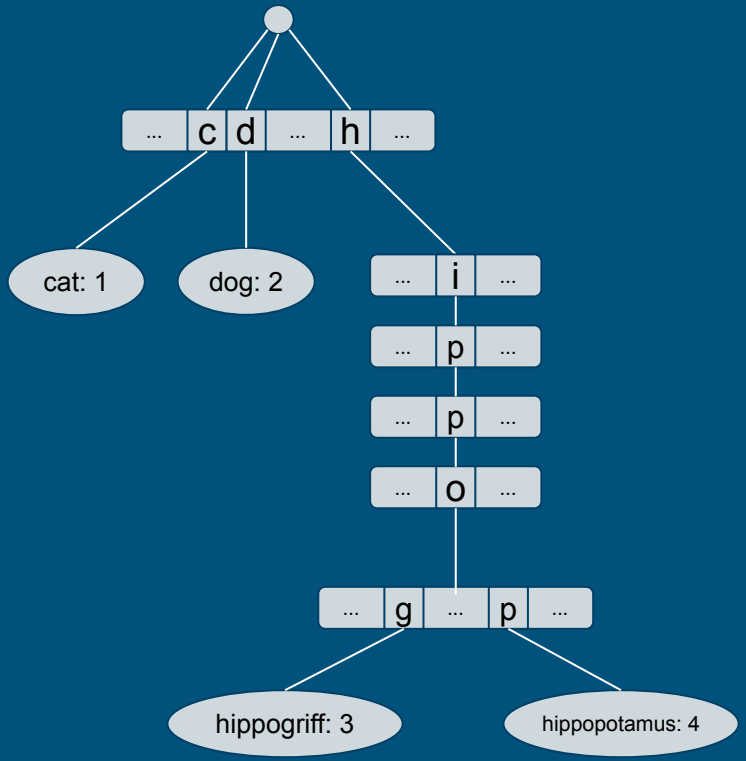
Verkle trees

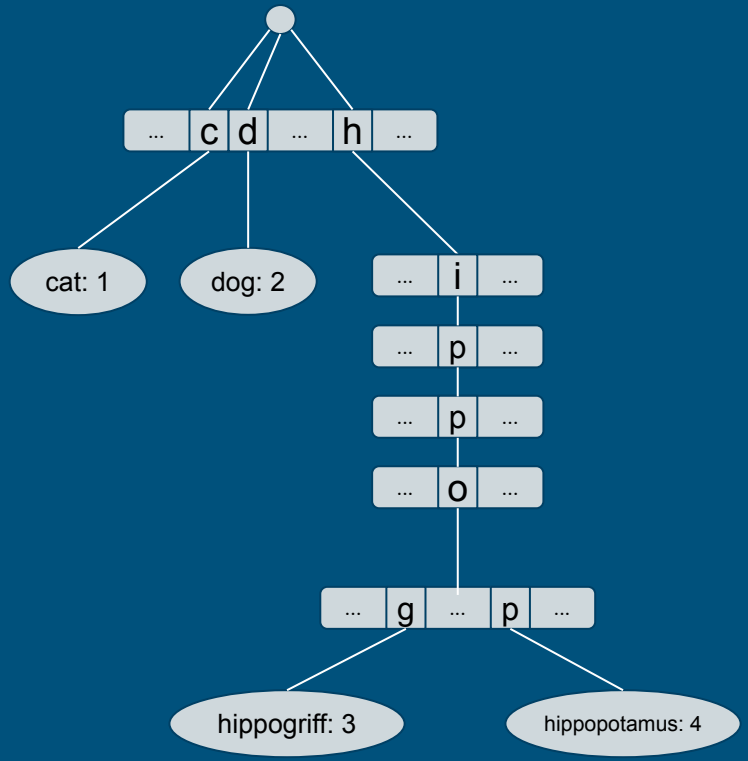
What is a Verkle tree?

- A data structure that combines the benefits of trees and cryptographic accumulators
- Idea: a tree of Kate commitments (eg. each with width 256)



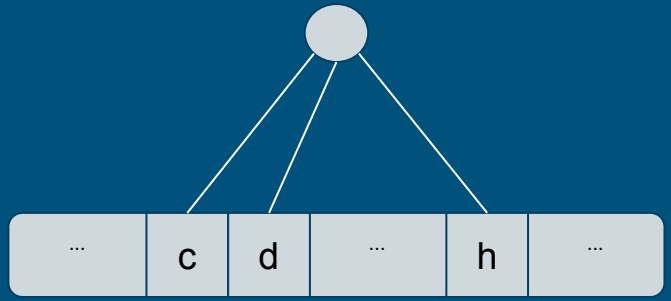
What is a Verkle tree?

- Assuming randomly distributed values, a path down to one value only takes $\log_{256}(n)$ nodes (so 8x fewer than a binary tree)
- Problem: but aren't each of those nodes 256 times bigger?
- This is where Kate commitments come in….



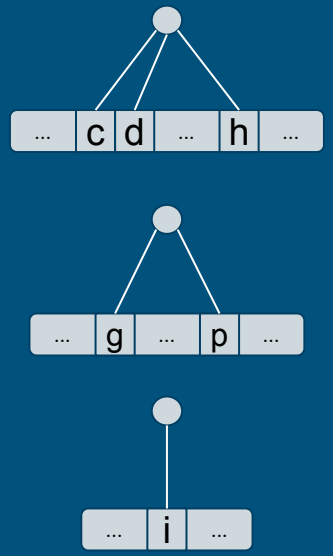
Kate commitments

- Use elliptic curves and polynomial math to create a data structure where the inclusion of *any* k of n values can be proven with a single (~48 byte) witness
- See description by Dankrad: https://dankradfeist.de/ethereum/2020/06/16/kate-polynomial-commitments.htm



Kate multi-proofs

- Given M commitments and N leaves spread anywhere across those commitments, you can *make a single proof* (<200 bytes) to prove all of these values
- Description of algorithm: https://notes.ethereum.org/nrQqhVpQRi6acQckwm1Ryg

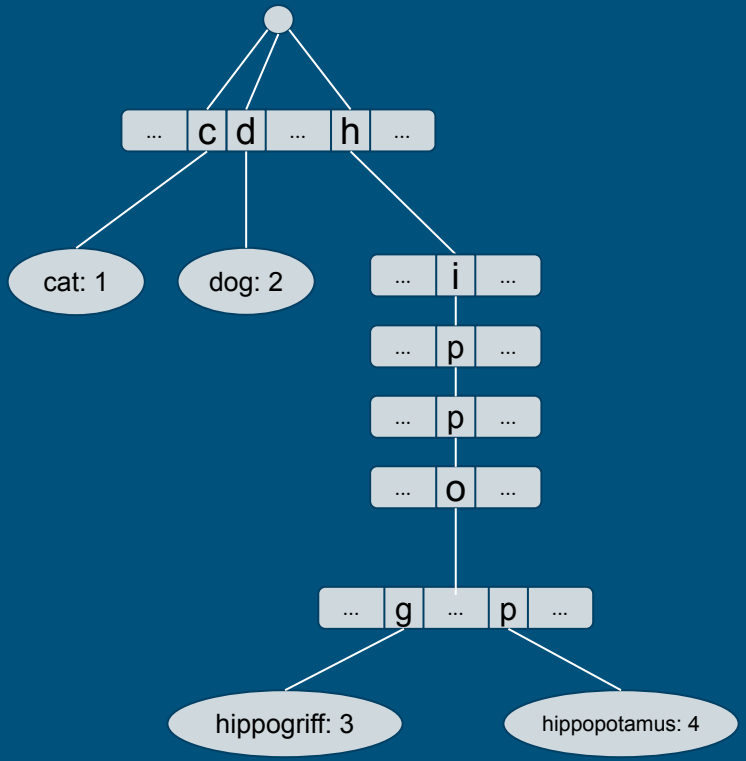


Why not just use one Kate commitment for the whole state?

- Problem 1: requires a very large trusted setup, which is insecure
- Problem 2: computing a witness takes O(N) time
  - You *can* precompute all N witnesses in O(N * log(N)) time, but then even a single state update requires recomputing the witness

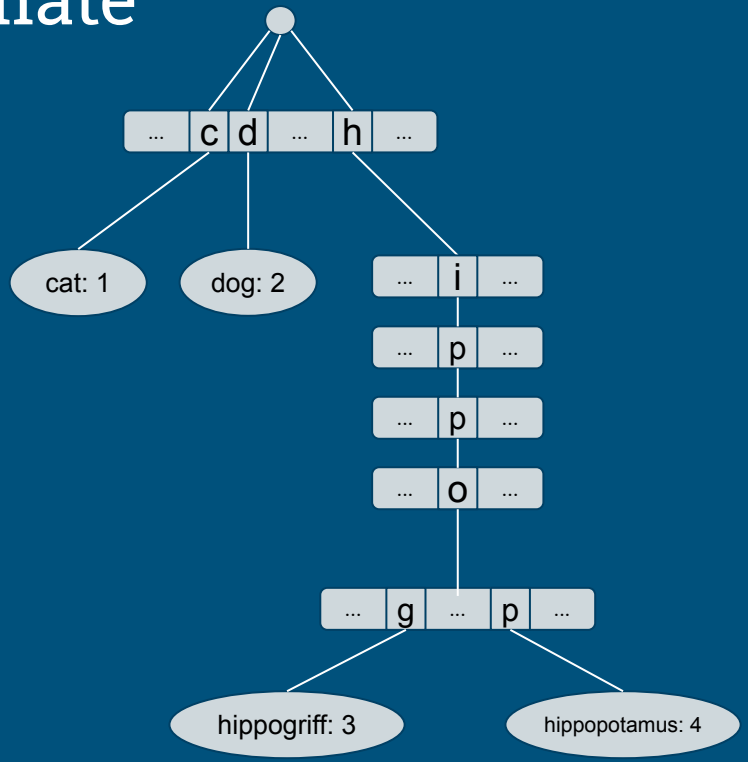
Witness sizes

- A Verkle tree witness requires:
  - The actual data
  - All intermediate Kate commitments (just the commitments)
  - One single proof (~200 bytes)



Witness sizes: example estimate

- 2^{32} state size, 5000 keys to be proven
  - Approx post-EIP-2929 max, not incl code
- Three levels of intermediate commitments
  - First level: all 256 commitments
  - Second level: ~ 4814 commitments (a bit of redundancy)
  - Third level: 5000 commitments
  - Total: ~10070 * 48 = 483k bytes
- Data: 320k bytes (keys+values)
- Proof: 200 bytes
- Total ~= 803k bytes



Verkle state trees

- Use a Verkle tree to store the state
- Easy to provide witnesses for stateless nodes
  - Witness verification can be done within a few hundred milliseconds
  - Witness generation currently takes up to ~3s but more optimization still pending
- How to store wide-node trees?
  - Pretend they're narrower-node trees at DB level
  - In-place updating, see PoC here: https://github.com/ethereum/research/blob/master/generic_in_place_tree/tree.py