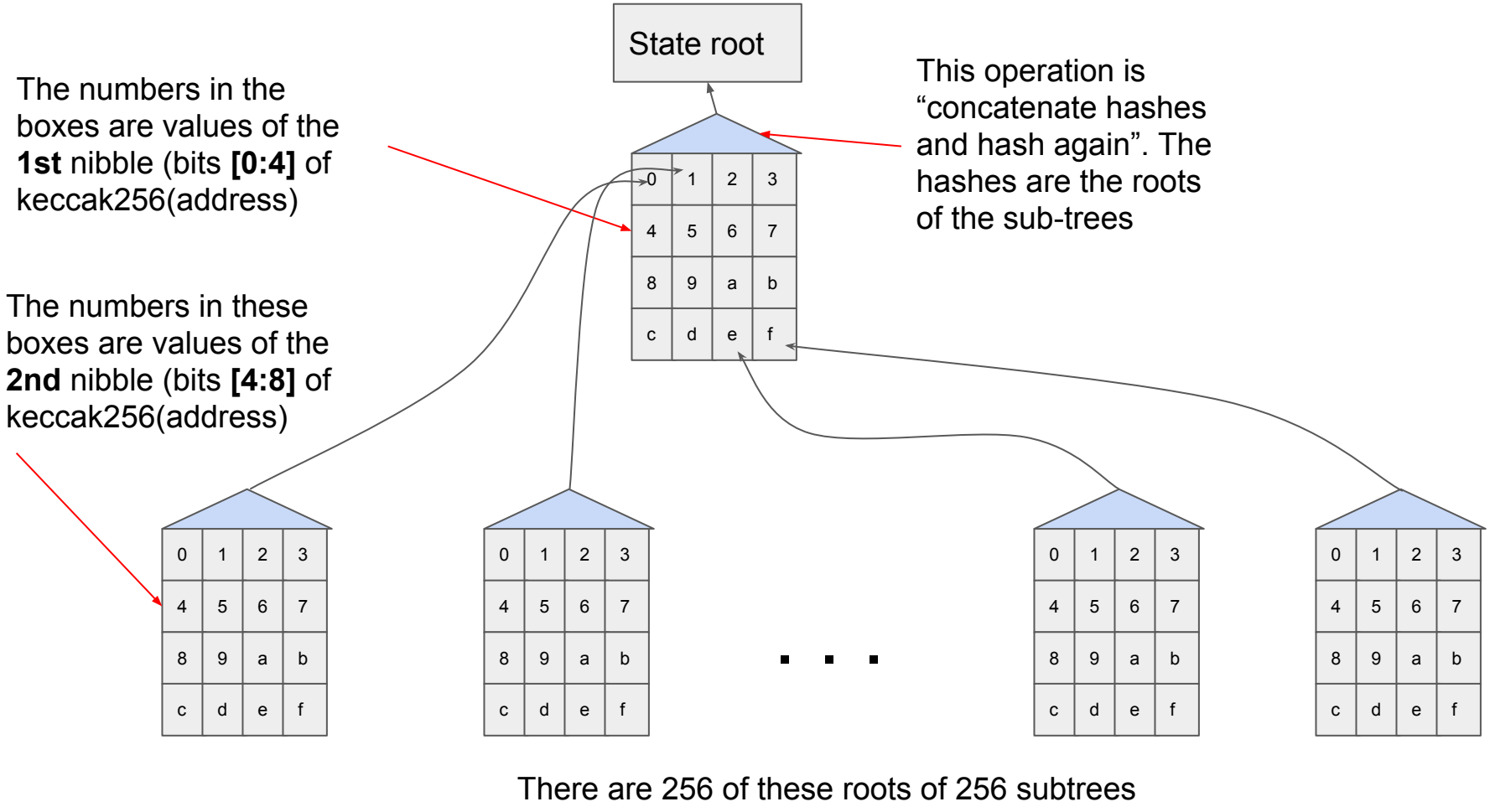
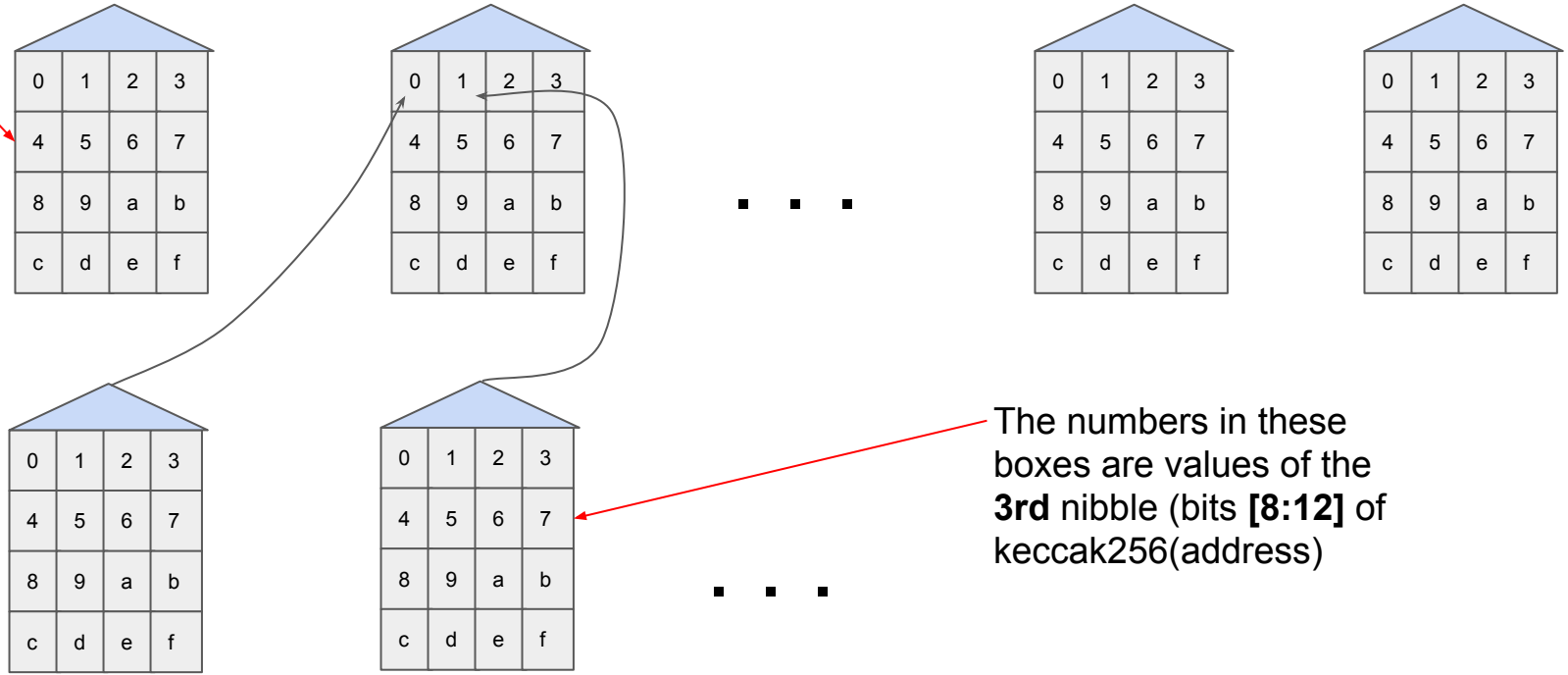

State root
The numbers in the boxes are values of the **1st** nibble (bits **[0:4]** of keccak256(address)
This operation is “concatenate hashes and hash again”. The hashes are the roots of the sub-trees
The numbers in these boxes are values of the **2nd** nibble (bits **[4:8]** of keccak256(address)
0 1 2 3
4 5 6 7
8 9 a b
c d e f
There are 256 of these roots of 256 subtrees

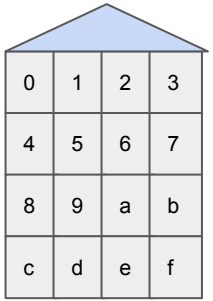
The numbers in these boxes are values of the **2nd** nibble (bits **[4:8]** of keccak256(address)

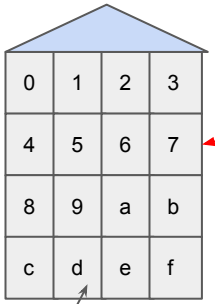
There are 256 of these roots of 256 subtrees



The numbers in these boxes are values of the **3rd** nibble (bits **[8:12]** of keccak256(address)

There are 4096 of these roots of 4096 subtrees

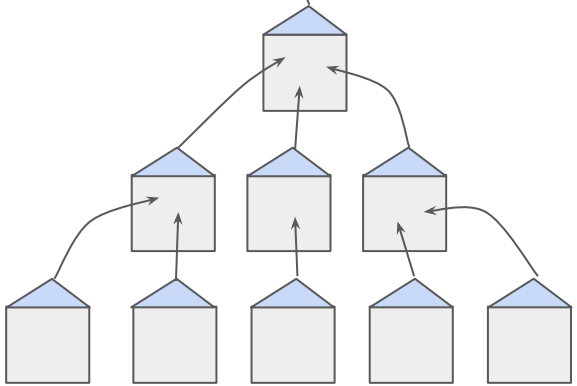




The numbers in these boxes are values of the **3rd** nibble (bits **[8:12]** of keccak256(address)

. . .

There are 4096 of these roots of 4096 subtrees



This subtree can be constructed from a collection of all accounts with keccak256(address) starting with 0x11d as their first 12 bits (3 nibbles).

This is one chunk. There are 4096 chunks.

Client receiving a chunk, can construct the subtree and compute the hash root of the chunk. In order to verify that this chunk is indeed in the tree (and in the correct position), it requires the following:

1. All sibling hashes on level 3 (in our example, hash roots of subtrees 0x110, 0x111, 0x112, …, 0x1c, 0x1e, 0x1f). Size: 15 * 32 = 480 bytes
2. All sibling hashes on level 2 (in our example, hash roots of subtrees 0x10, 0x12, 0x13, 0x14, …, 0x1f). Size: 15 * 32 = 480 bytes
3. All sibling hashes on level 1 (in our example, hash roots of subtrees 0x0, 0x2, 0x3, 0x4, …, 0xf). Size: 15 * 32 = 480 bytes

Total size of such proof is 480+480+480 = 1140 bytes per chunk.

Alternatively, the receiving client can first receive all 4096 subtree hash roots upfront, so that no further chunk proofs are necessary. This would require receiving 4096*32 = 131072 bytes upfront, but saves sending 4096*1140 = 4’669’440 bytes of proofs with each chunk.

With the current state size estimates, the size of each chunk would be around 1-2 Mb. Adding level 4 into the scheme would reduce the chunk size to around 200 kilobytes. In this scenario, one can send 4096 subtree hash roots upfront, and then attach extra 480 bytes of proof (15 sibling hashes on level 4) with each chunk.

Storage of large contracts can be transmitted in a similar way, with perhaps fewer levels. Storage of small contracts can be simply packed in a single chunk, or multiple contracts per chunk