why do we even care about efficiency?

- Important to figure out most optimal strategy
- Otherwise, those who do have unfair advantage

Efficient Pebbling: A Space/Time Tradeoff

Naive method: start by pebbling the sources, then the next layer, then the next layer, etc.

This is fast but space-inefficient because we never remove unnecessary pebbles, so we end up having to store the value of every single vertex the entire time

What if every time we pebble the next layer of the graph we remove the pebbles of the previous layer?

It turns out this algorithm is indeed better in terms of storage.

Summary

To recap, we talked about:

- Proof of Work and its problems… so we turn to Proof of Space
- Hard-to-pebble graphs, specifically the PTC graph and the $c \cdot \frac{n}{\log n}$ bound
- Using Merkle trees for efficient verification by checking root consistency
- Efficient pebbling algorithms

Current and Future Work

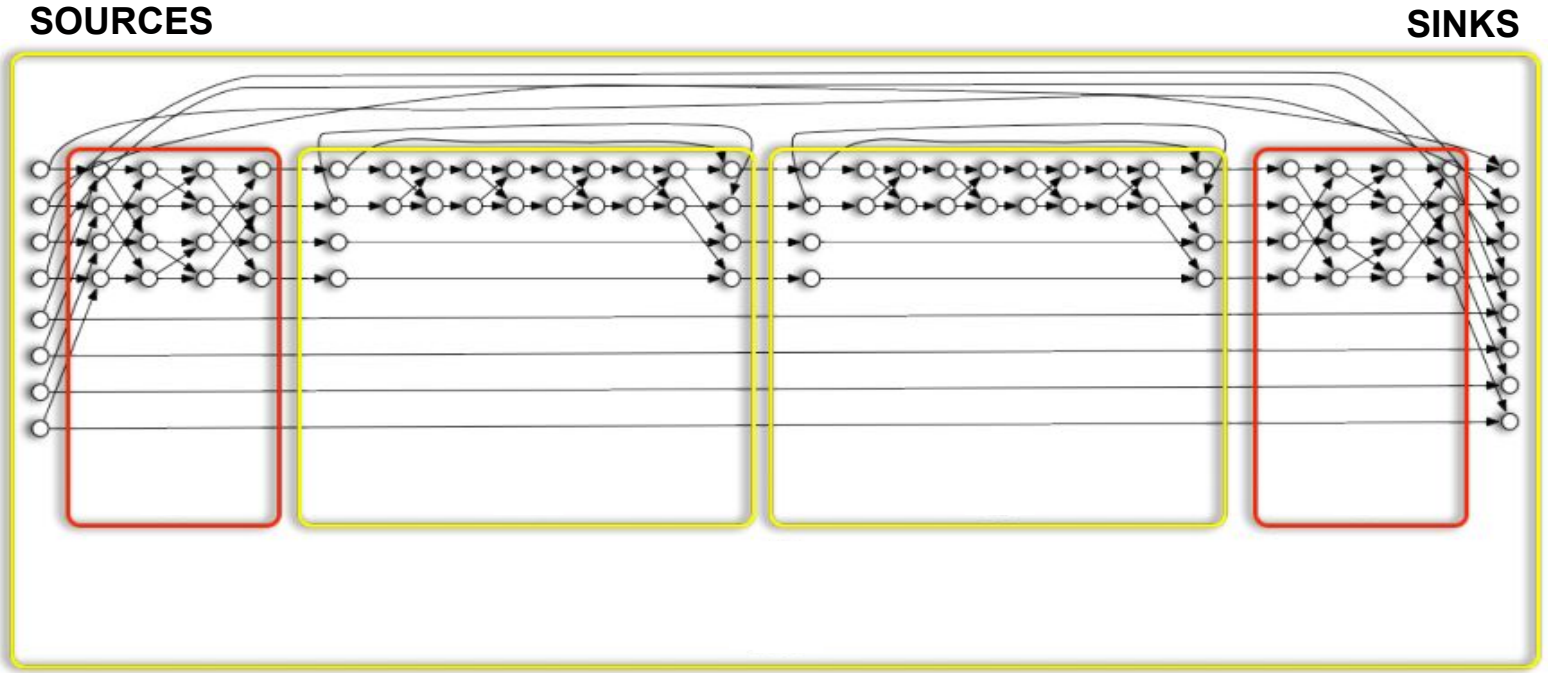
- Investigating the performance of other pebbling algorithms (e.g. depth-first pebbling)
- Building better hard-to-pebble graphs based on so-called “linear superconcentrators,” as opposed to the butterfly superconcentrators used in the PTC graph
- Build a consensus protocol (similar to Bitcoin) but with proof of space instead of proof of work

Thank You!

We would like to thank our mentors, Ling Ren and Albert Kwon, for their support and guidance. Without them and their expertise this project would not have been possible.

We also want to thank PRIMES for this wonderful opportunity, and you all for being such a great audience.

QUESTIONS?



References

1. Wolfgang J. Paul, Robert E. Tarjan, James R. Celoni, “Space Bounds for a Game on Graphs.” In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, STOC, pages 149–160. ACM, 1976.
2. Nikolaos P. Karvelas Master’s Thesis, “Proofs of Secure Erasure.”
3. Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak, “Proofs of Space,” 2013.