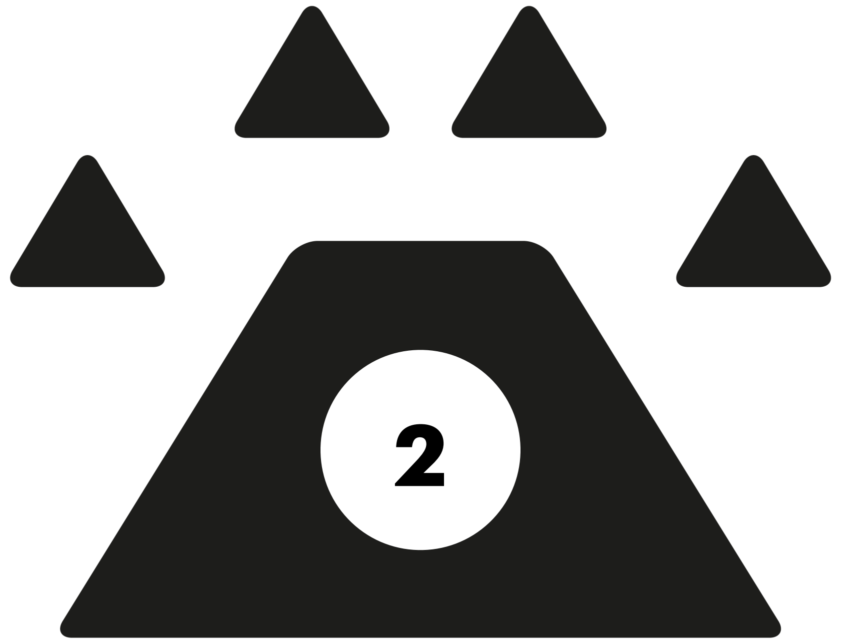
Foundations of High Speed Cryptography

Module 1 - Theory

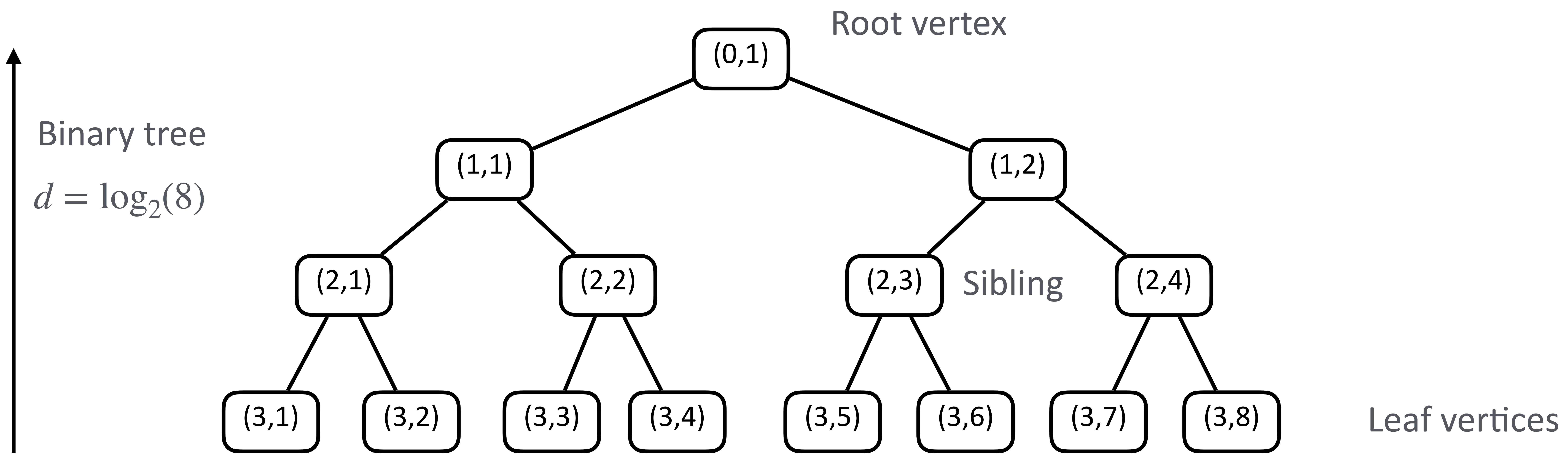
Lesson 3 - Hashes and Merkle Trees



Merkle Tree

Merkle Tree: General Principles

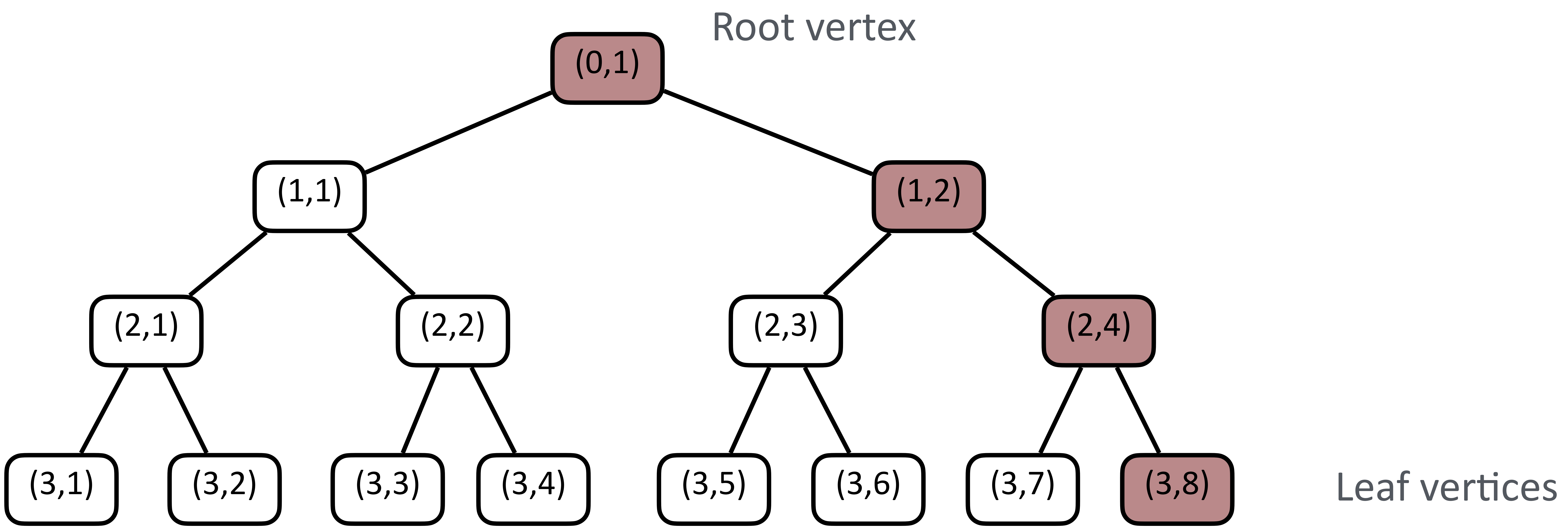
- A Merkle tree is a tree graph data structure of depth $d = \log_k l$ with l leaves, and k arity



- Leaves: contain data blocks or Cryptographic hashes of data blocks
- Non leaf nodes contain hashes of their children node's hashes
- The root uniquely represent the entire data set

Merkle Tree: General Principles

- The root uniquely represent the entire data set

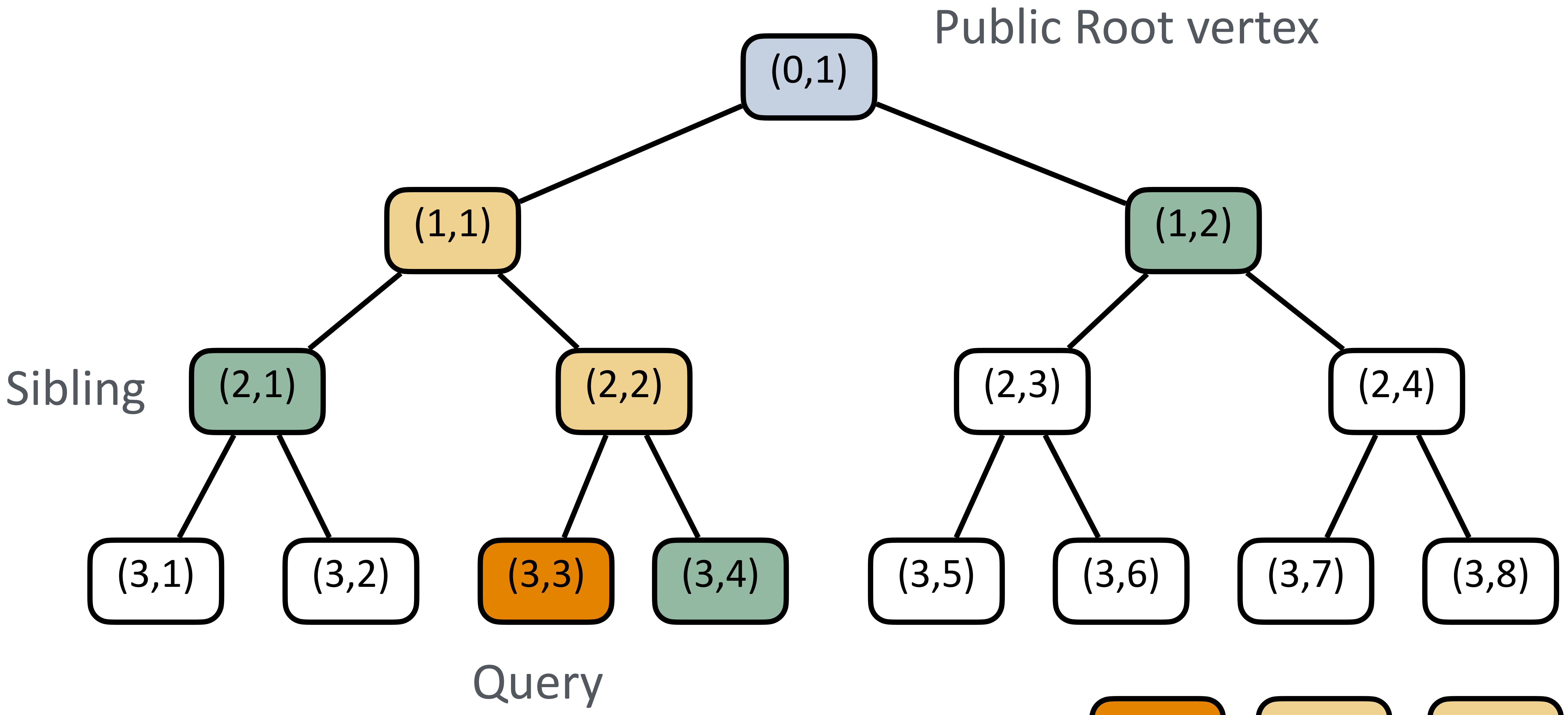


- If any leaf block changes, the root hash changes, and thus Merkle trees can be used to verify data integrity.

- On the other hand, if the root is public, any private leaf node can be authenticated!

Merkle Tree: General Principles

- Authentication paths: node values that when hashed will produce the correct root



- Path (i) : computable vertices that lie from a leaf to the root

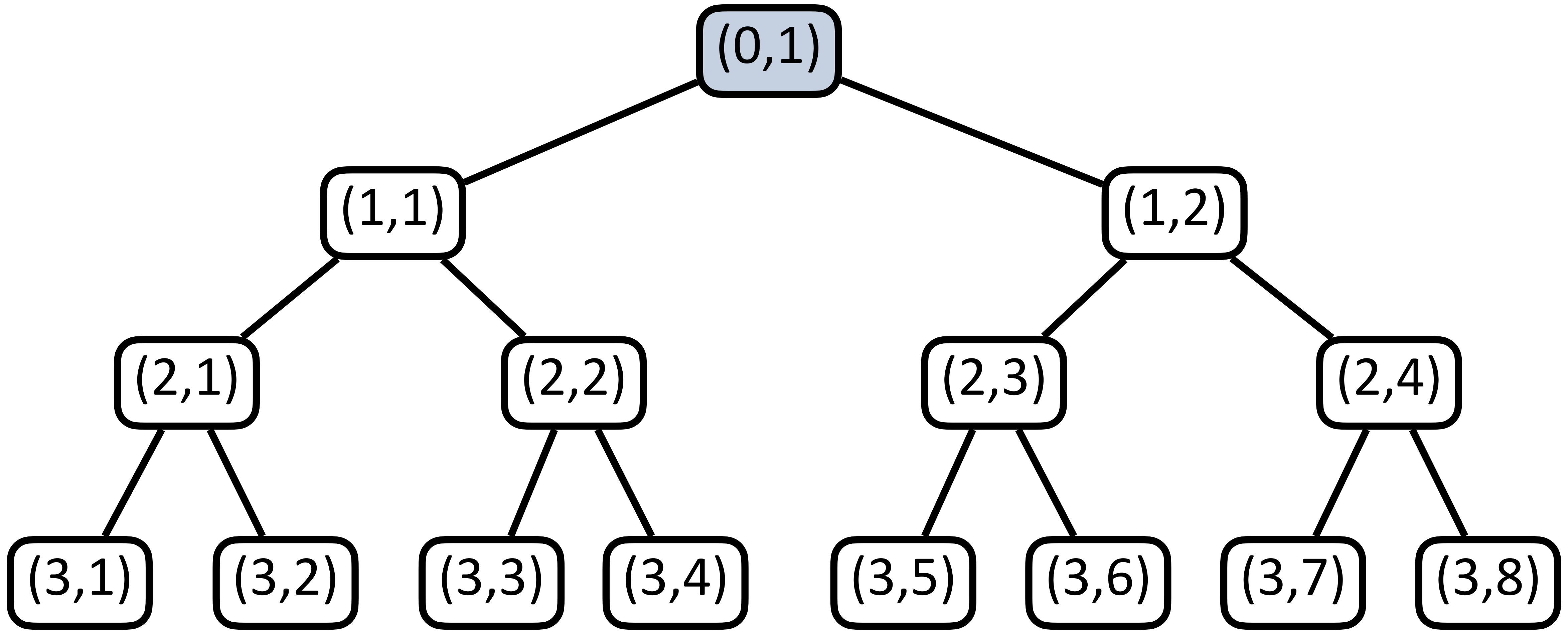
path(3)= (3,3) (2,2) (1,1) (0,1)

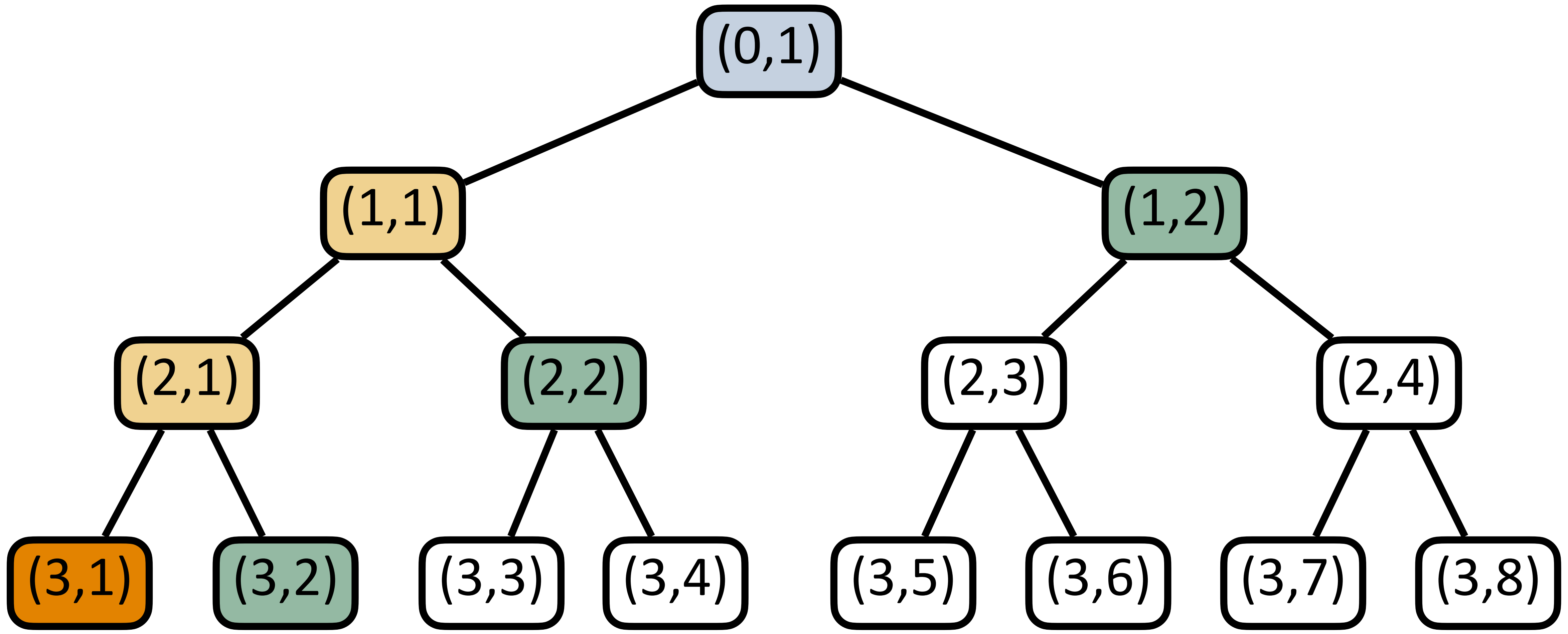
copath(3)= (3,4) (2,1) (1,2)

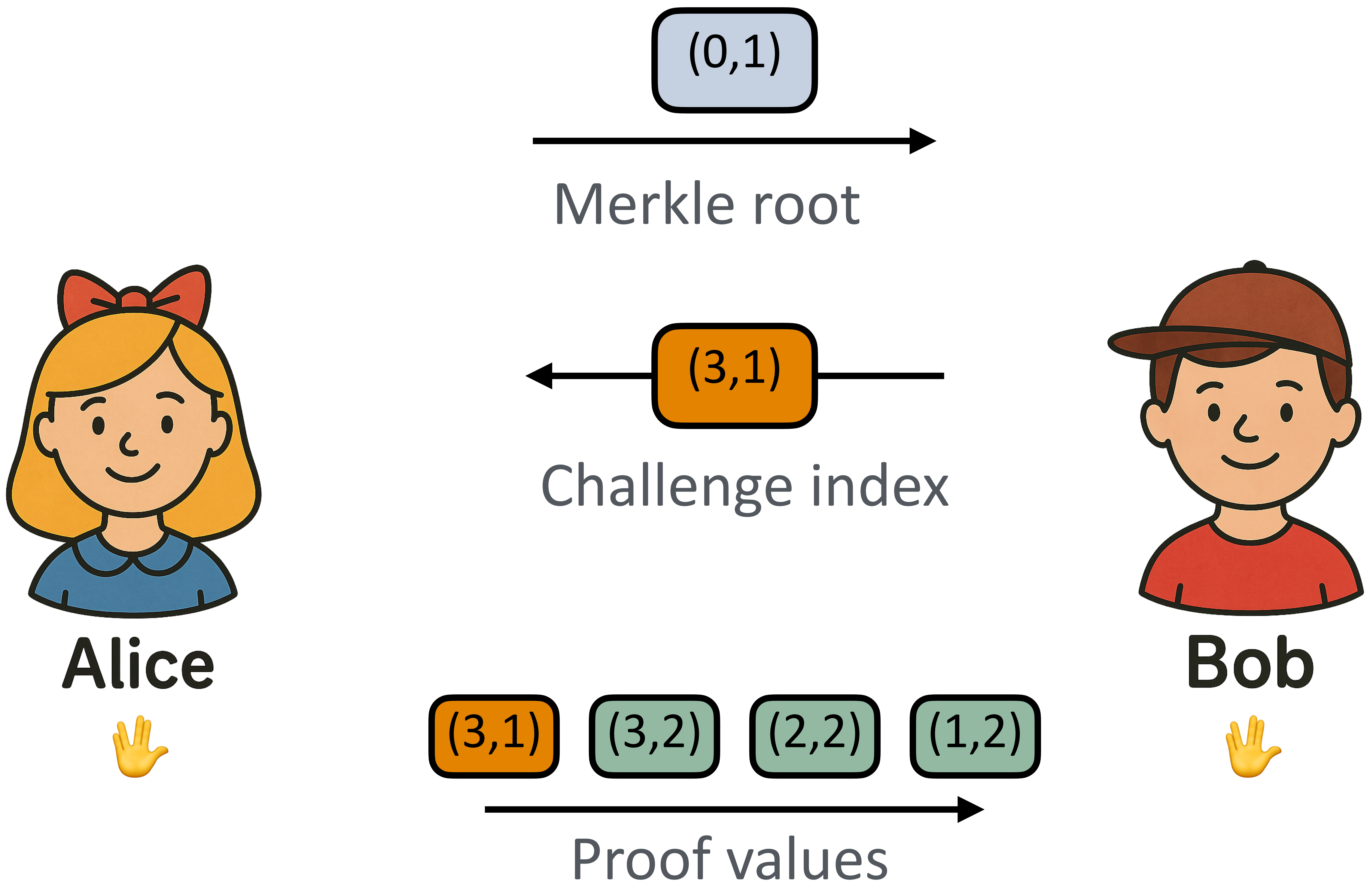
- Co-Path(i): All the siblings of each vertex in path. Authentication proof! Of length $(k-1)log_k(n)$

Merkle Tree: Commitment Scheme

Alice wants to convince Bob that she knows a vector of elements, they agree on a Hash function H







Bob can compute (2,1) (1,1) all the way to the root

Computed root =? committed root

Merkle Tree: Depth vs arity vs proof size

Merkle Tree: Depth vs arity vs proof size

- The depth of the tree graph can be reduced by increasing the arity

Arity 4 tree of 16 elements

- Ideal for low memory situations:, where proof size is not relevant, eg: proofs of integrity in a non block chain setting

path(3)= (2,3) – (1,1) – (0,1) Shorter path = Less nodes to compute

copath(3)= (2,1) (2,2) (2,4) (1,2) (1,3) (1,4) More siblings = larger proof size

Merkle Tree: Depth vs arity vs proof size

- **Higher arity** reduces the **depth** but increases the **co-path width per level**, leading to larger proofs overall.

$$\frac{\text{Proof size k-ary}}{\text{Proof size binary}} \qquad \frac{(k-1)\log_k n}{\log_2 n} = \frac{k-1}{\log_2 k}$$

Strictly monotonically increasing for $k \geq 2$

- **Tradeoff**: Shallower trees have benefits in parallelism,computation, and memory due to lower tree depth

- **Binary trees:** most efficient in proof size, but have larger depth, leading to prover memory overheads for large sizes.

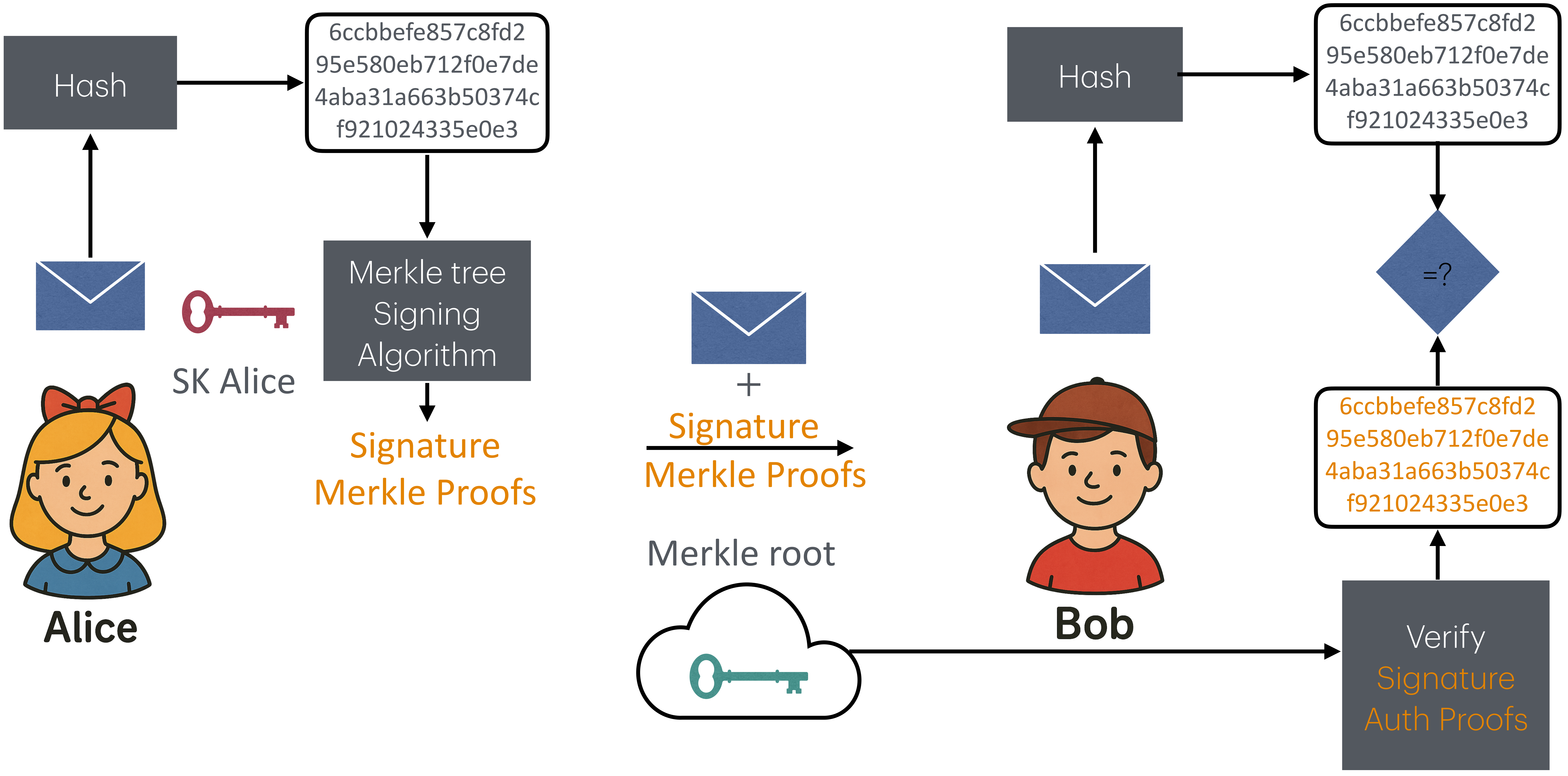
Merkle Tree: Multiple queries

- eg: Symmetric indices

Merkle Tree: Multiple queries

- Symmetric query indices: Bit reverse the vector indices at commitment

Exercise: Toy Merkle Tree digital signature scheme



https://github.com/ingonyama-zk/research_POCs/blob/course/course/examples/toy_digital_sig.rs