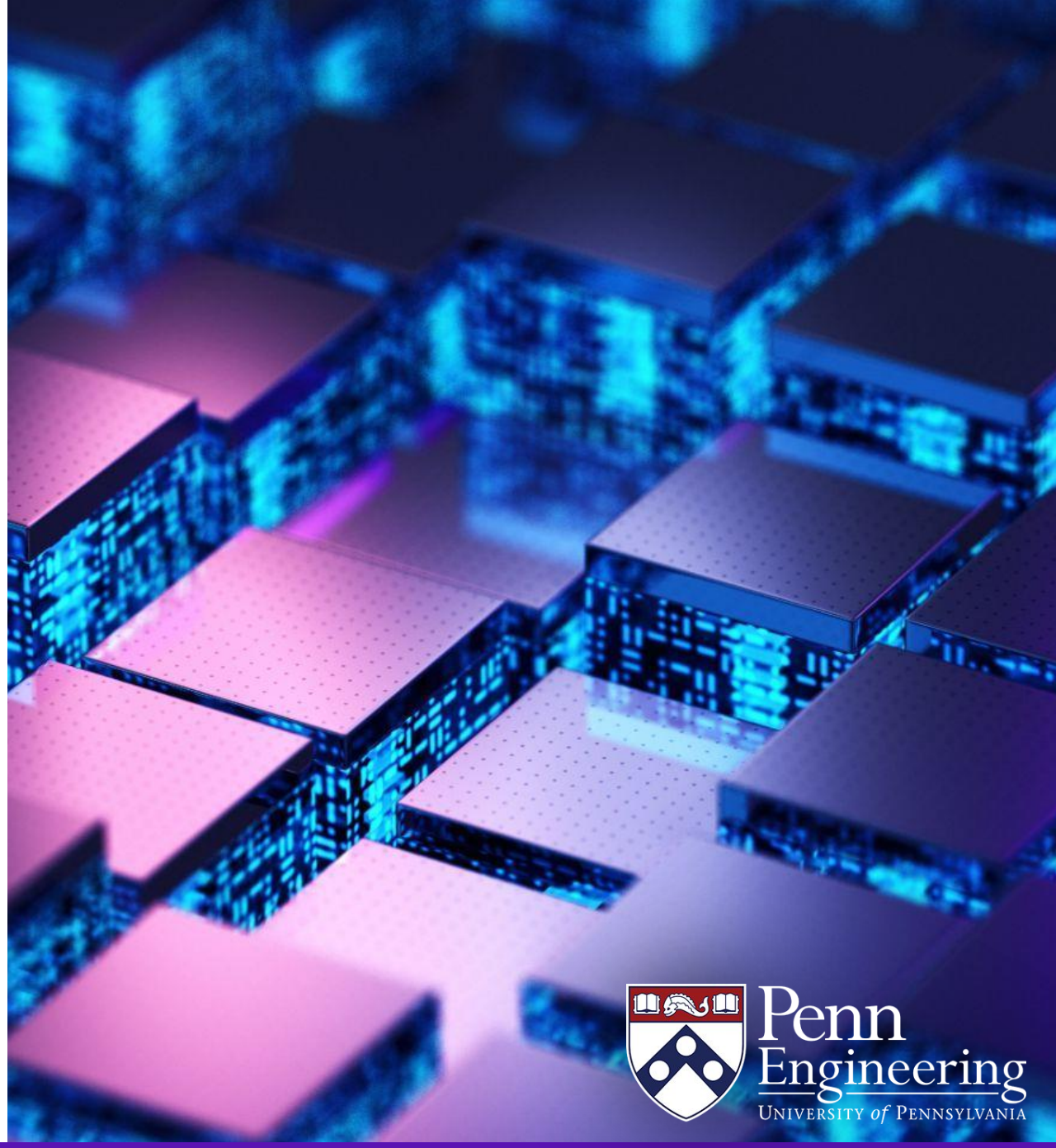
EAS 5830: BLOCKCHAINS

# Merkle Trees

Professor Brett Hemenway Falk



Penn Engineering
University of Pennsylvania

# Fingerprinting

- Hashes create a compact “fingerprint” of an arbitrary file / data
- How do you create a fingerprint of multiple files?

$$H(x_1, x_2, x_3, ..., x_n)$$

- How can you “prove” $x_i$ was included in the hash?
- How can you update hash value when a single $x_i$ changes?

# Proving membership

Prover

Verifier

$h = H(x_1, x_2, x_3, ..., x_n)$

$h$

What's the *i*th element?

$x_i$

Can't verify this unless prover sends $x_1, x_2, x_3, ..., x_n$
Verifier has to do $O(n)$ work to verify membership

# Updates

Prover

Verifier

$$h = H(x_1, x_2, x_3, ..., x_n)$$

$h$

$$x_3 \rightarrow x_3{}'$$

# Updates

Prover

Verifier

$$h = H(x_1, x_2, x_3', ..., x_n)$$

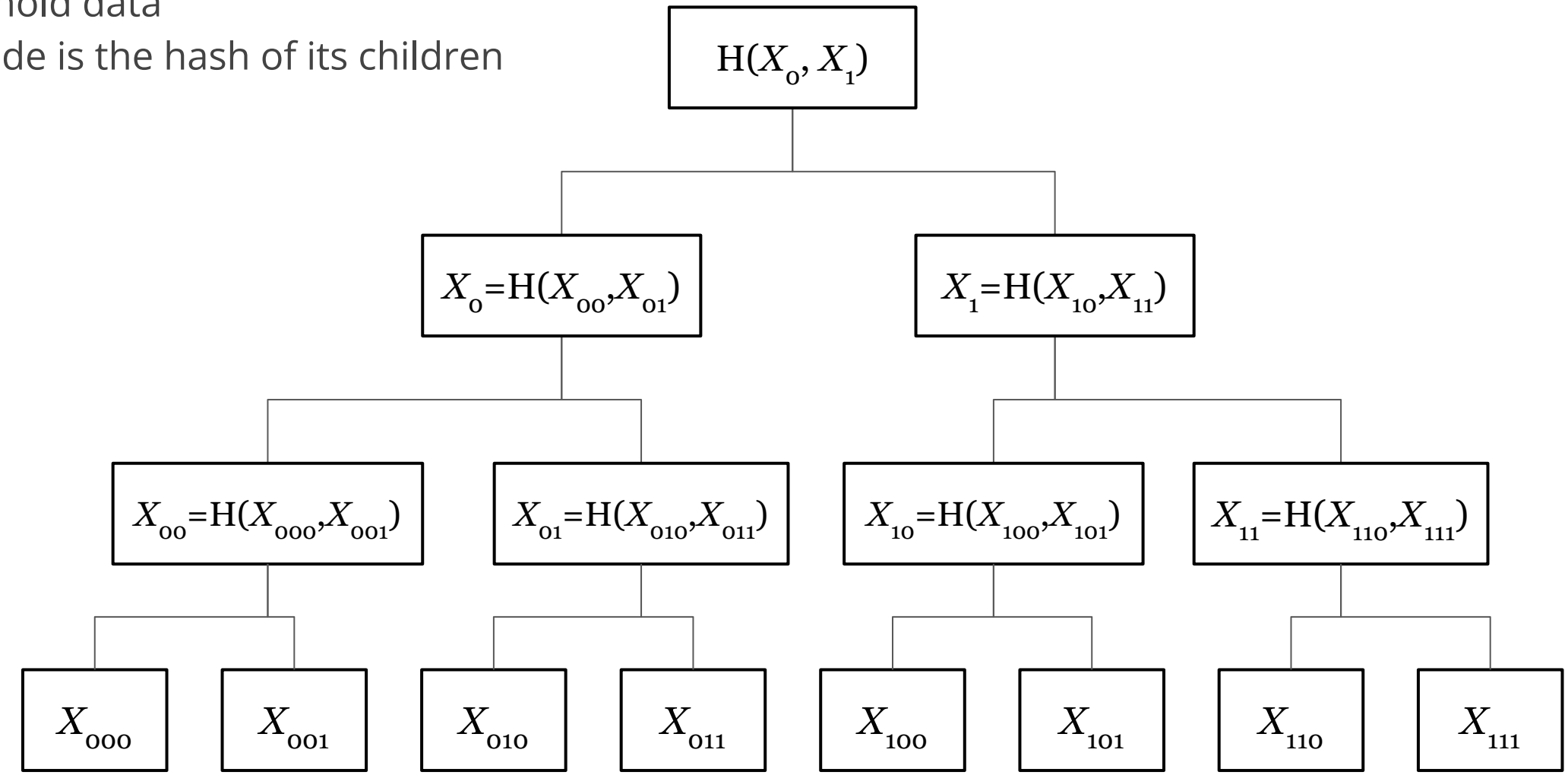
$h'$

$$x_3 \rightarrow x_3'$$

Prover has to do $O(n)$ work to update the hash

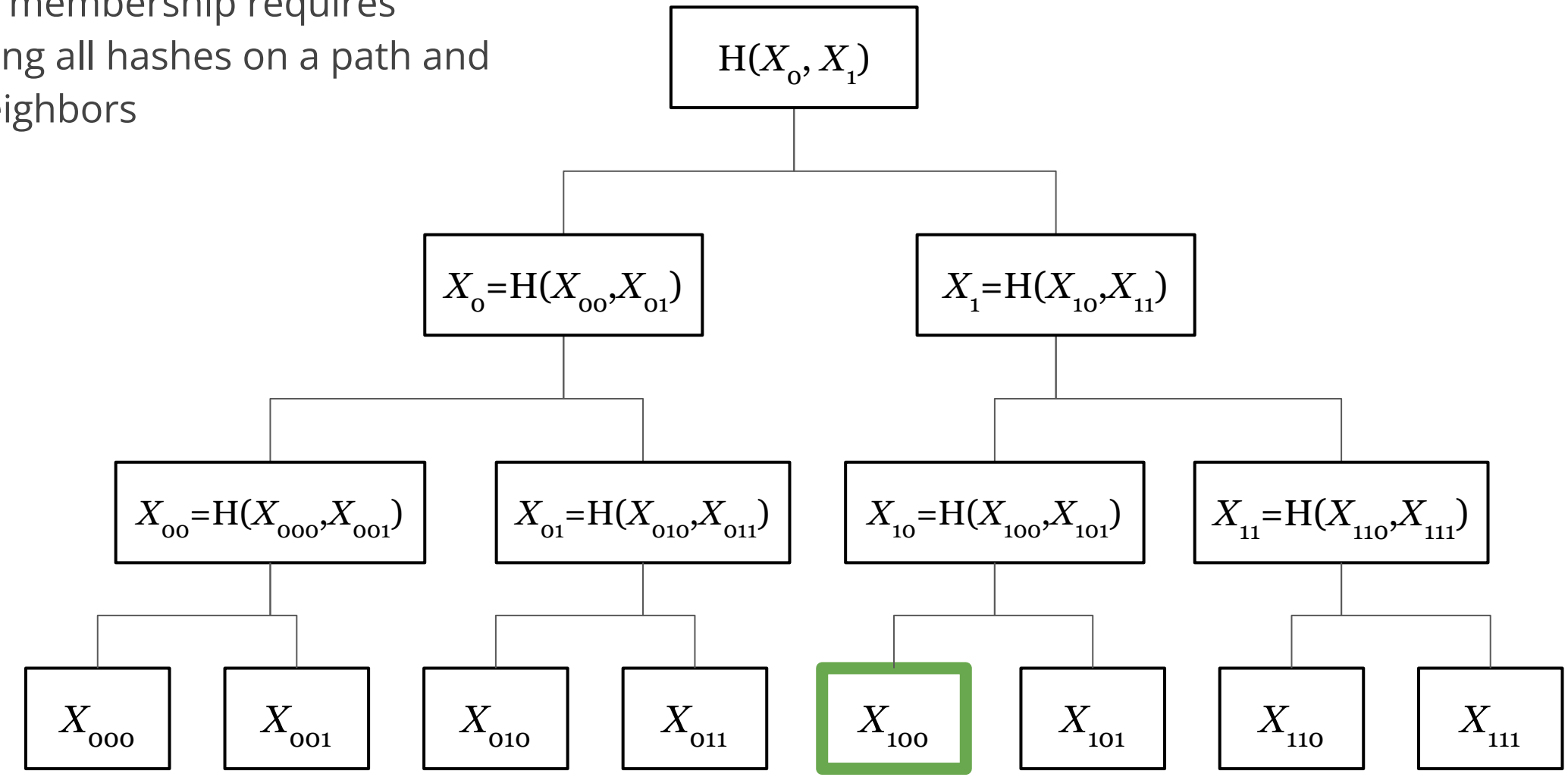
# Merkle Trees

- Leaves hold data
- Each node is the hash of its children



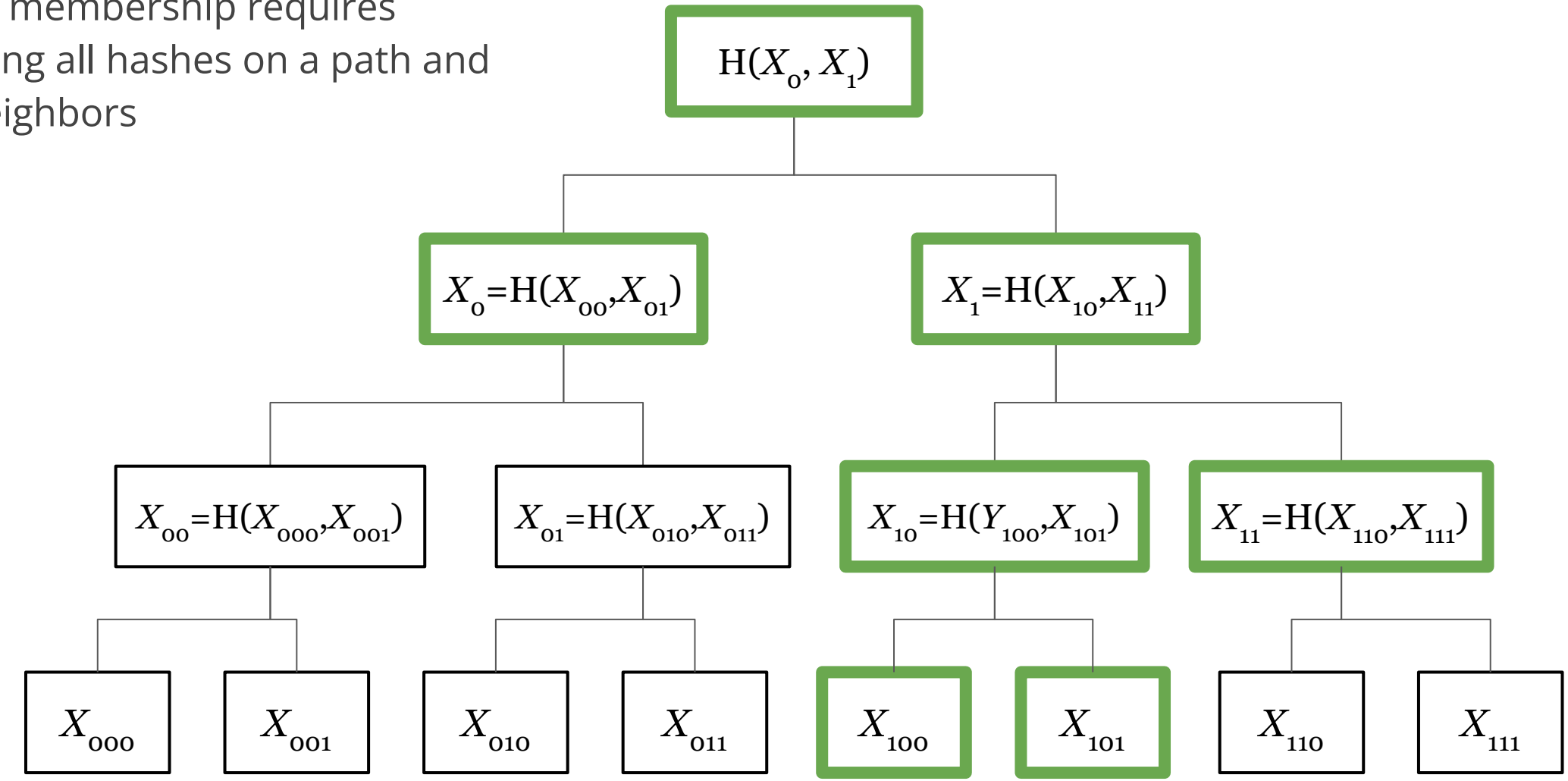
# Merkle Tree Proofs

Proving membership requires producing all hashes on a path and their neighbors



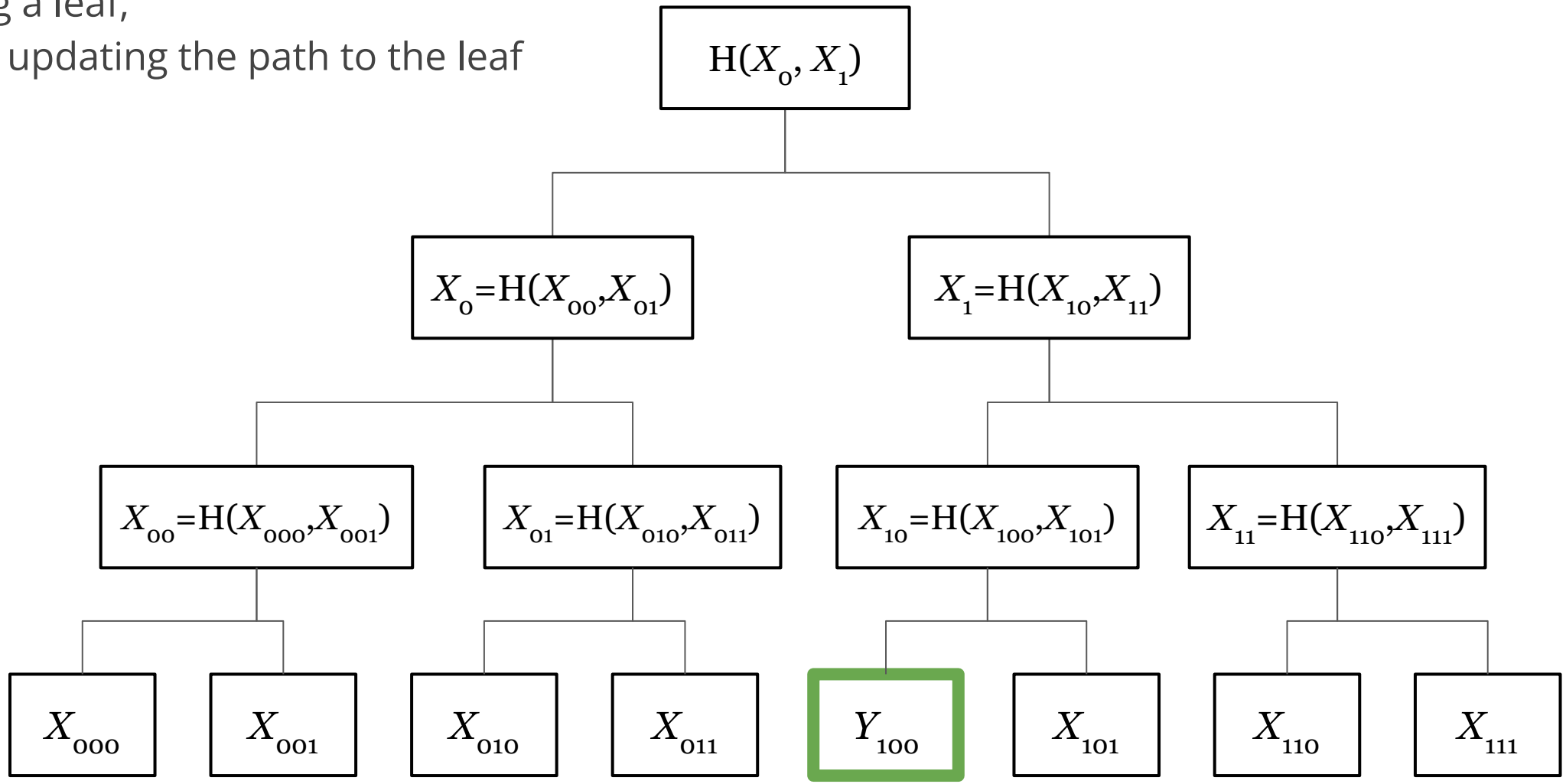
# Merkle Tree Proofs

Proving membership requires producing all hashes on a path and their neighbors



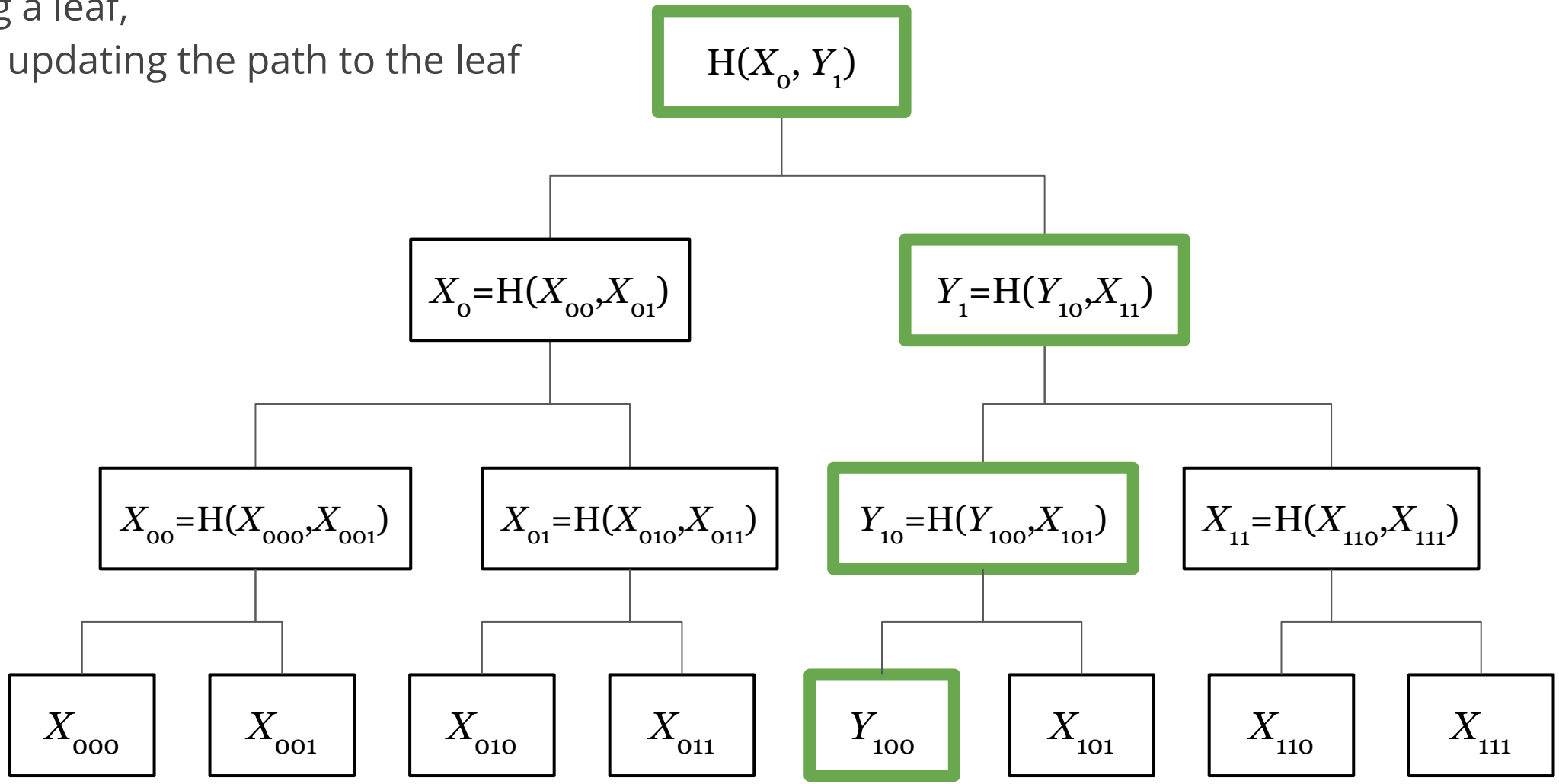
# Merkle Tree Updates

- Updating a leaf,
  requires updating the path to the leaf



# Merkle Tree Updates

- Updating a leaf,
  requires updating the path to the leaf

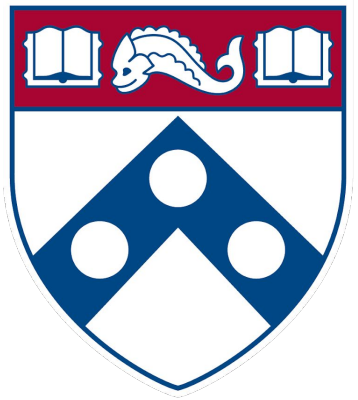


# Merkle Trees

- Root hash is independent of size of tree (e.g. 256 bits)
- If tree has $n$ elements
  - Proving membership requires $\mathrm{O}(\log(n))$ hashes
  - Updating root requires $O(\log(n))$ hashes

# Merkle Trees in Blockchains

- Average Bitcoin block has over 1500 transactions
- Each Bitcoin block contains root of Merkle Tree of all its transactions
- Validators can check whether a single transaction was in the block *without* downloading all transactions
- Merkle trees used similarly in almost all cryptocurrencies



Penn
Engineering
UNIVERSITY of PENNSYLVANIA