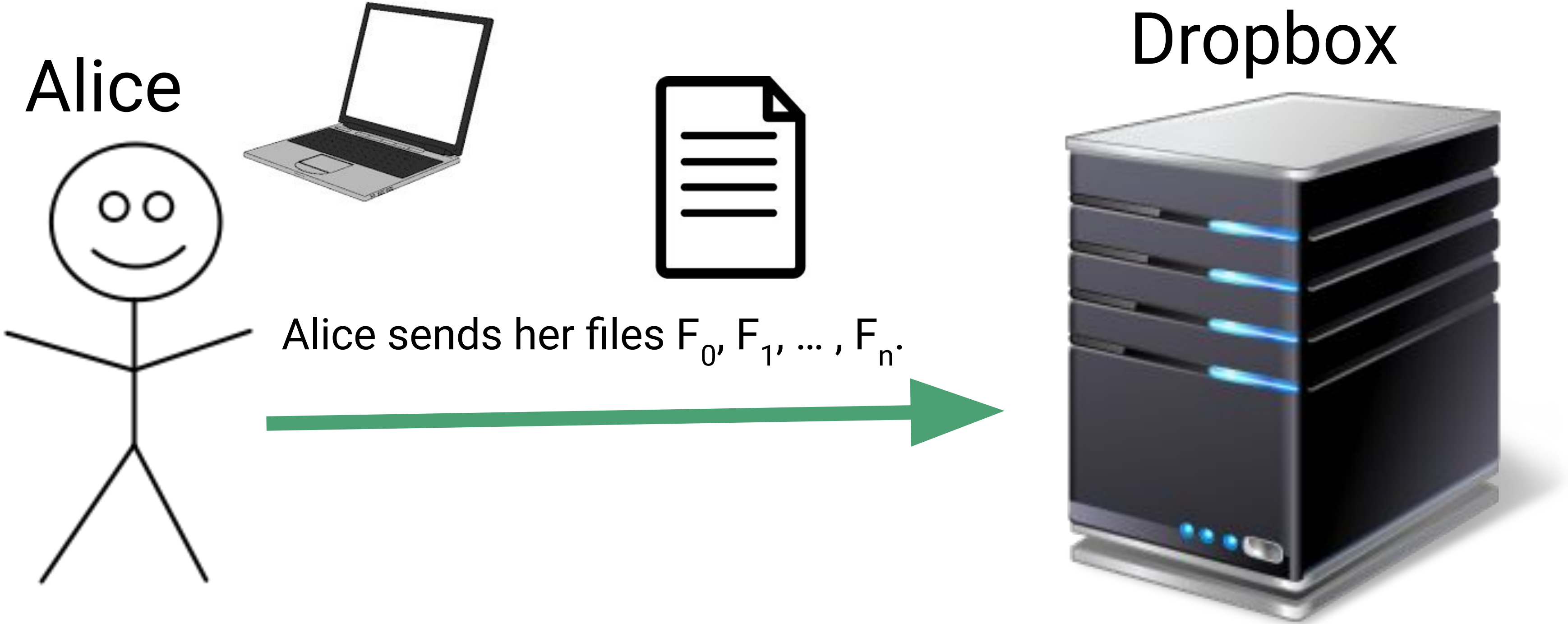
Verkle Trees:
Ver(y Short Mer)kle Trees

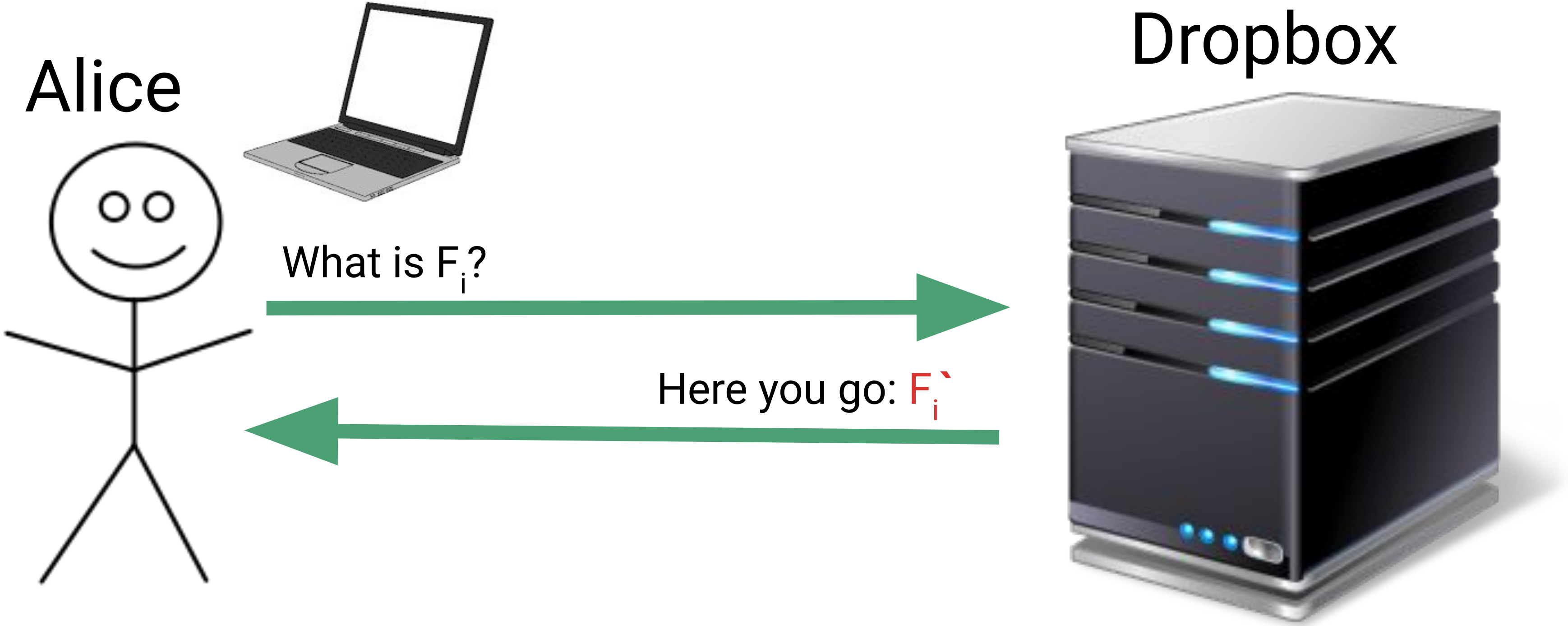
John Kuszmaul
Mentored by Alin Tomescu

PRIMES Conference - 5/19/2019

Storing Files Remotely

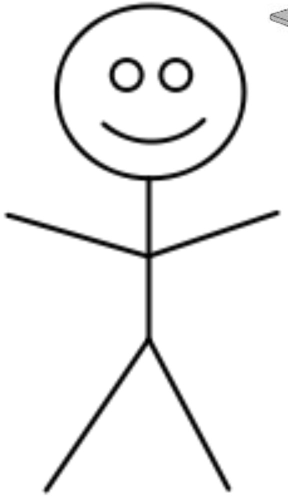


Storing Files Remotely



Proving/Verifying Integrity (or Correctness)

Alice

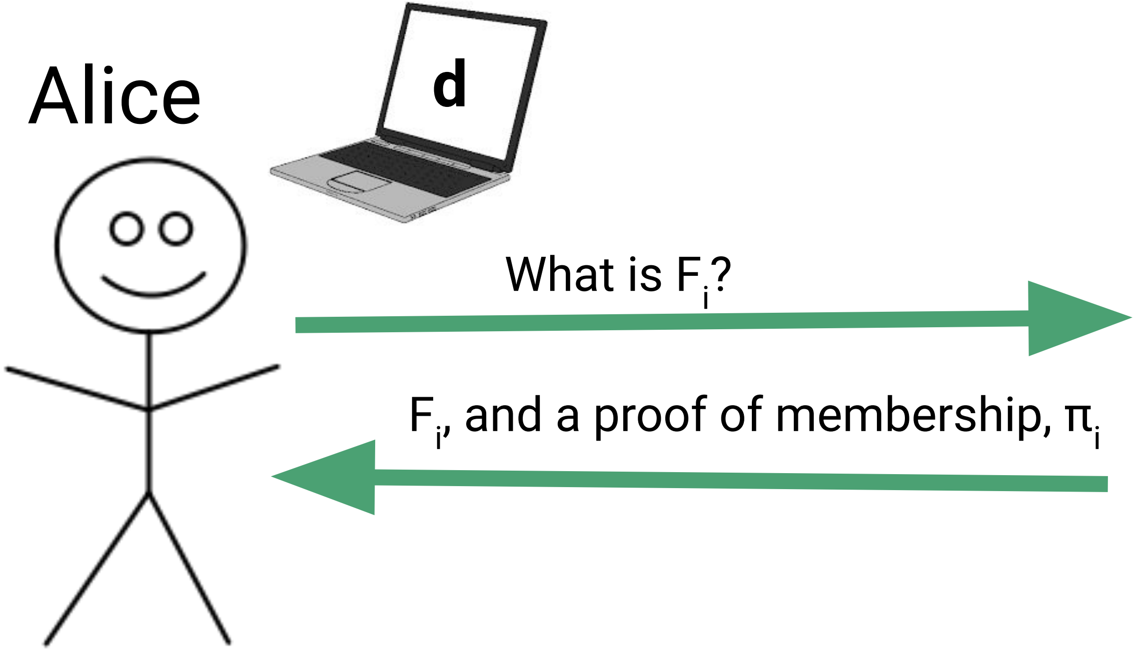


Alice sends her files $F_0, F_1, \dots , F_n$.

Dropbox

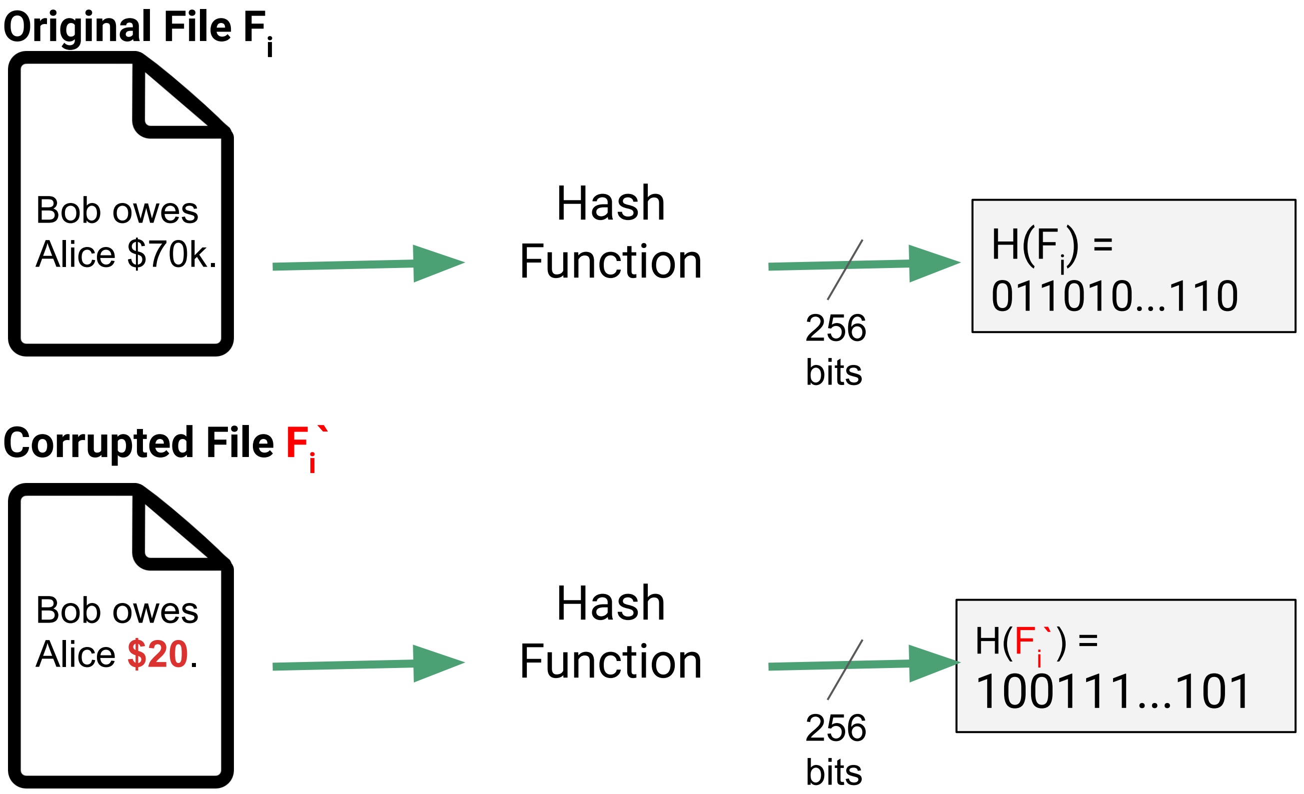
Alice generates a **digest** *d* of her files.

Proving/Verifying Integrity



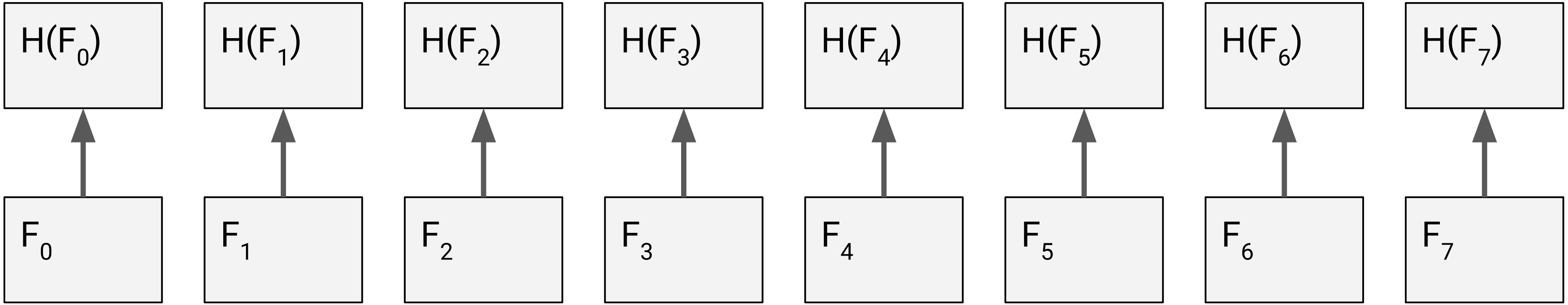
Alice verifies the proof π_i against *d* to make sure F_i has not been modified.

Secure Hash Functions

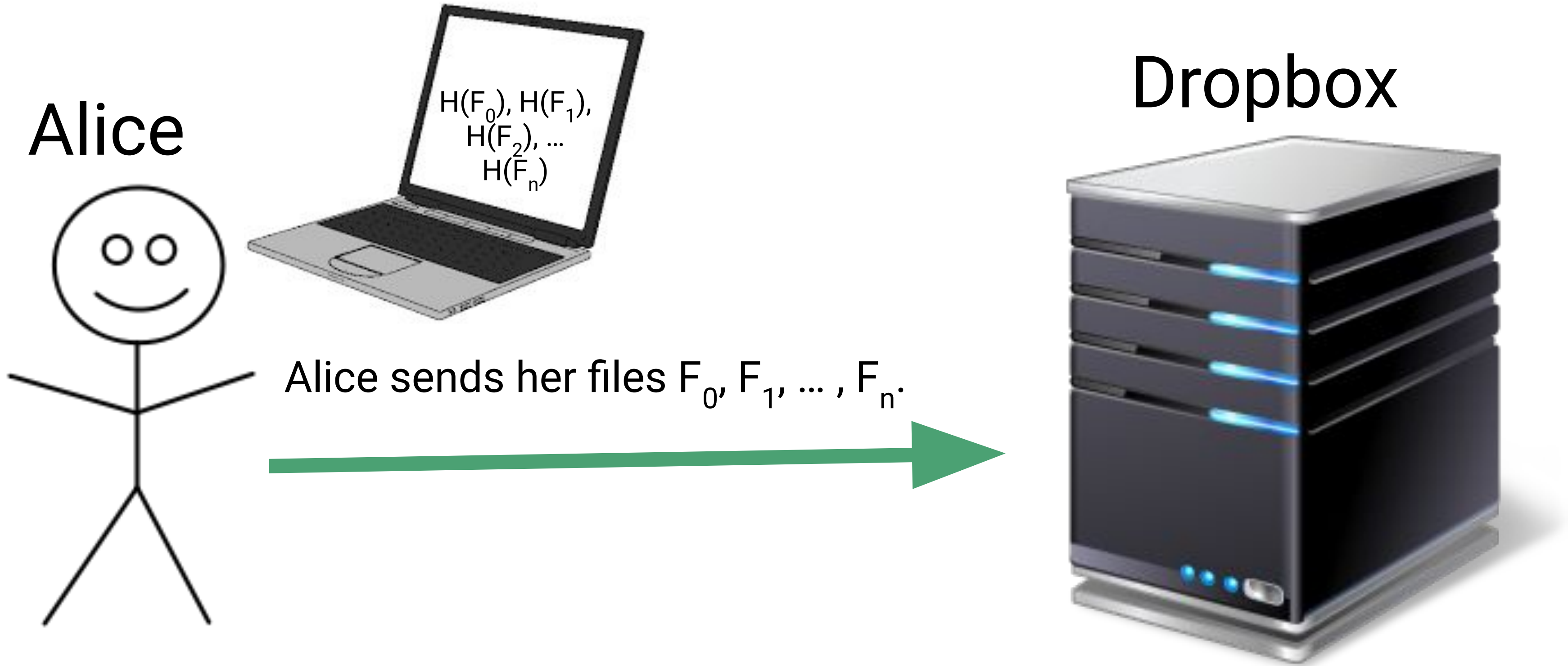


A Simple Scheme for Verifying File Integrity

Alice hashes each of her files:

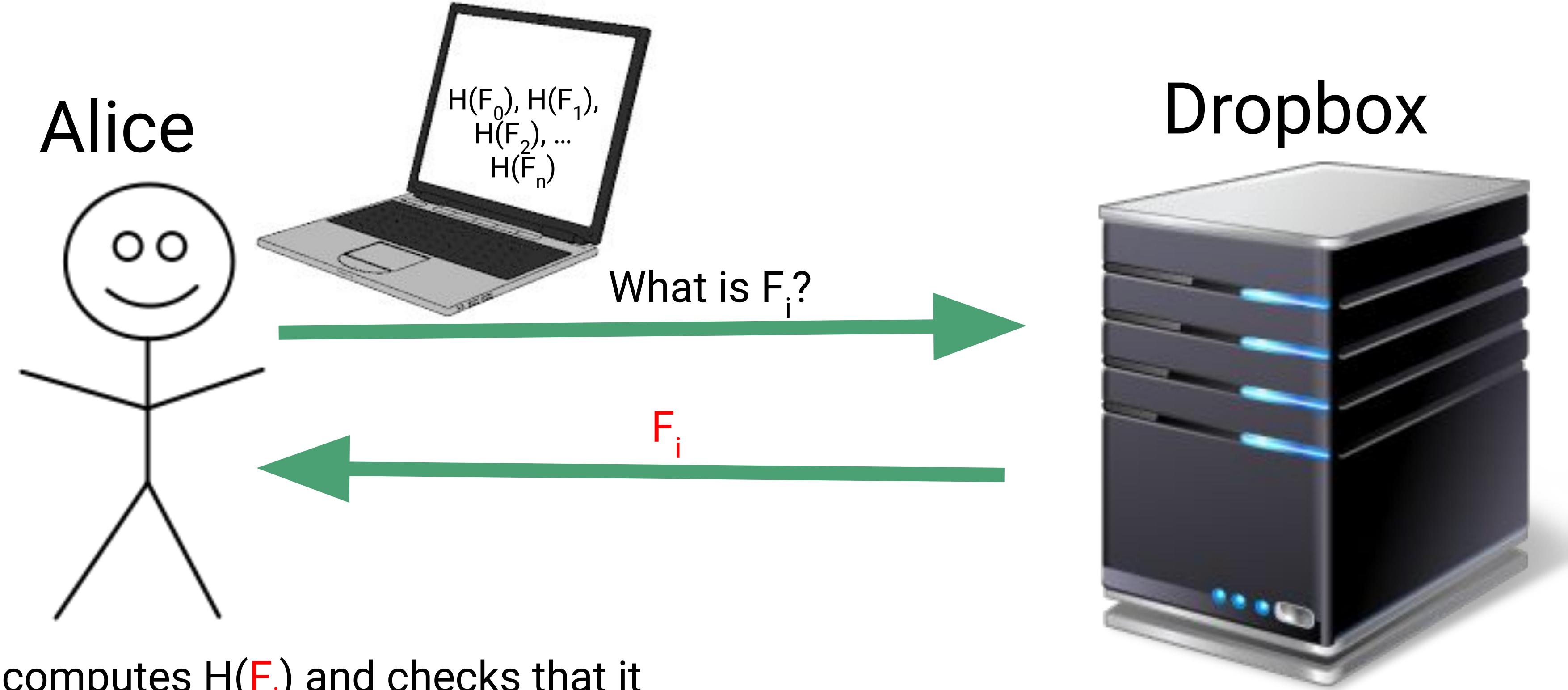


Proving/Verifying Integrity: Simple Scheme



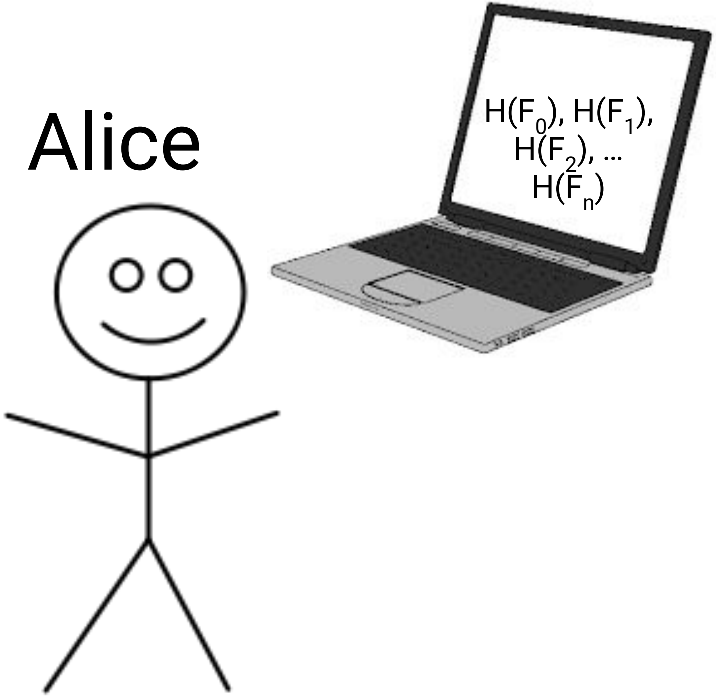
Alice computes and stores the hashes locally.

Proving/Verifying Integrity: Simple Scheme



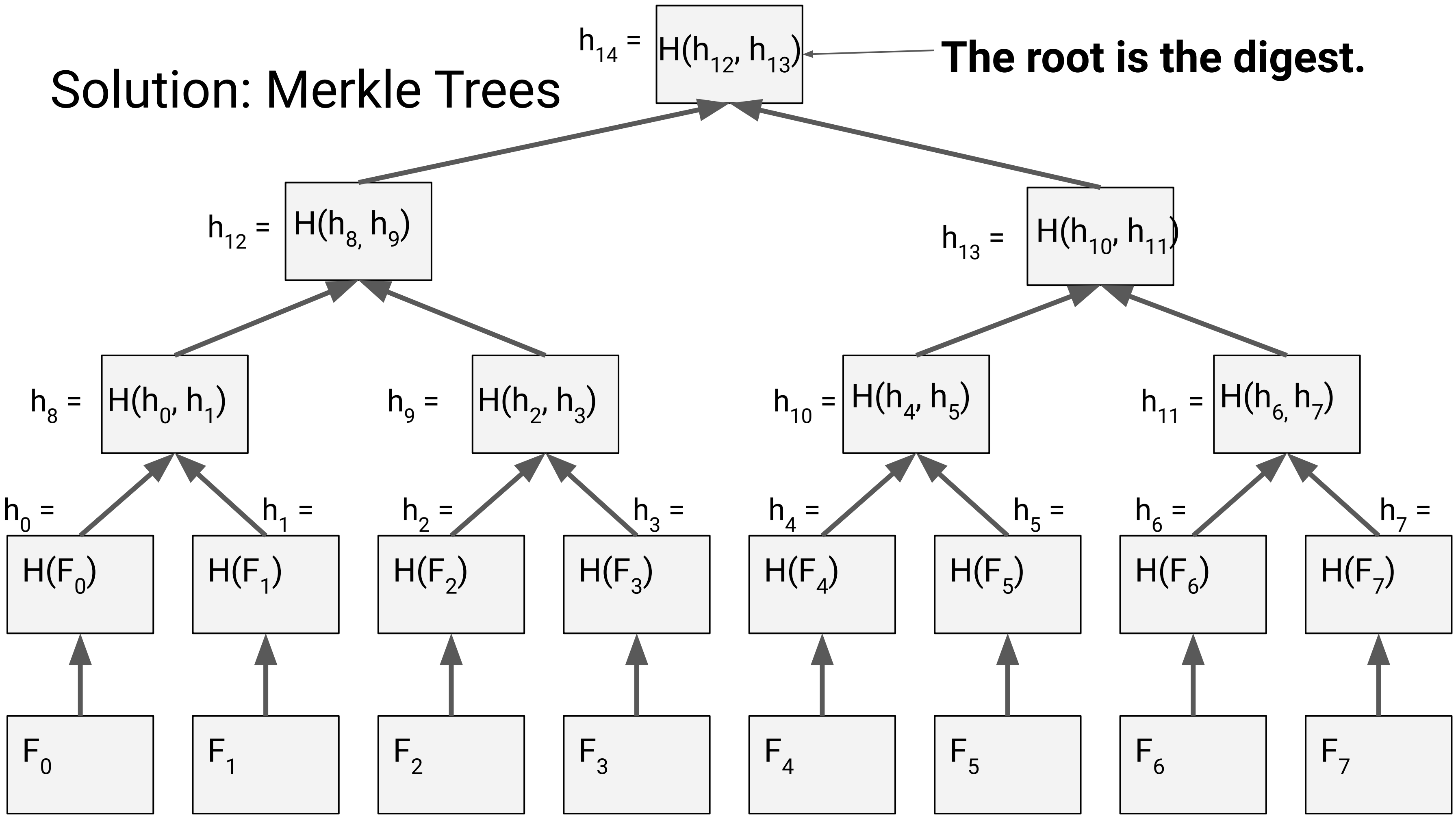
Alice computes $H(F_i)$ and checks that it equals stored $H(F_i)$.

Problem: Alice has to store n hashes.



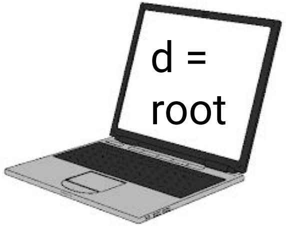
Alice's digest must be constant-sized.

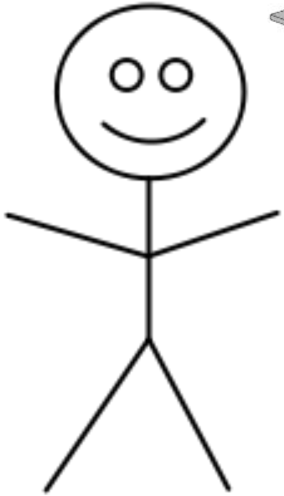
Solution: Merkle Trees



Proving/Verifying Integrity: Merkle Tree

Alice



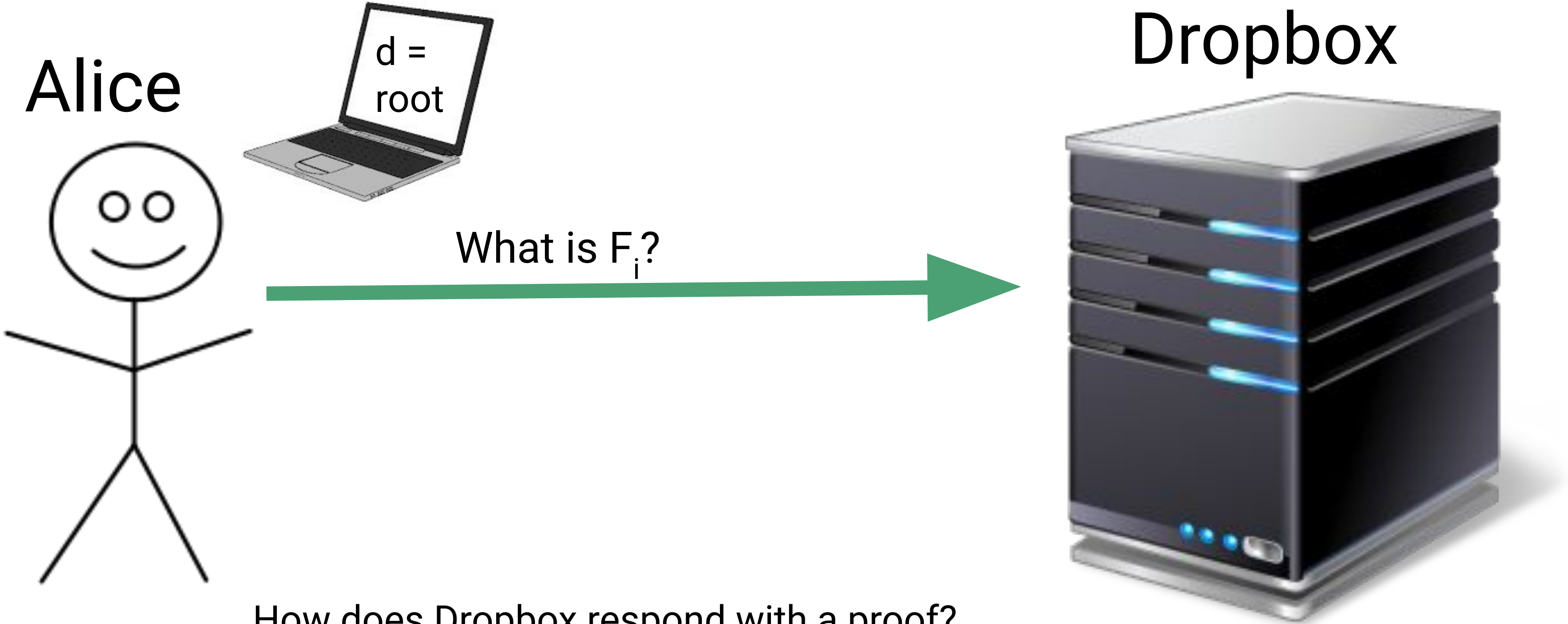


Dropbox

Alice sends her files $F_0, F_1, \dots, F_n$.

Alice computes the Merkle tree and stores the root locally.

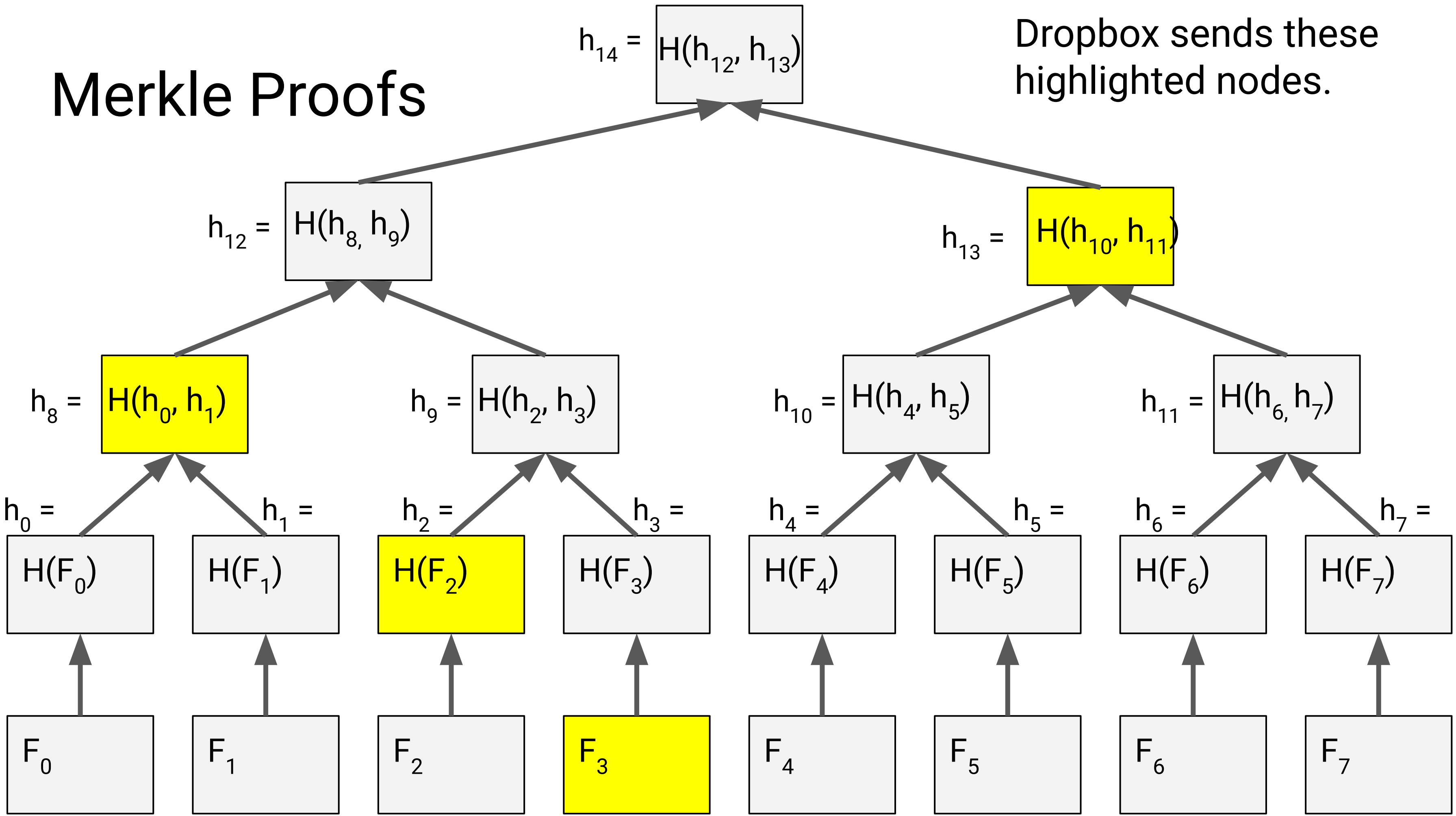
Proving/Verifying Integrity: Merkle Tree



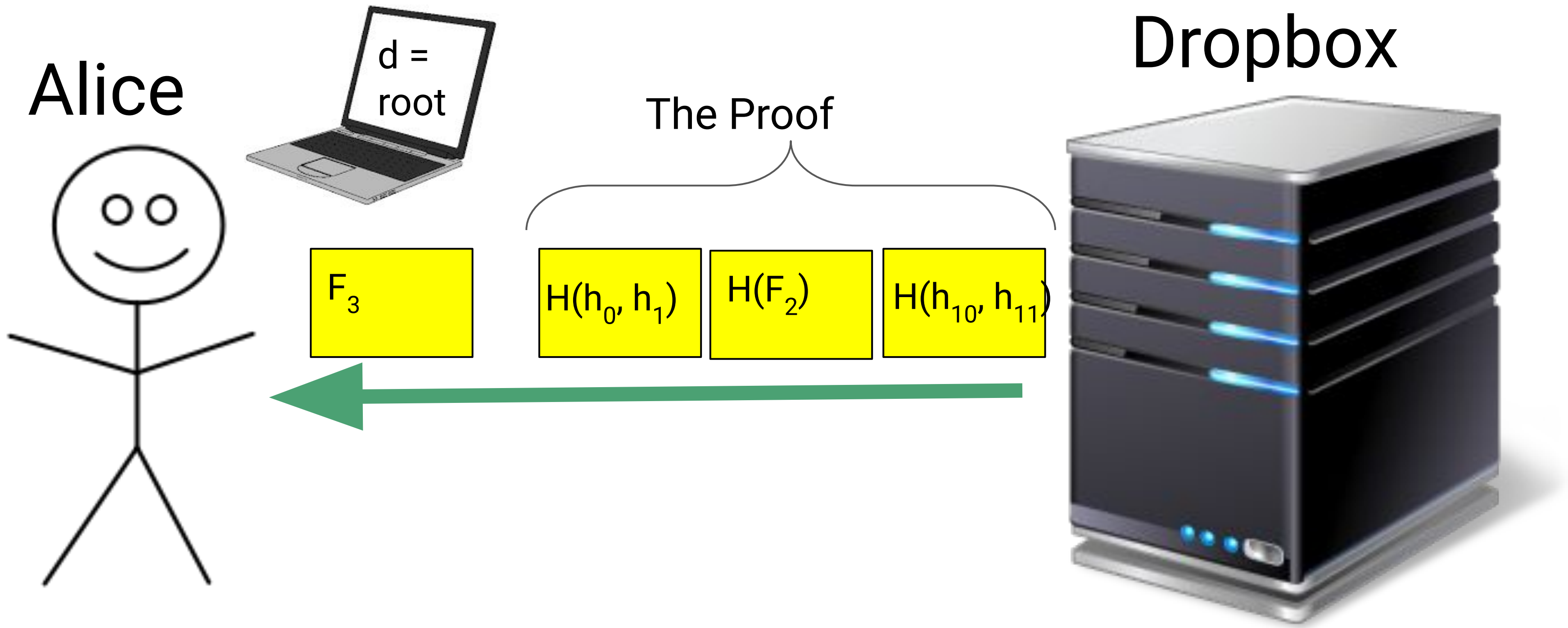
How does Dropbox respond with a proof?

Merkle Proofs

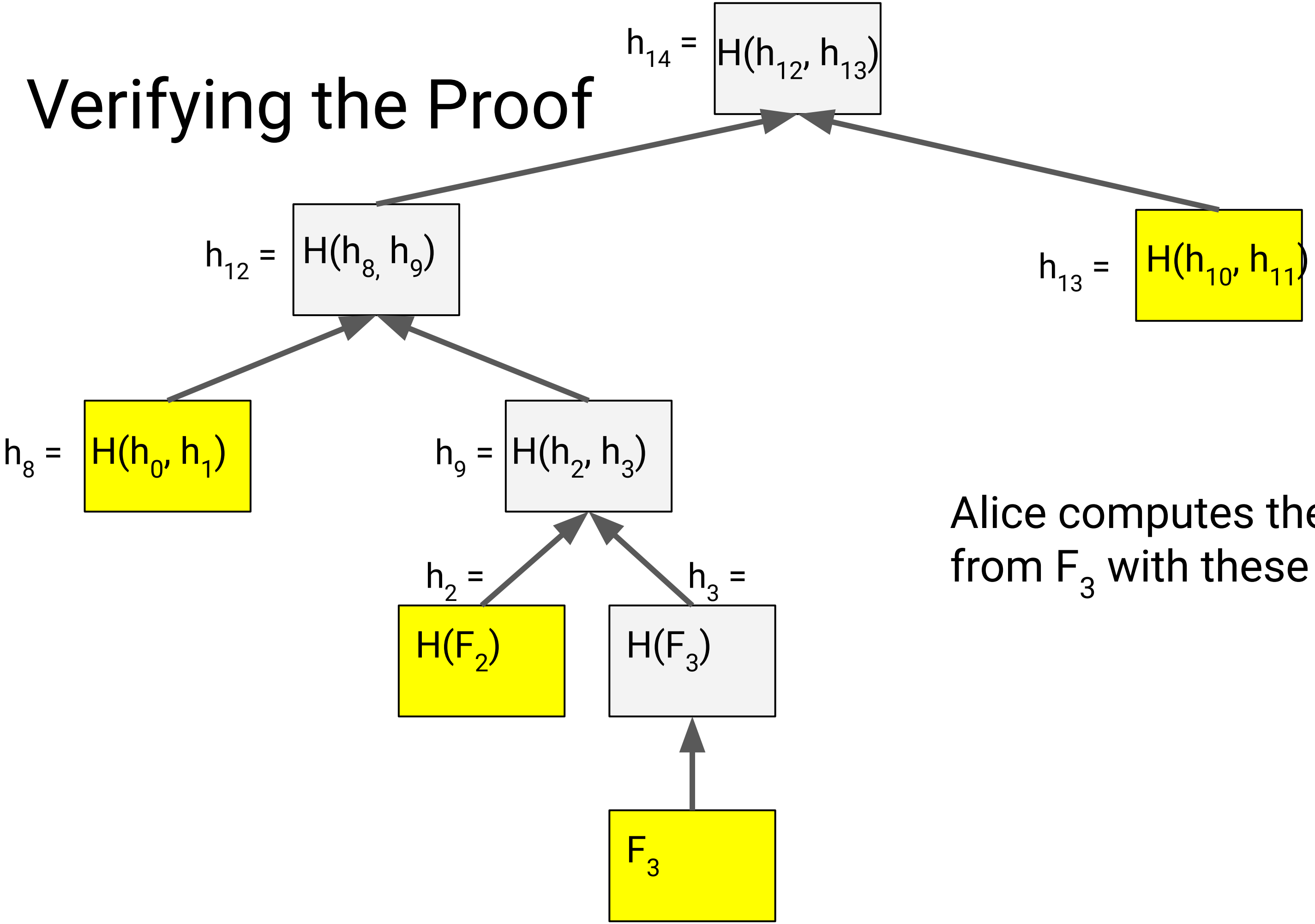
Dropbox sends these highlighted nodes.



Proving/Verifying Integrity: Merkle Tree



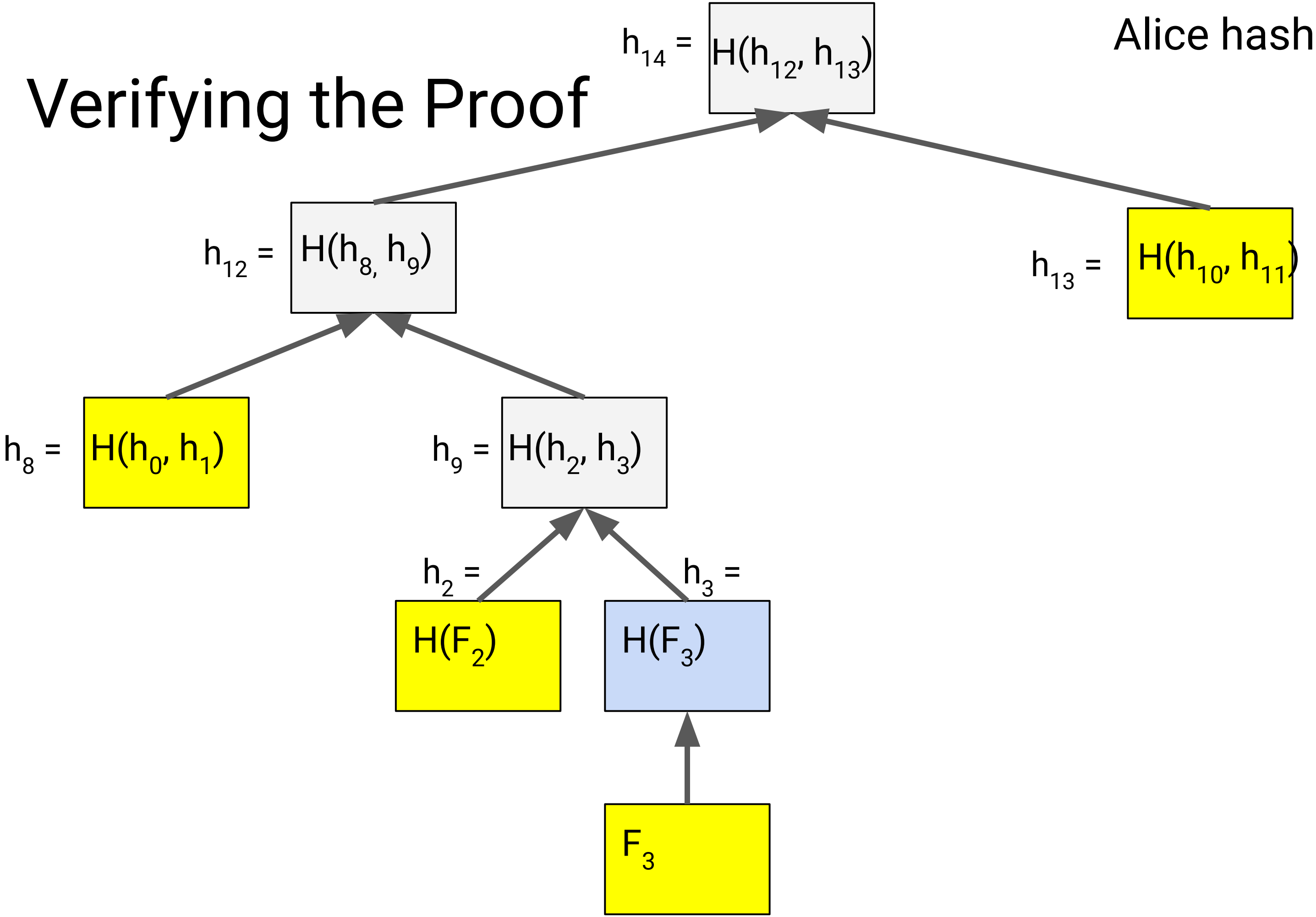
Verifying the Proof



Alice computes the root starting from F_3 with these highlighted proof.

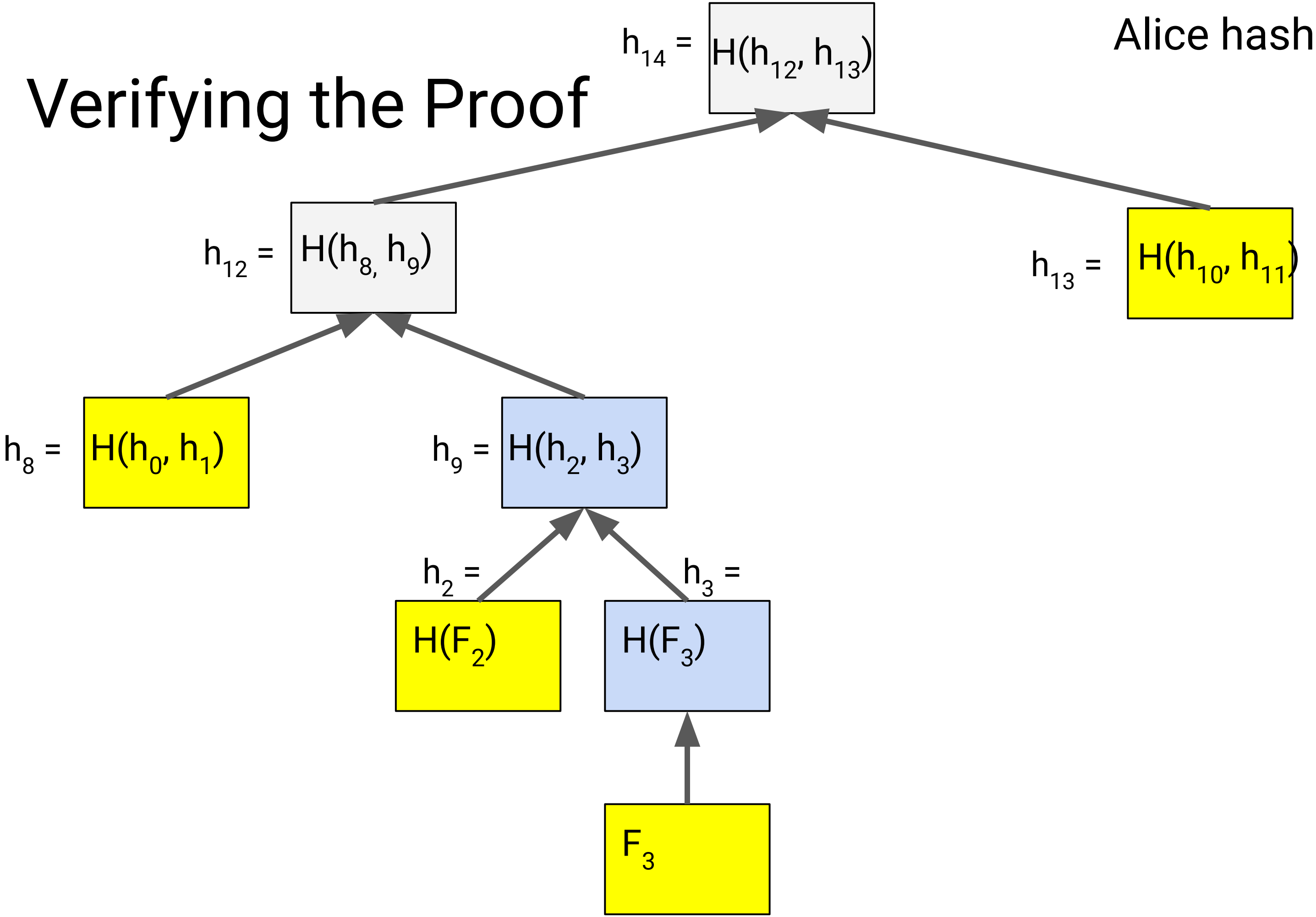
Verifying the Proof

Alice hashes up the tree.



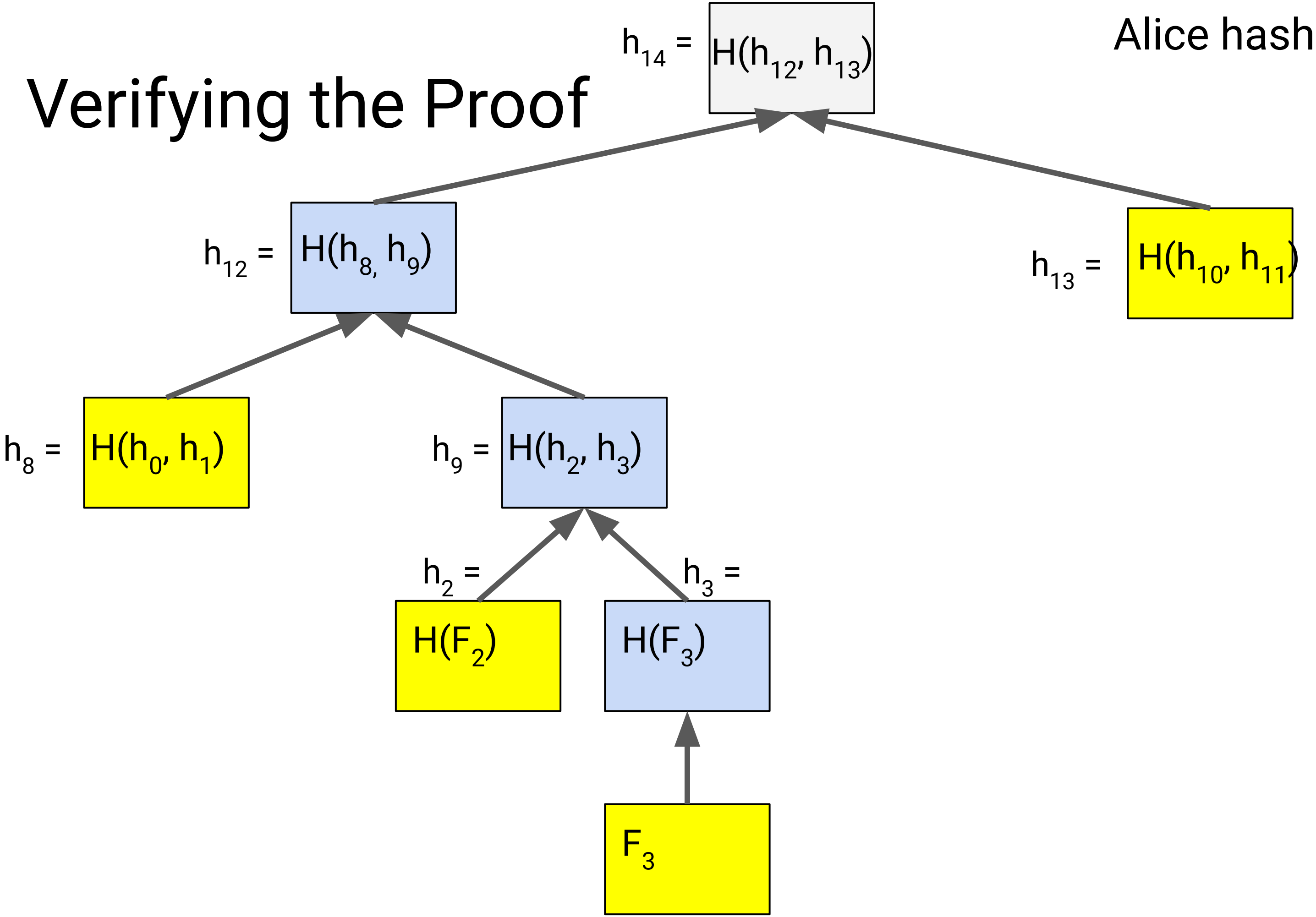
Verifying the Proof

Alice hashes up the tree.

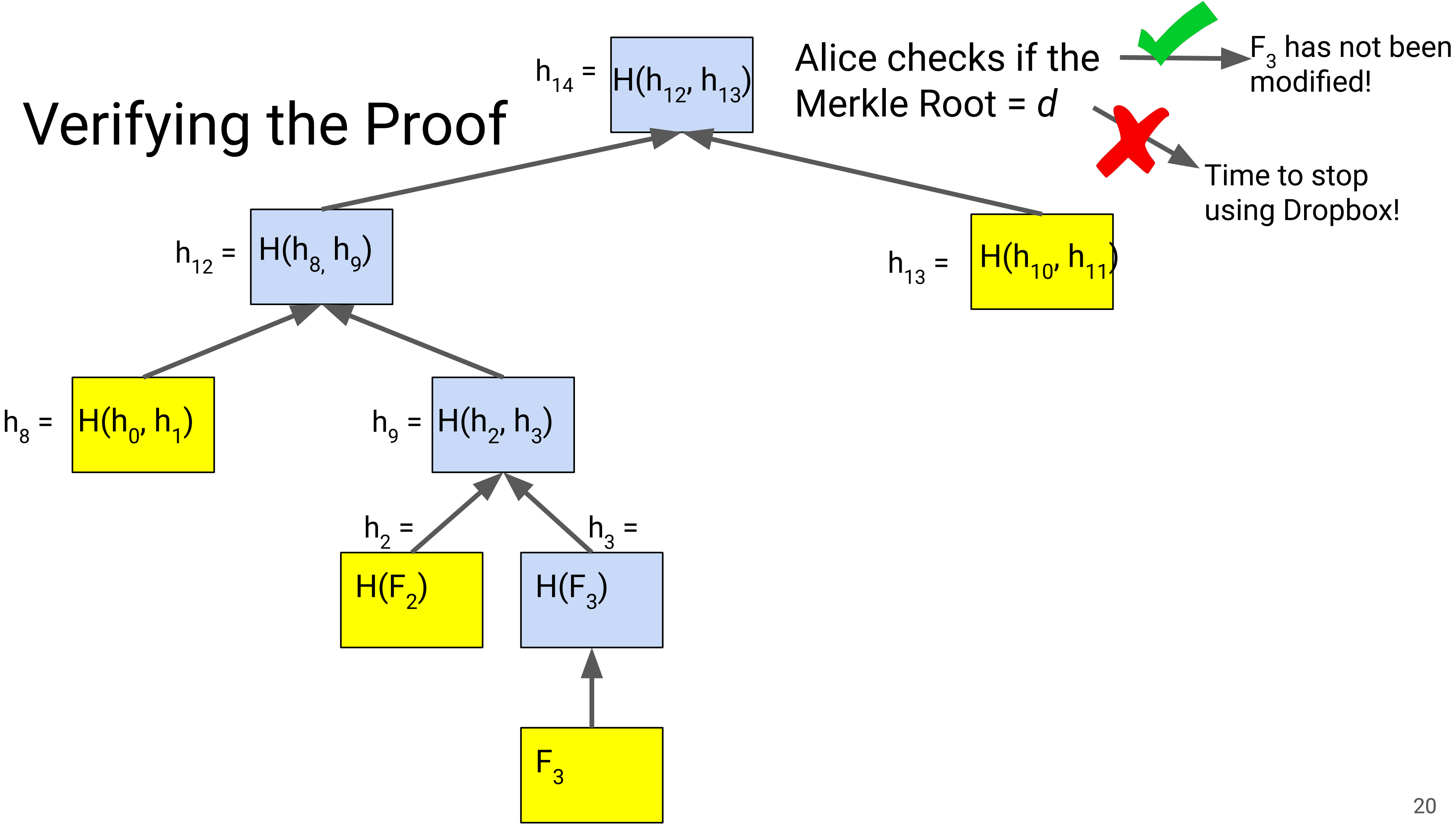


Verifying the Proof

Alice hashes up the tree.



Verifying the Proof

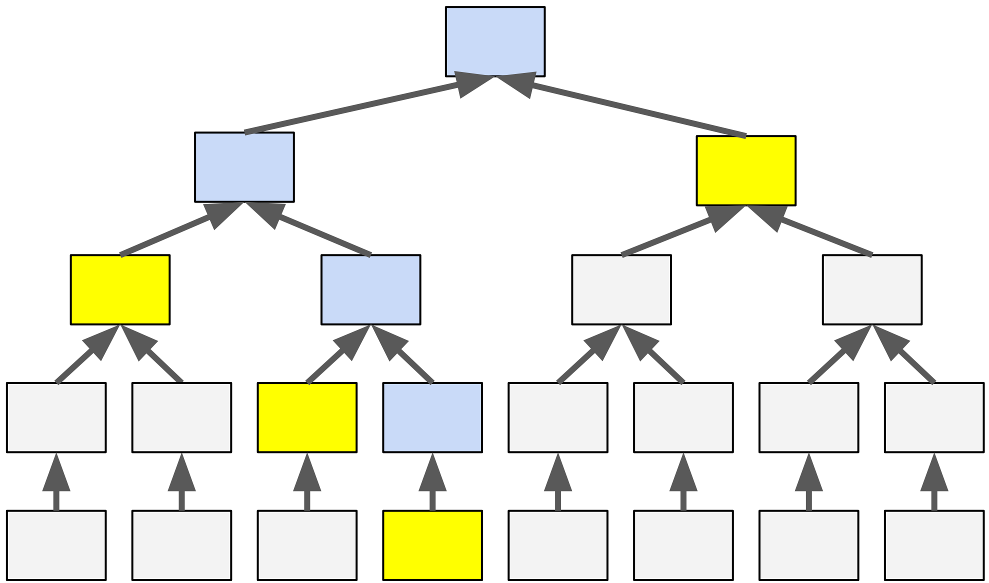


Everyone loves Merkle Trees!

- They're beautiful.
- They're efficient.

n = number of leaves (files)

	Merkle Tree
Construct Tree	O(n)
Proof size	O(log n)
Update File	O(log n)



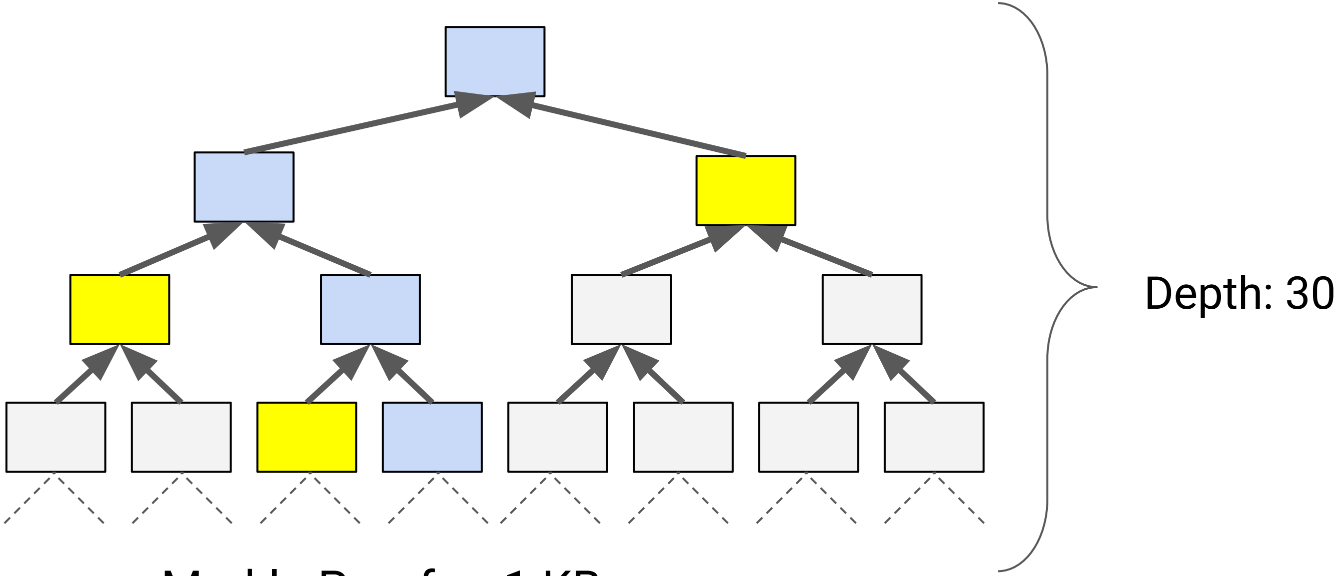
Problem: Many small files ⇒ Merkle proofs too large.

Problem: Many small files ⇒ Merkle proofs too large.

- Suppose Alice has one billion ≈ 2^{30} files.

Problem: Many small files ⇒ Merkle proofs too large.

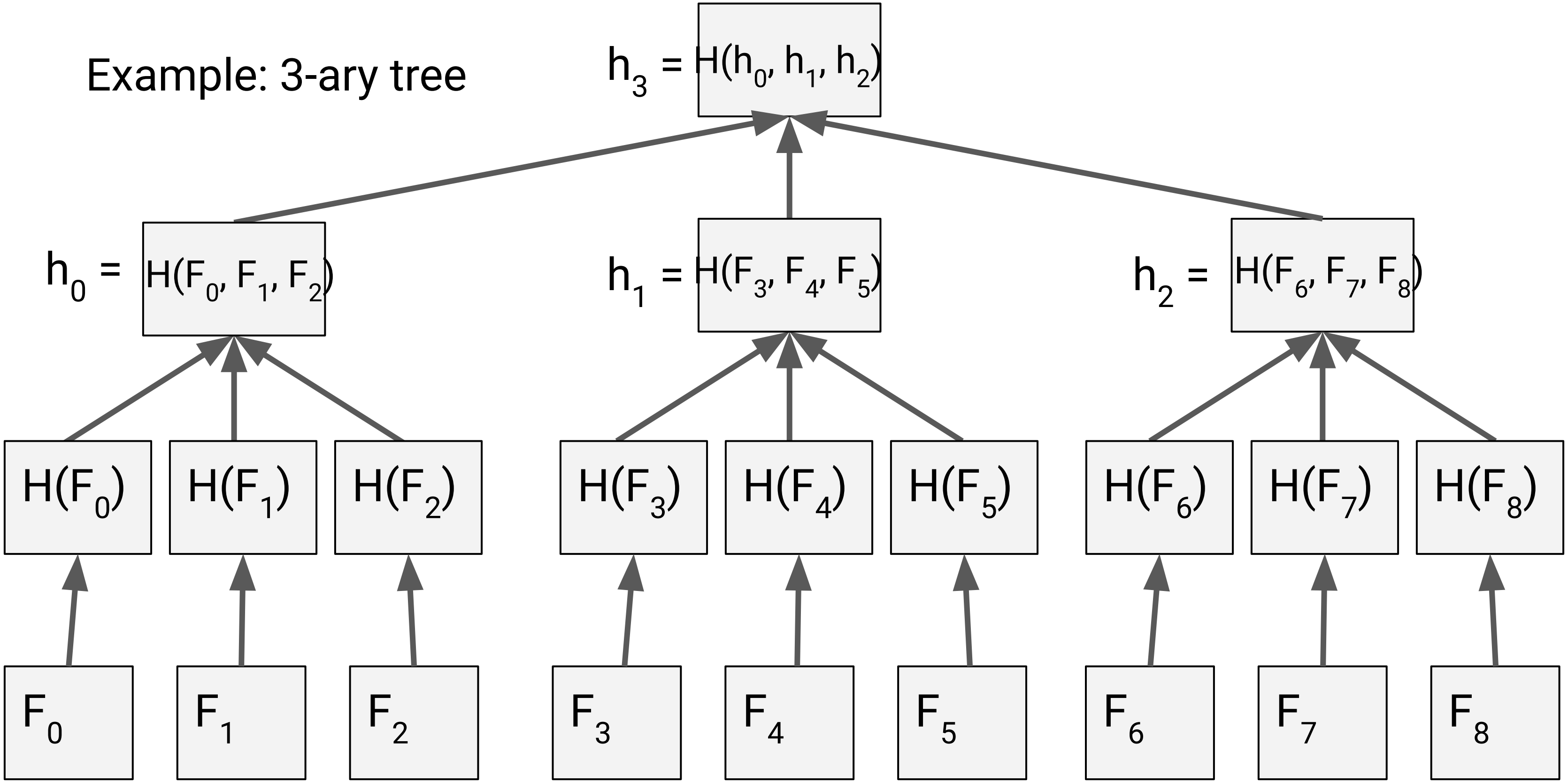
- Suppose Alice has one billion ≈ 2^{30} files.



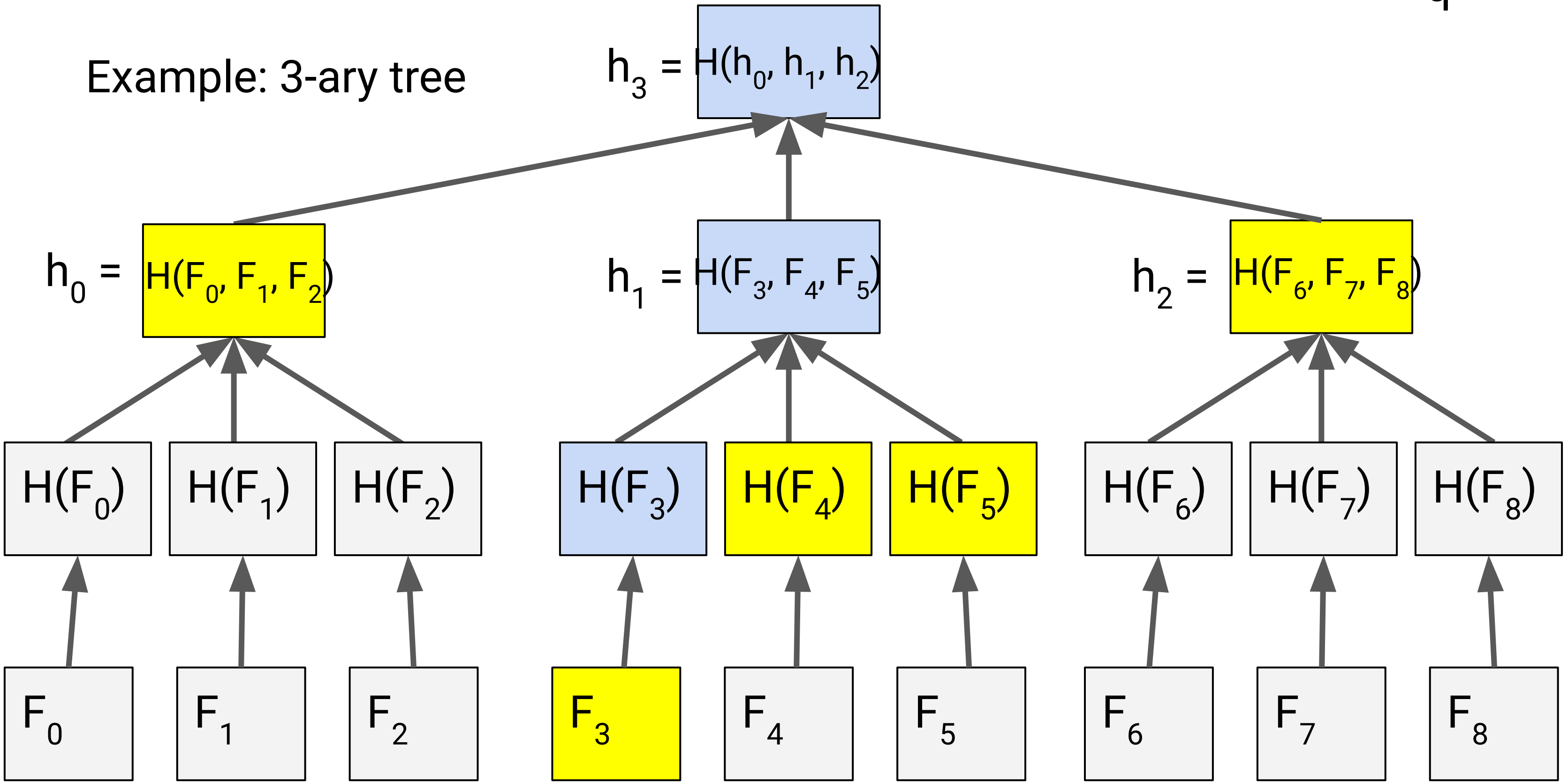
Merkle Proof: ~ 1 KB
(in addition to the file itself)

Possible Solution: q-ary Merkle Tree

Example: 3-ary tree

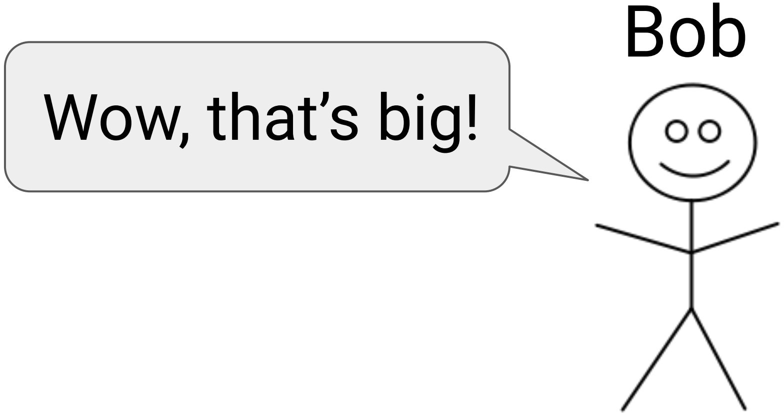


Problem: The Proof Becomes Bigger, $O(q \log_q n)$



Our Work: Verkle Trees reduce the proof size

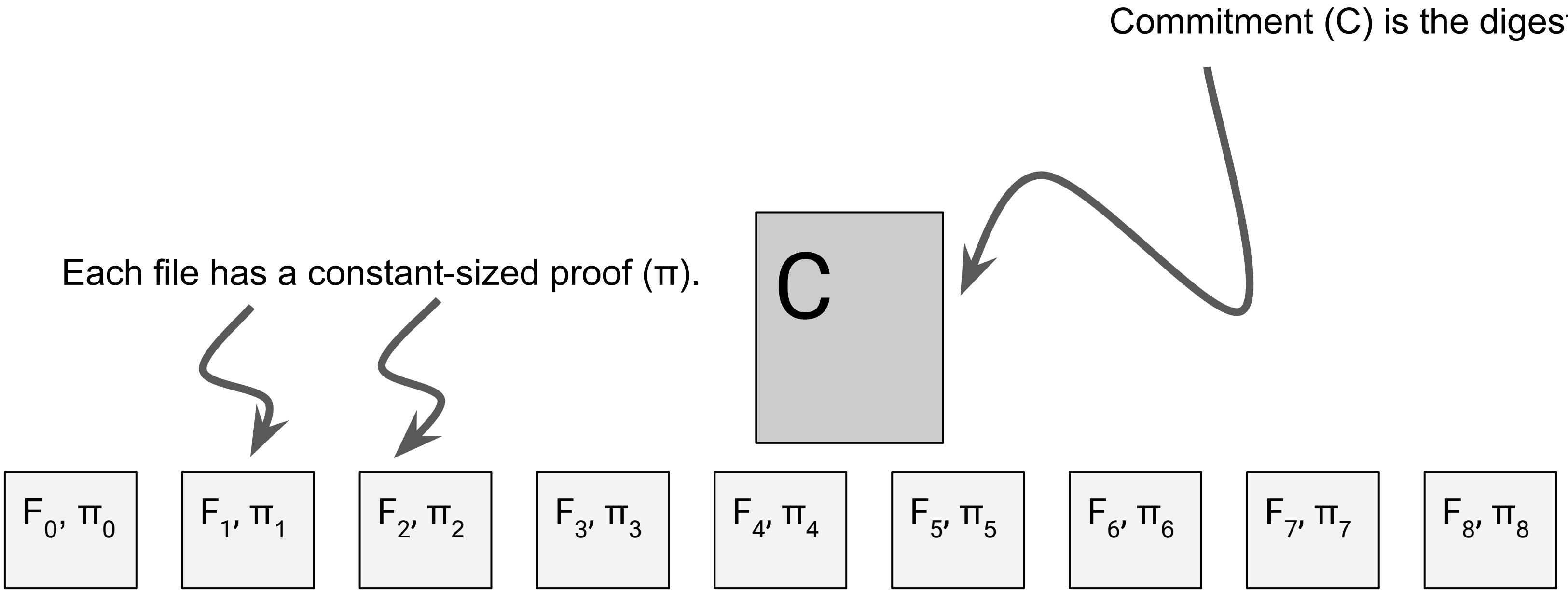
- We pick a q.
- We reduce the proof size from $\log_2 n$ to $\log_q n = \log_2 n / \log_2 q$.
- Factor of **$\log_2 q$ less bandwidth!**
- At the cost of **q times more computation to construct.**
- **Proof verification is $\log_2 q$ times faster.**



Does this matter? (Hint: Yes)

- Merkle hash trees are everywhere in cryptography:
  - Consensus Protocols
  - Public-Key Directories
  - Cryptocurrencies
  - Encrypted Web Applications
  - Secure File Systems

Vector Commitment (VC) Schemes by Catalano and Fiore (2013)

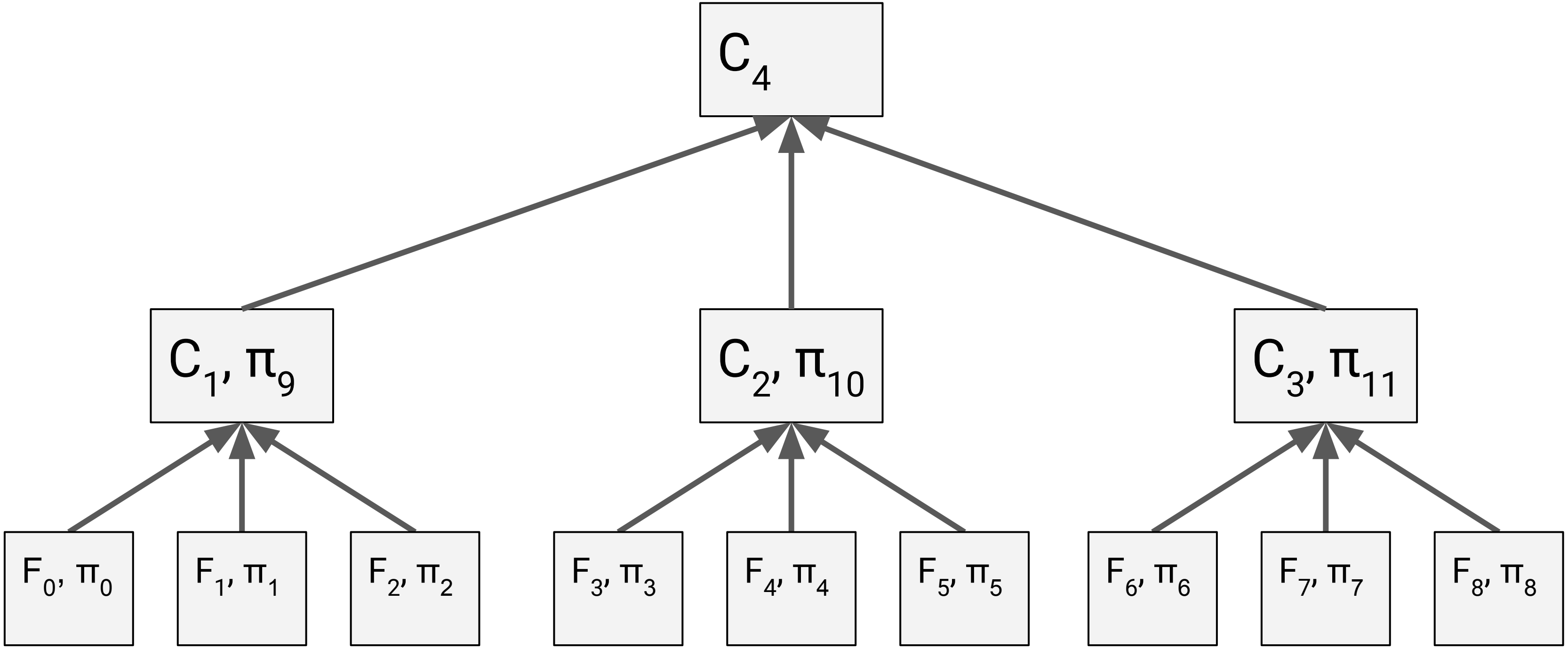


VC Schemes are Computationally Impractical

Scheme/op	Construct	Proof size
Merkle	$O(n)$	$O(\log_2 n)$
VC scheme	$O(n^2)$	$O(1)$

Our Solution: Replace Hash Functions with VC Schemes

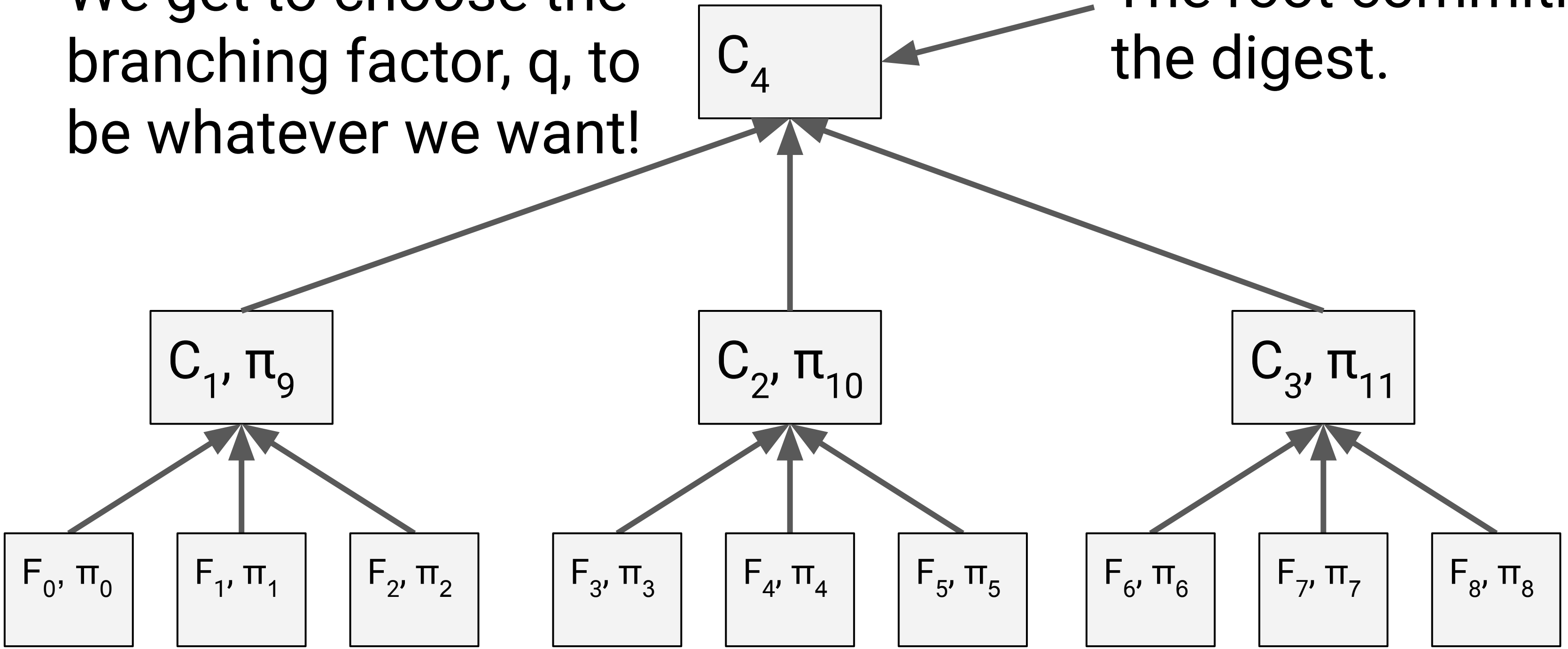
This is the Verkle Tree.



We now have a Verkle Tree!

We get to choose the branching factor, q, to be whatever we want!

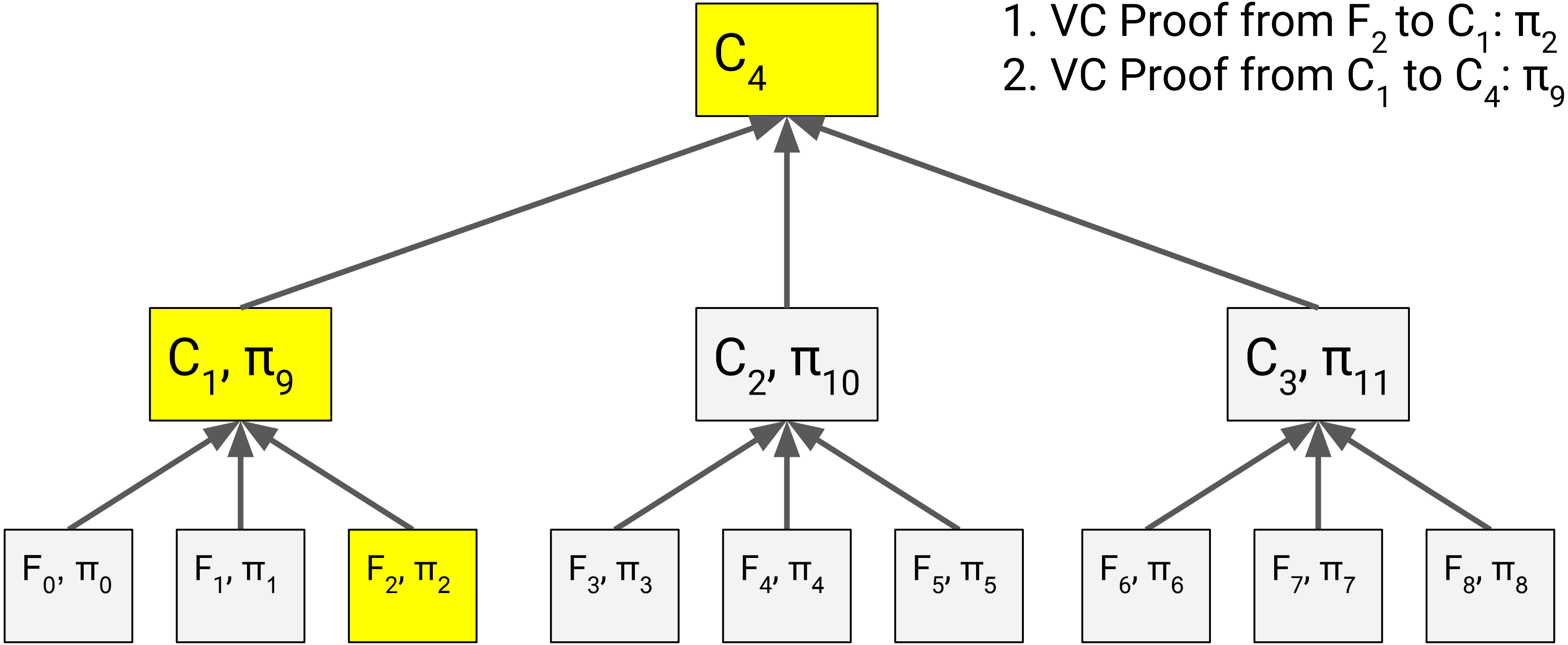
The root commitment is the digest.



Alice Receives $\log_q n$ Constant-Sized π's.

Alice verifies:

1. VC Proof from F_2 to C_1: π_2
2. VC Proof from C_1 to C_4: π_9



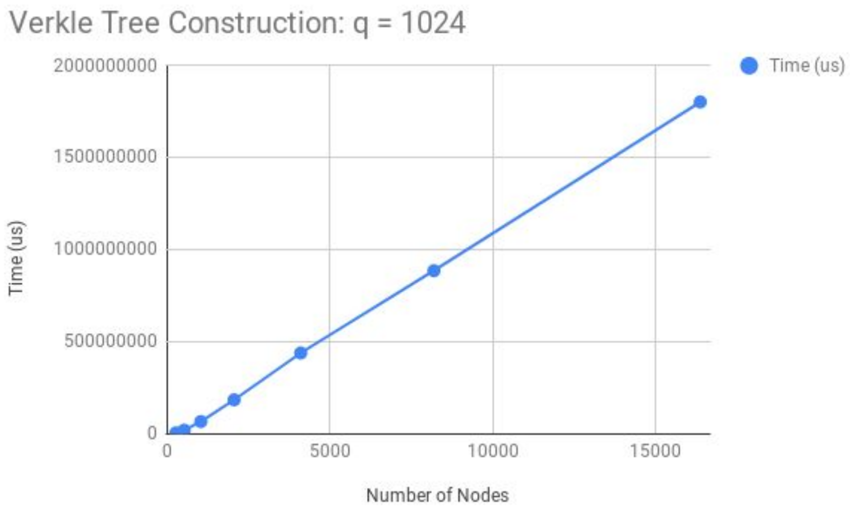
Comparison

Scheme/op	**Construct**	**Update file**	**Proof size**
Merkle	**$O(n)$**	$O(\log_2 n)$	**$O(\log_2 n)$**
q-ary Merkle	**$O(n)$**	$O(q \log_q n)$	**$O(q \log_q n)$**
VC scheme	**$O(n^2)$**	**$O(n)$**	**$O(1)$**
q-ary Verkle	*$O(qn)$*	*$O(q \log_q n)$*	*$O(\log_q n)$*

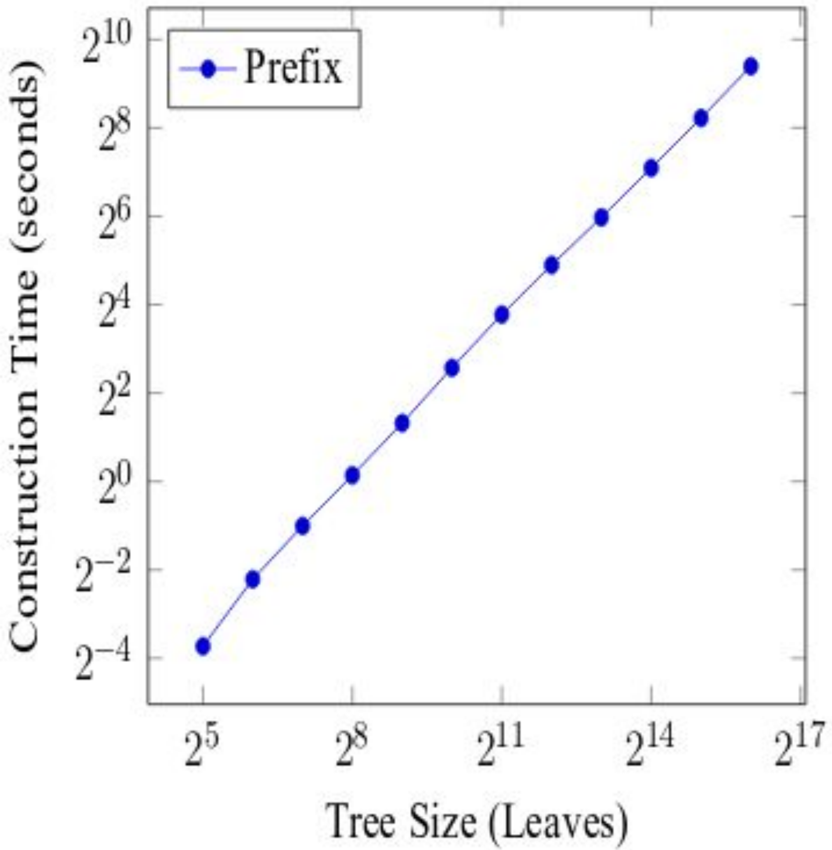
Verkle Trees let us trade off proof-size vs. construction time.

My Contribution

- I proved complexity bounds for Verkle Trees.
- I implemented and optimized Verkle Trees in C++.
- Benchmarked implementations.

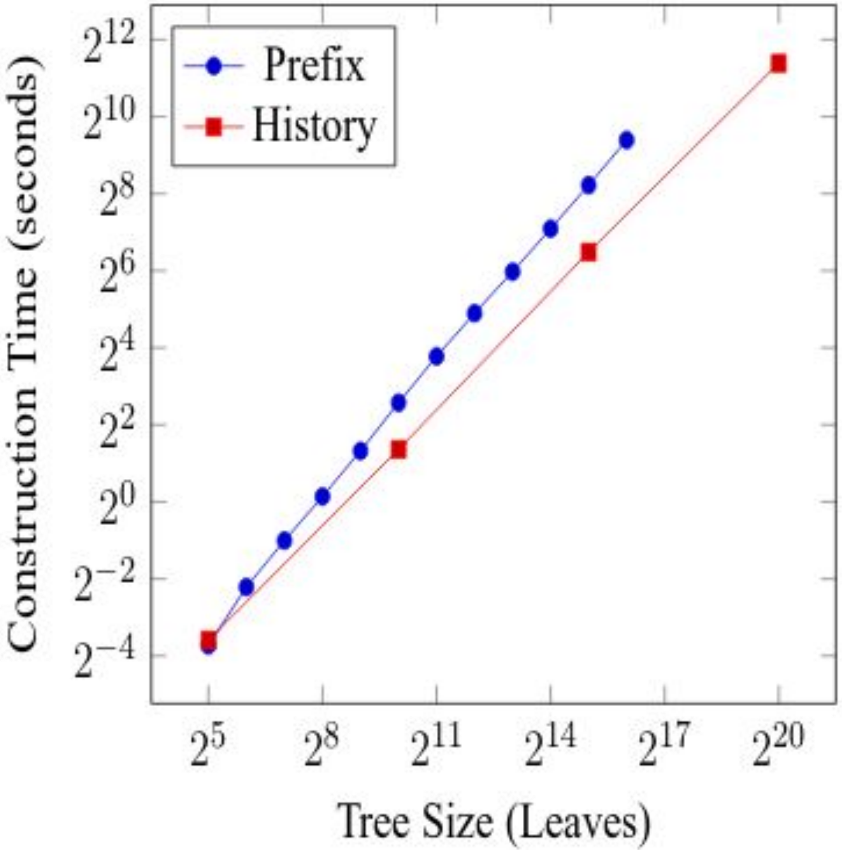


Prefix Tree

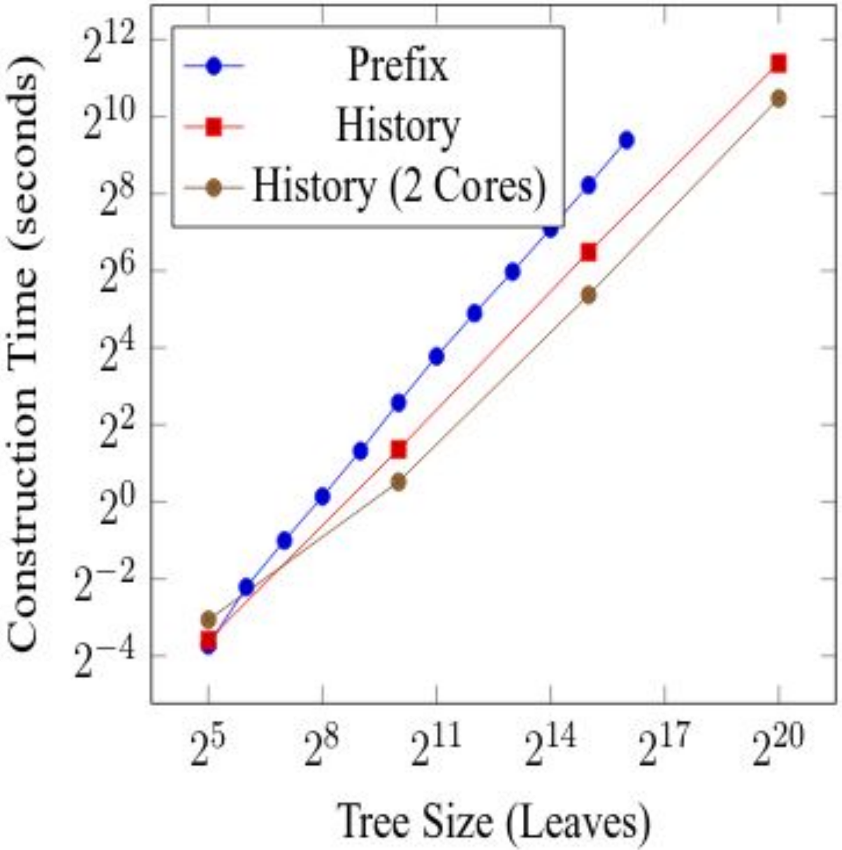


Prefix
Construction Time (seconds)
2^{10}
2^{8}
2^{6}
2^{4}
2^{2}
2^{0}
2^{-2}
2^{-4}
2^{5}
2^{8}
2^{11}
2^{14}
2^{17}
Tree Size (Leaves)

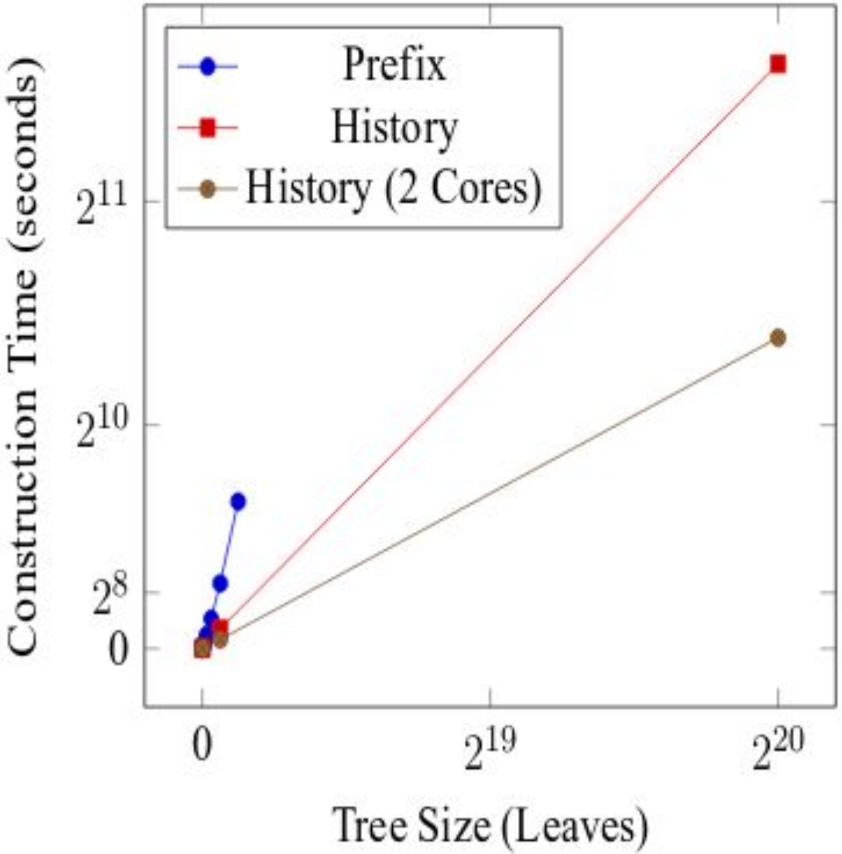
Prefix Tree vs. History Tree



Prefix Tree vs. History Tree vs. Parallelized History



On a Linear Scale:



Acknowledgements

- Thank you Alin!
- Thank you PRIMES!
- Thank you Mom and Dad!