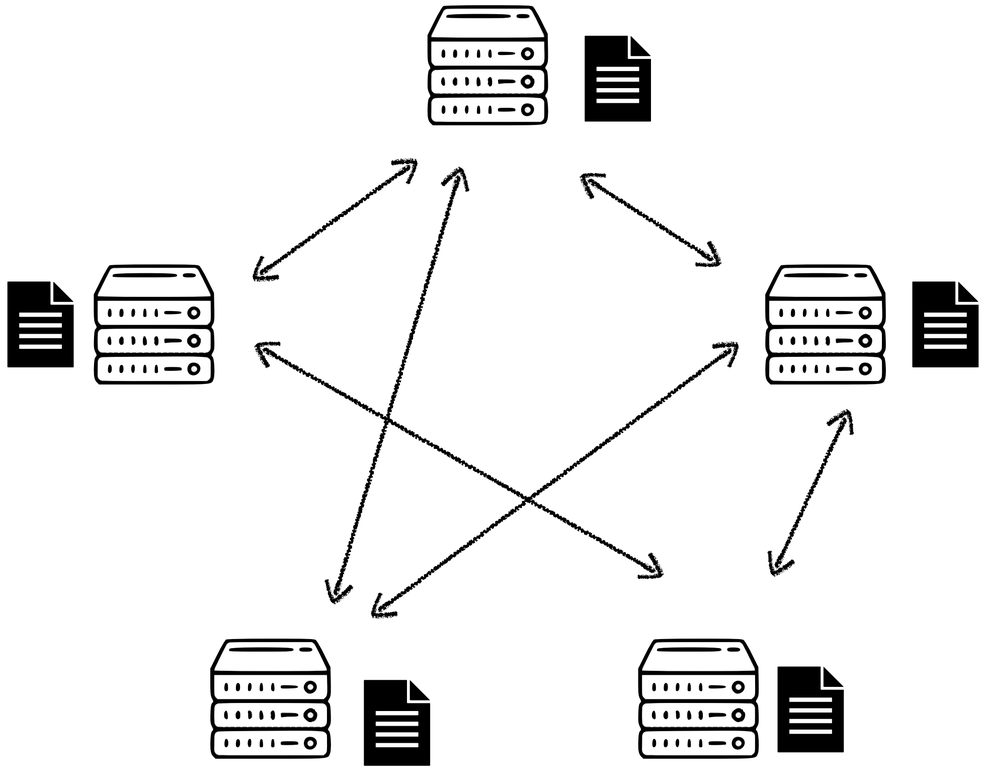
Superlight Client for Proof of Stake Ethereum

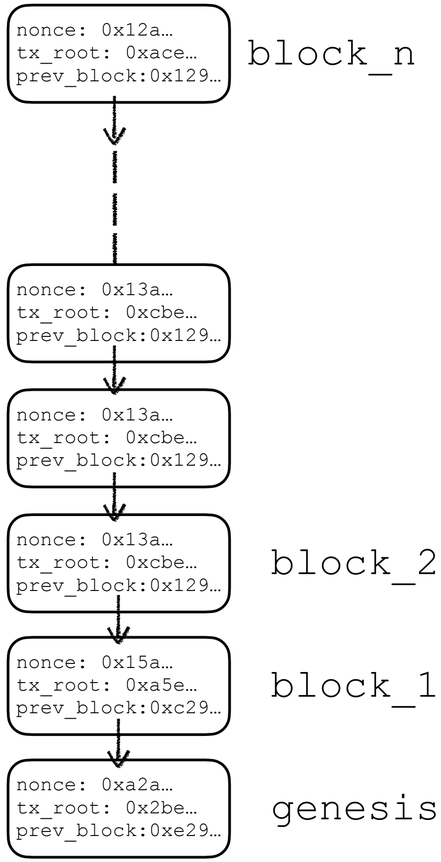
Research Supervisor: Dionysis Zindros

Internal Supervisor: Sören Petrat

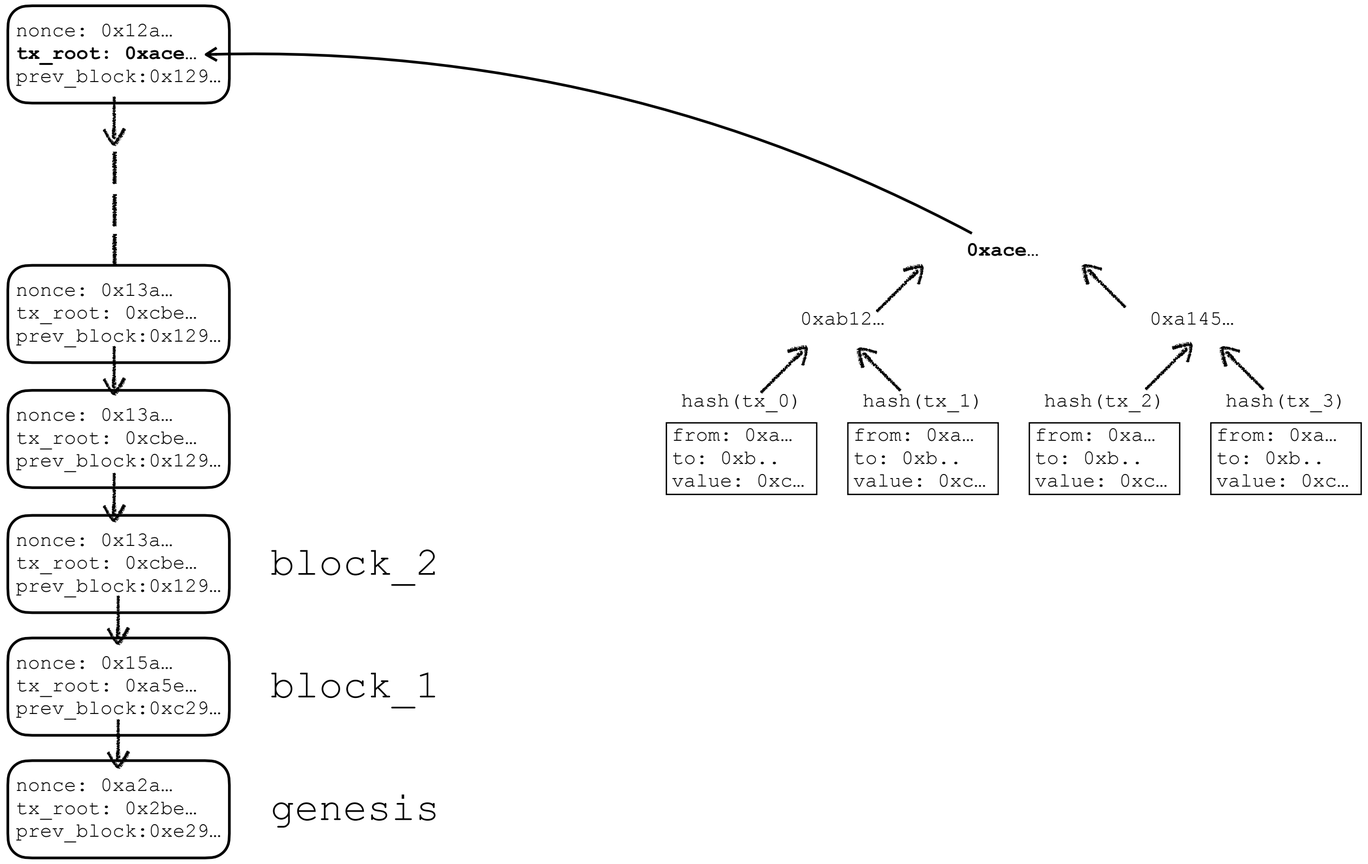
Blockchain



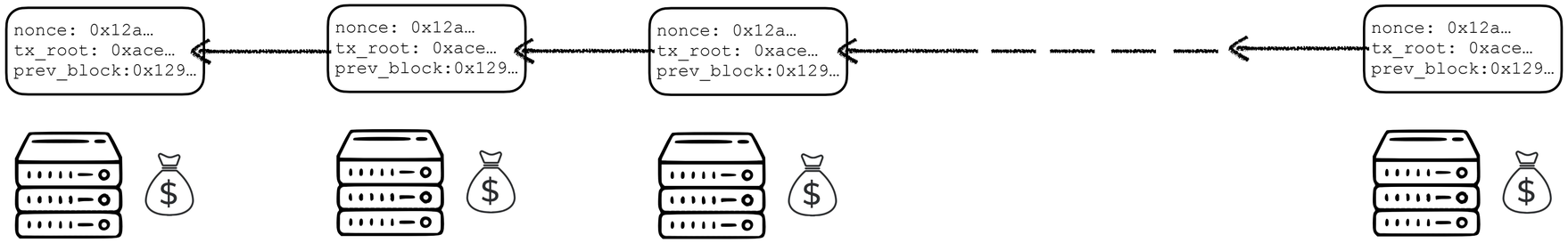
Blockchain


nonce: 0x12a…
tx_root: 0xace…
prev_block:0x129…
block_n
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
block_2
nonce: 0x15a…
tx_root: 0xa5e…
prev_block:0xc29…
block_1
nonce: 0xa2a…
tx_root: 0x2be…
prev_block:0xe29…
genesis

Blockchain

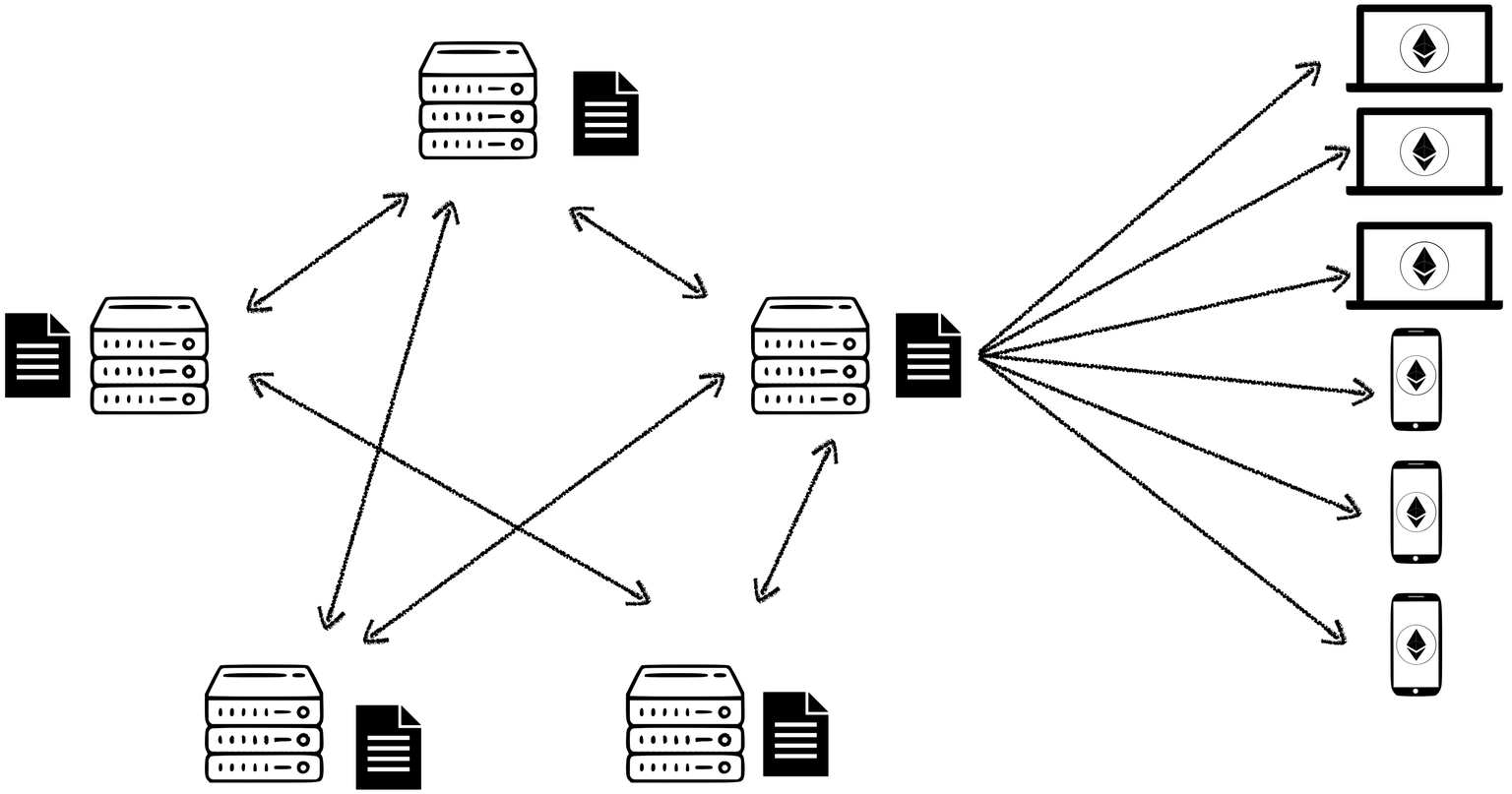

nonce: 0x12a…
tx_root: 0xace…
prev_block:0x129…
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
nonce: 0x13a…
tx_root: 0xcbe…
prev_block:0x129…
block_2
nonce: 0x15a…
tx_root: 0xa5e…
prev_block:0xc29…
block_1
nonce: 0xa2a…
tx_root: 0x2be…
prev_block:0xe29…
genesis
0xace…
0xab12…
0xa145…
hash(tx_0)
hash(tx_1)
hash(tx_2)
hash(tx_3)
from: 0xa…
to: 0xb..
value: 0xc…
from: 0xa…
to: 0xb..
value: 0xc…
from: 0xa…
to: 0xb..
value: 0xc…
from: 0xa…
to: 0xb..
value: 0xc…

Proof of Stake (PoS)



Proof of Stake Ethereum

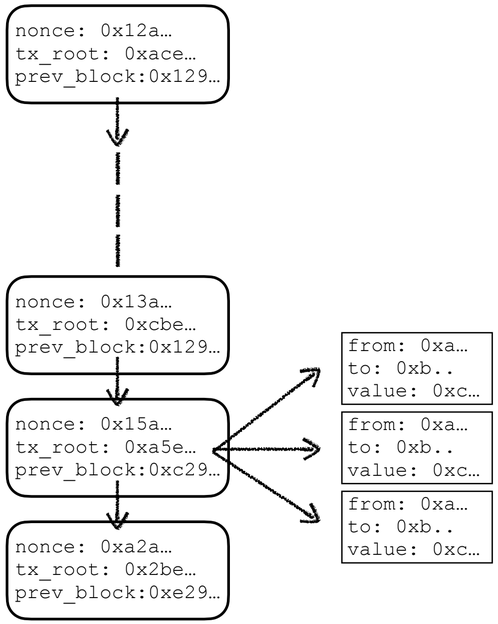
Centralisation of providers



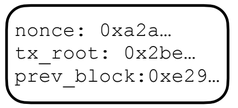
How can me make wallets secure?

Fullnode

fullnode



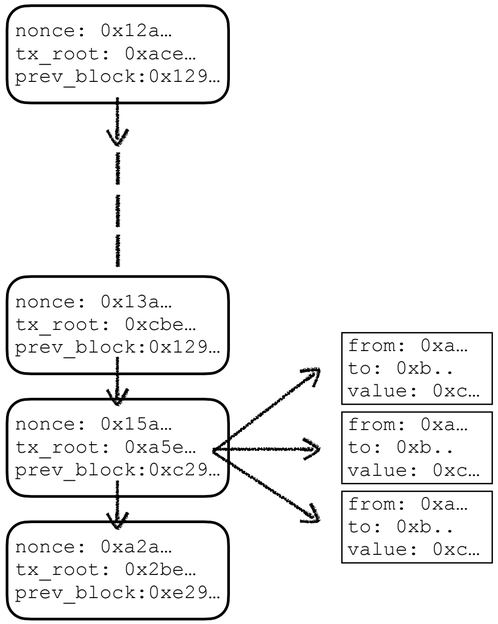
fullnode bootstrapping



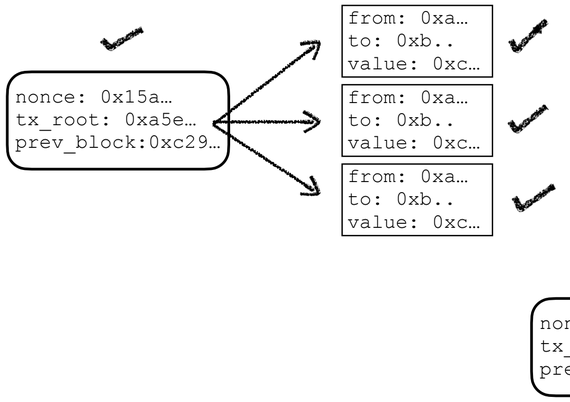
genesis block

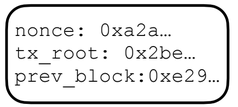
Fullnode

Fullnode



fullnode bootstrapping

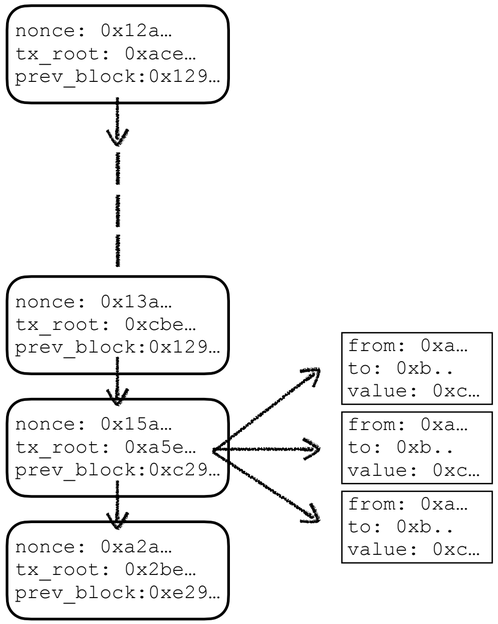




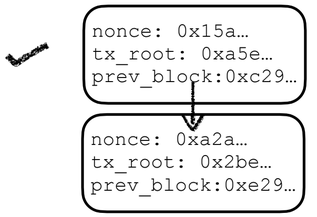
genesis block

Fullnode

fullnode



fullnode bootstrapping



genesis block

Time & space complexity of Fullnode sync

$$O(|n| + |t|)$$

$|n|$ is number of blocks

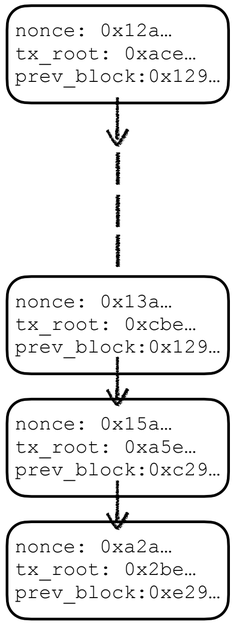
$|t|$ is number of transactions

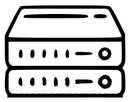
Ethereum Fullnode requires >500GB SSD and 25Mb/s network bandwidth

https://ethereum.org/en/developers/docs/nodes-and-clients

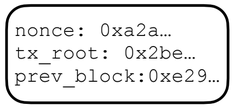
Light Client (SPV)

fullnode





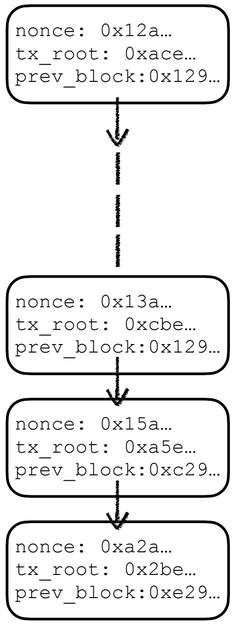
light client bootstrapping

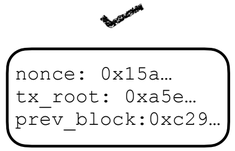


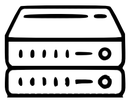
genesis block

Light Client (SPV)

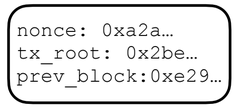
fullnode







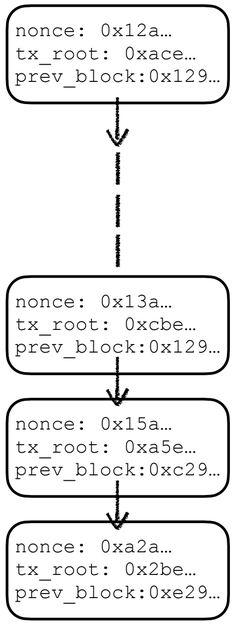
light client bootstrapping

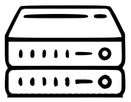


genesis block

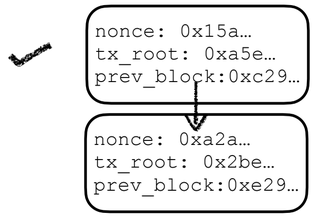
Light Client (SPV)

fullnode





light client bootstrapping



genesis block

Time & space complexity of Light Client sync

$$O(|n|)$$

$|n|$ is number of blocks

PoW Ethereum Light Client requires >400MB storage

Can we do better?

Superlight clients 👏

Contributions

1. Evaluation of the official light client specification based on sync protocol with respect to the security of underlying consensus.

Contributions

1. Evaluation of the official light client specification based on sync protocol with respect to the security of underlying consensus.

2. Propose a construction for building superlight clients for Proof of Stake Ethereum which is exponential better that the official specification and doesn’t require any hard fork.

Contributions

1. Evaluation of the official light client specification based on sync protocol with respect to the security of underlying consensus.

2. Propose a construction for building superlight clients for Proof of Stake Ethereum which is exponential better that the official specification and doesn’t require any hard fork.

3. Presents the first implementation of super light clients for Proof of Stake blockchains. This implementations can be easily extended to production grade code which can used by wallets.

Contributions

1. Evaluation of the official light client specification based on sync protocol with respect to the security of underlying consensus.

2. Propose a construction for building superlight clients for Proof of Stake Ethereum which is exponential better that the official specification and doesn’t require any hard fork.

3. Presents the first implementation of super light clients for Proof of Stake blockchains. This implementations can be easily extended to production grade code which can used by wallets.

4. Benchmark the light client w.r.t to the superlight client with real network conditions to compare storage, bandwidth and interactivity requirements.

Contributions

1. Evaluation of the official light client specification based on sync protocol with respect to the security of underlying consensus.

2. Propose a construction for building superlight clients for Proof of Stake Ethereum which is exponential better that the official specification and doesn’t require any hard fork.

3. Presents the first implementation of super light clients for Proof of Stake blockchains. This implementations can be easily extended to production grade code which can used by wallets.

4. Benchmark the light client w.r.t to the superlight client with real network conditions to compare storage, bandwidth and interactivity requirements.

Verification PoS

- To verify blocks in PoS system we need to know the correct block proposer

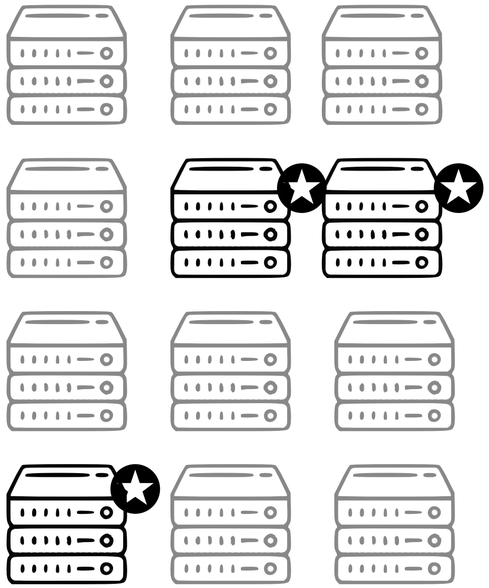
Verification PoS

- To verify blocks in PoS system we need to know the correct block proposer
- To know the block proposer we need to:
  - Stake of each validator
  - Random value used for selection

Verification PoS

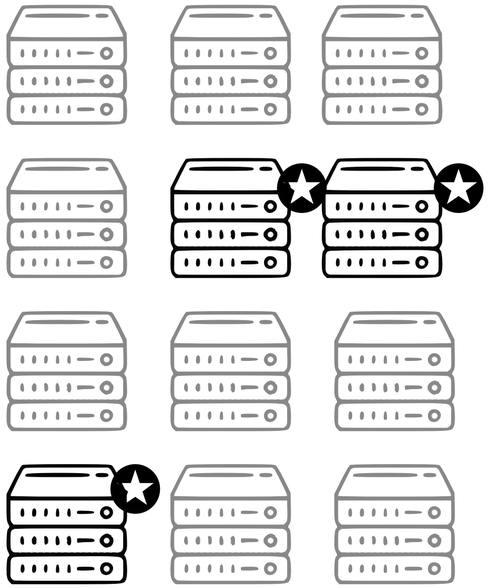
- To verify blocks in PoS system we need to know the correct block proposer
- To know the block proposer we need to:
  - Stake of each validator
  - Random value used for selection
- Stake table is difficult to verify without accessing the complete table

Sync Committee



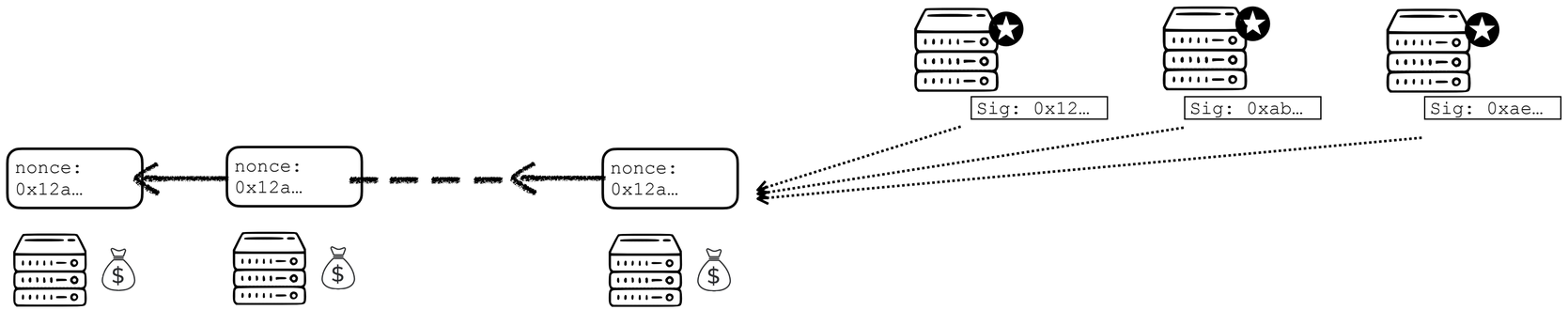
Set of 512 randomly selected validator
from the validator set

Sync Committee

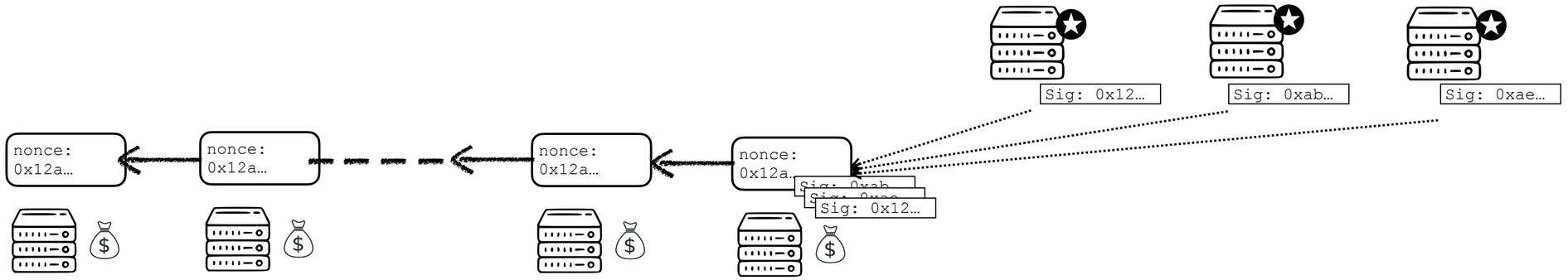


Responsible to sign the tip of the chain
for Sync Period (~1 day)

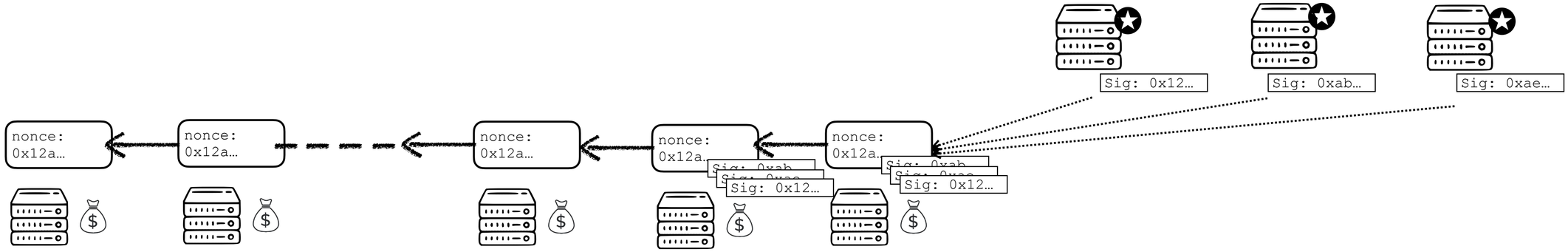
Sync Committee



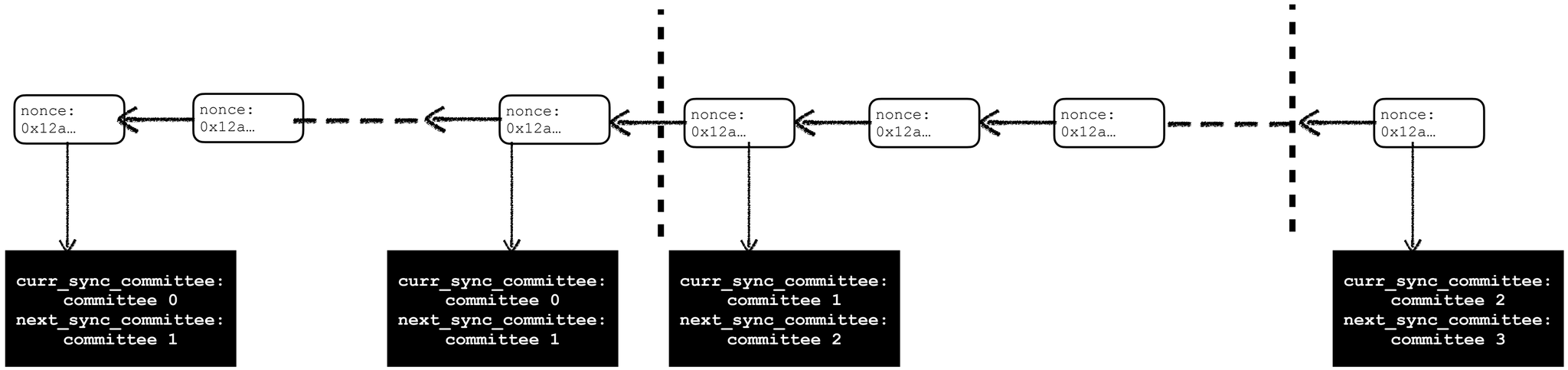
Sync Committee



Sync Committee



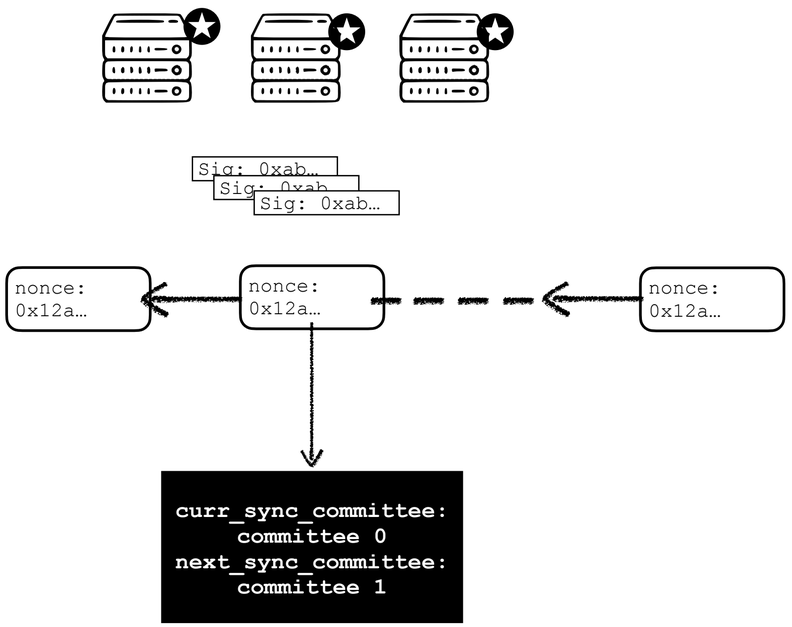
Sync Committee



Genesis Period *Sync Period 1* *Sync Period 2*

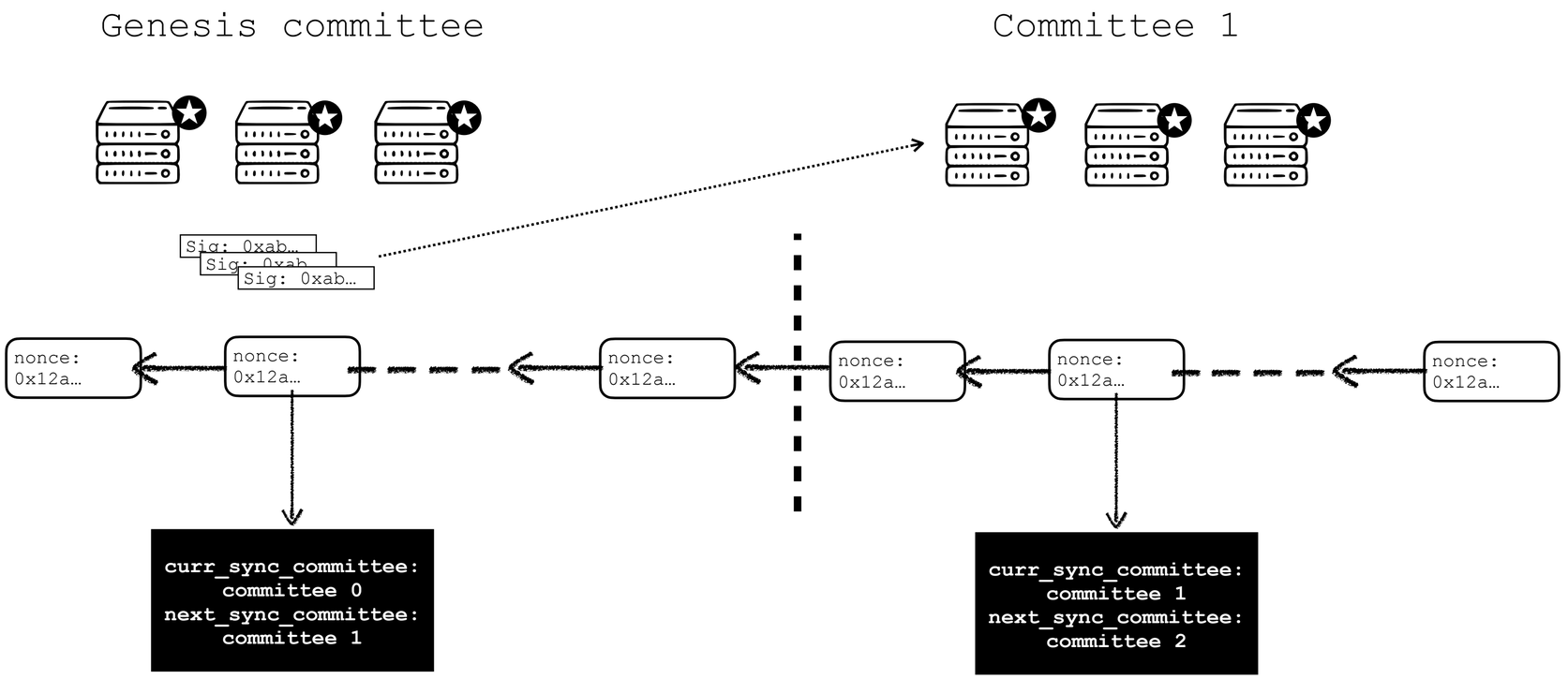
Sync Committee

Genesis committee

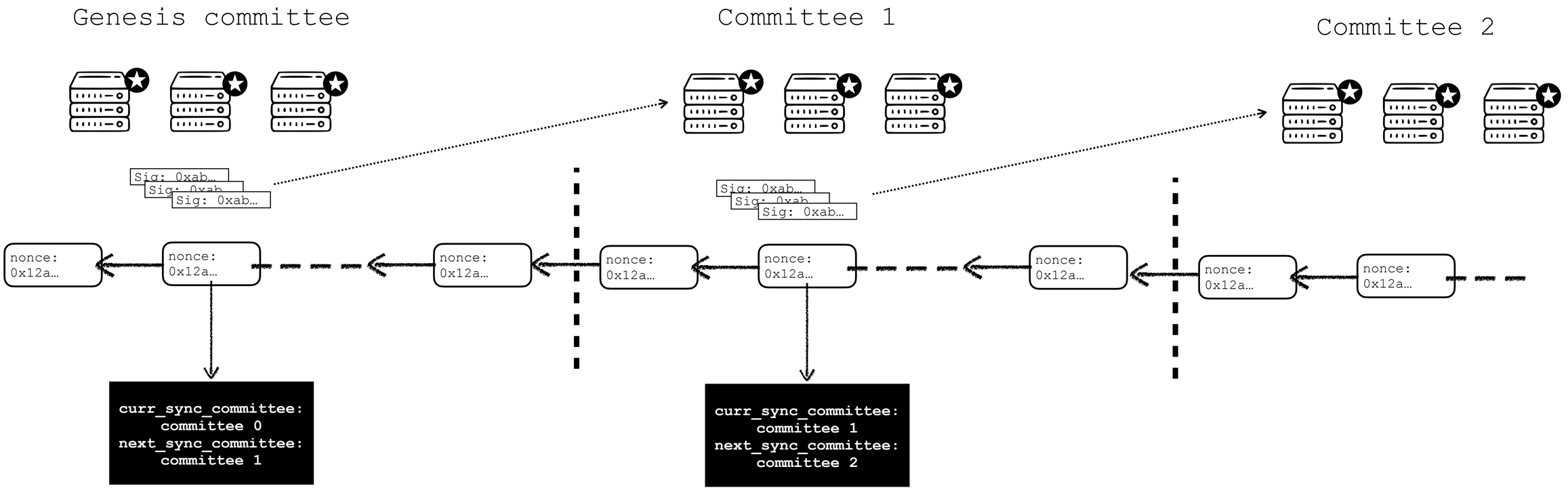


Sync Period 0

Sync Committee

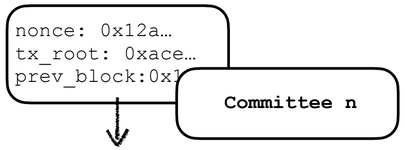


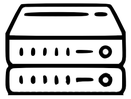
Sync Committee



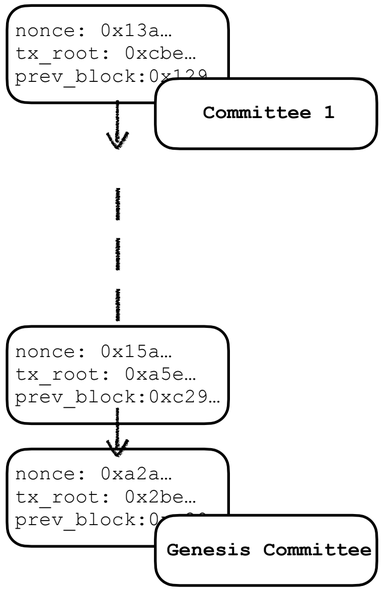
Ethereum PoS Light Client

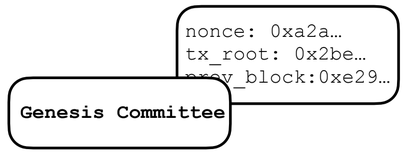
fullnode





light client bootstrapping

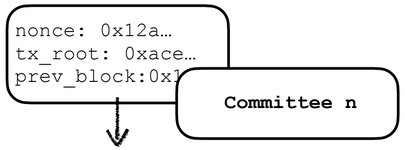


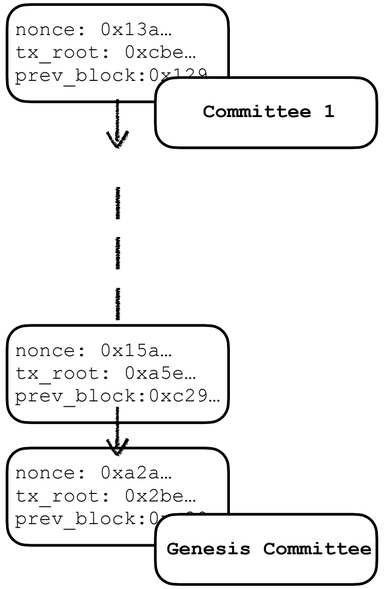


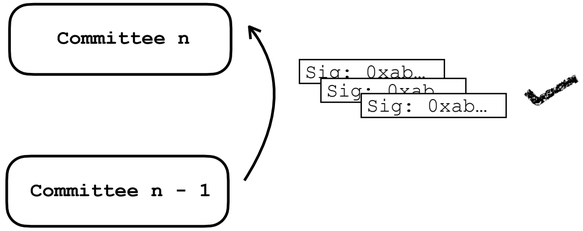
genesis block

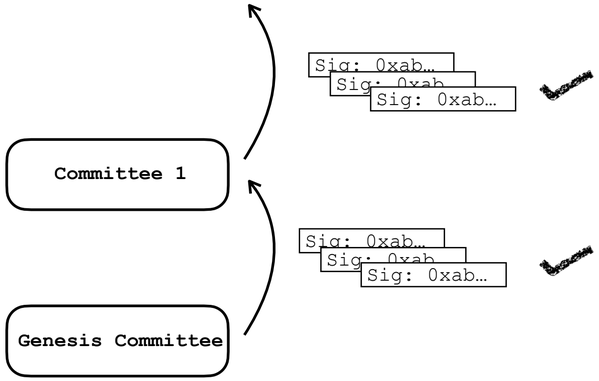
Ethereum PoS Light Client

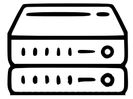
fullnode



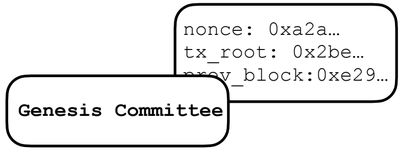








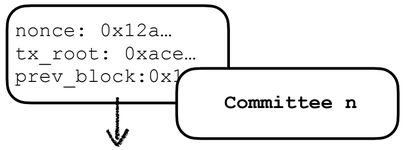
light client bootstrapping

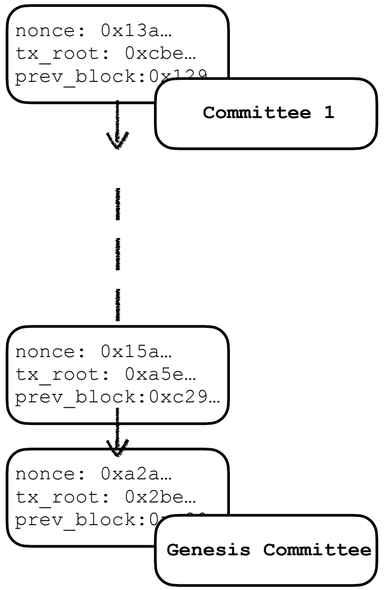


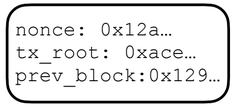
genesis block

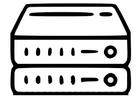
Ethereum PoS Light Client

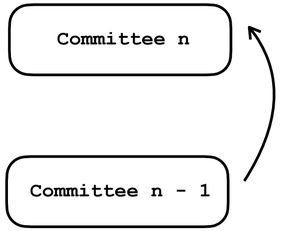
fullnode



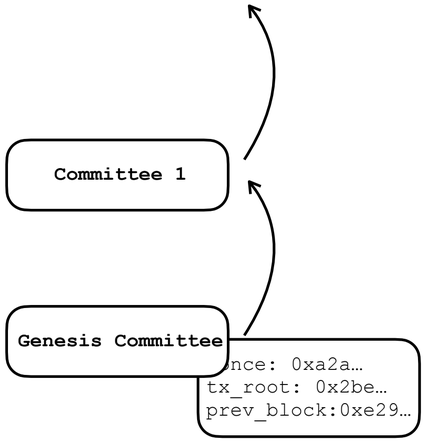








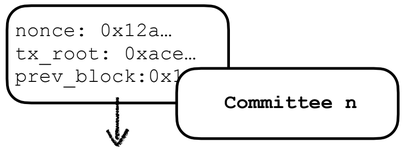
light client bootstrapping

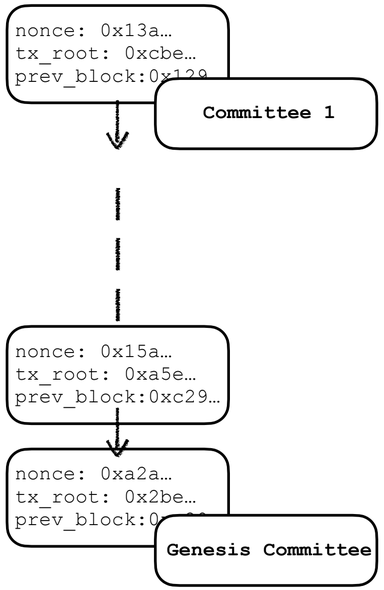


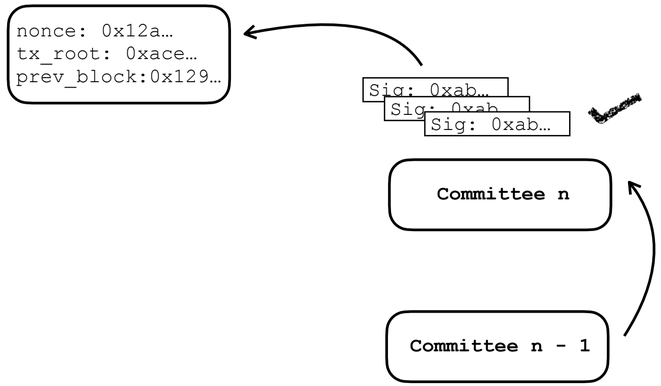
genesis block

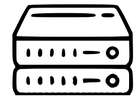
Ethereum PoS Light Client

fullnode

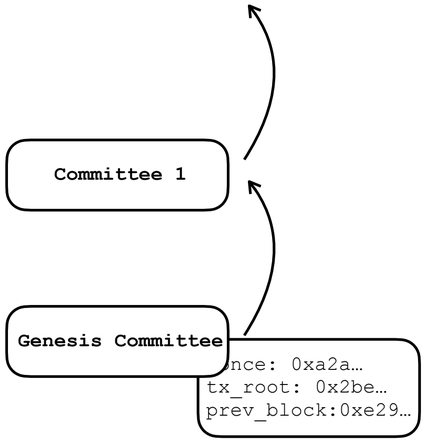








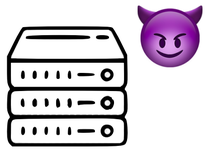
light client bootstrapping



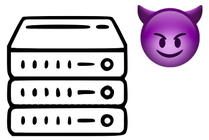
genesis block

Lets build Superlight client

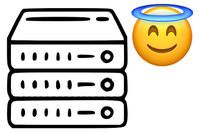
Non eclipsing assumption



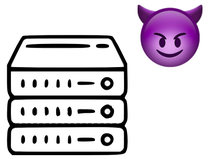
fullnode 1



fullnode 2

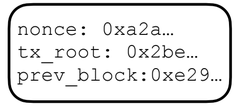


fullnode k



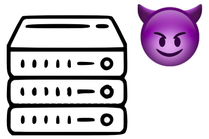
fullnode n

superlight client bootstrapping

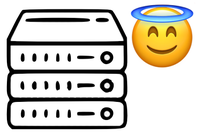


genesis block

Base case 2 provers

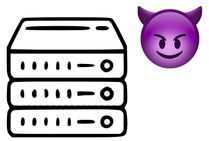


fullnode alice



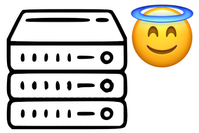
fullnode bob

Base case 2 provers



fullnode alice

Genesis Committee | Committee 1 | Committee 2 | - - - - - - - - - - - - | Committee n'

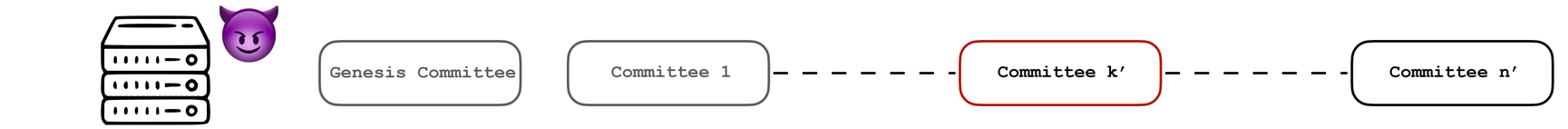


fullnode bob

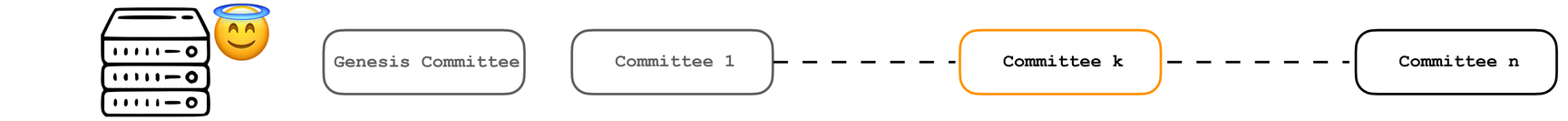
Genesis Committee | Committee 1 | Committee 2 | - - - - - - - - - - - - | Committee n

Base case 2 provers

Find first point of disagreement



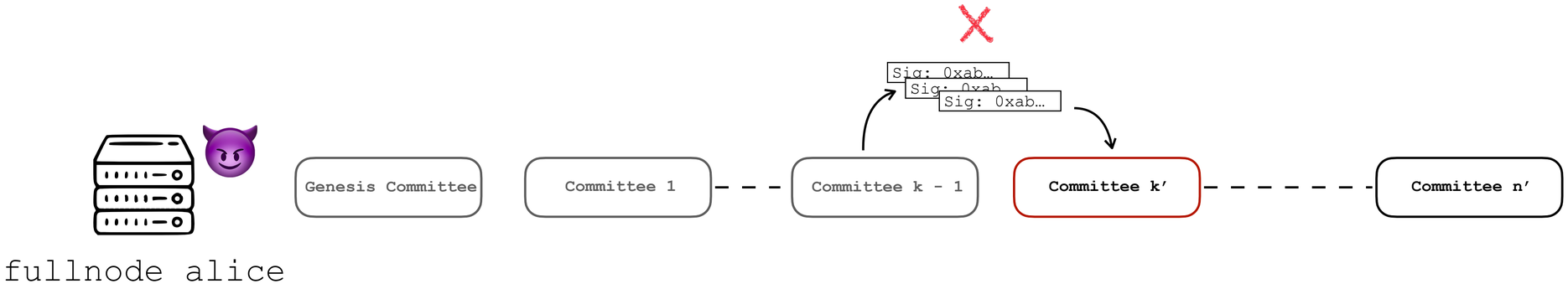
fullnode alice

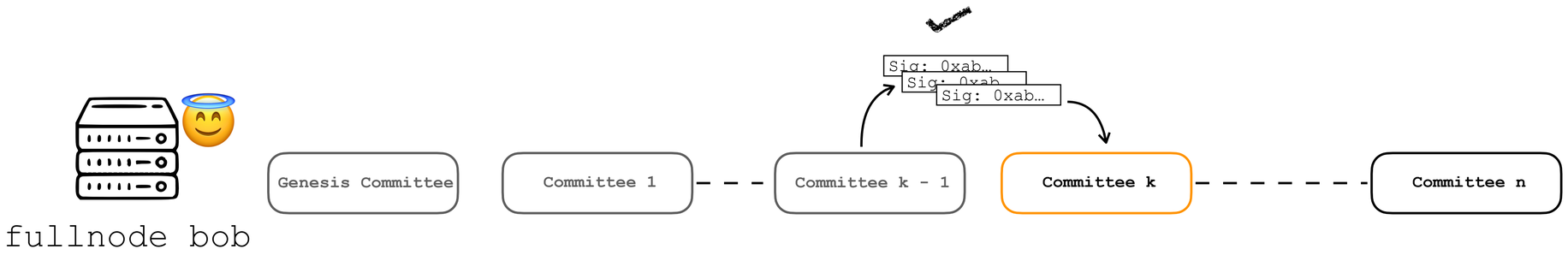


fullnode bob

Base case 2 provers

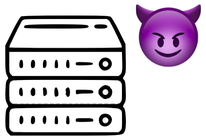
Check sync committee signatures



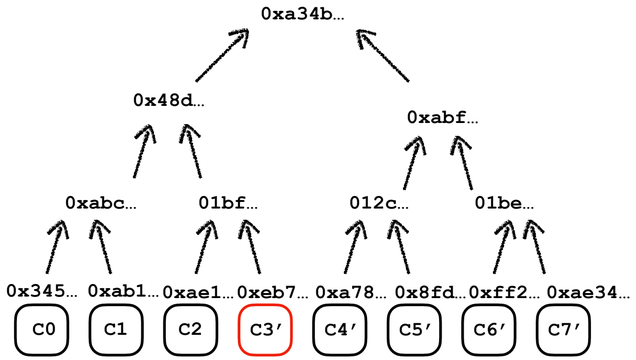


How to find first point of disagreement efficiently?

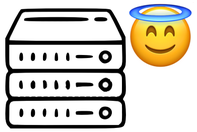
Interactive bisection game



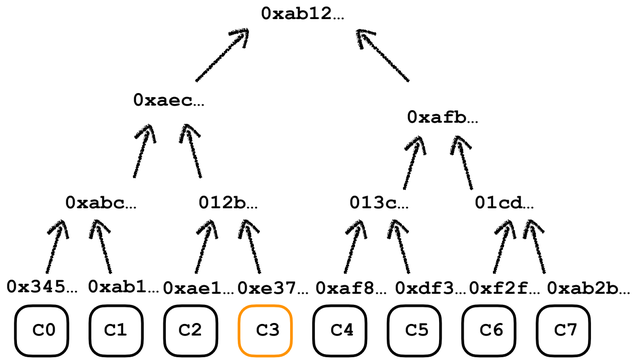
fullnode alice



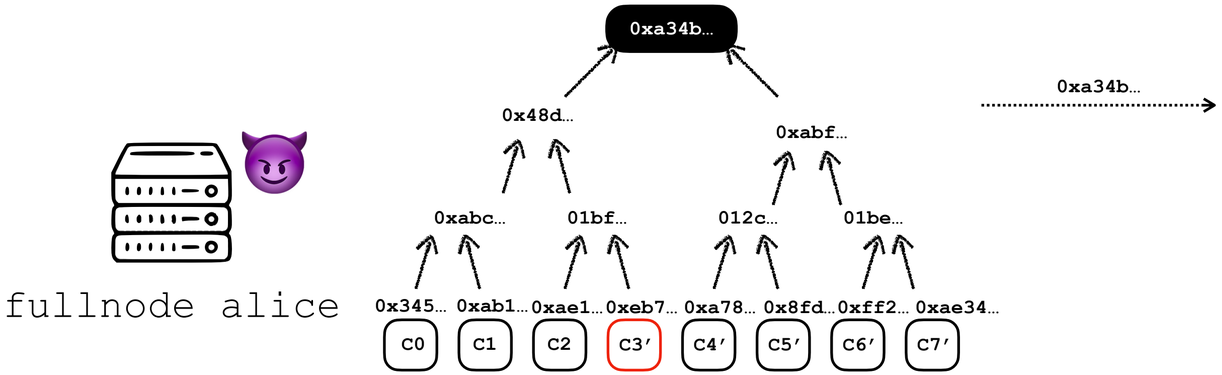
superlight client bootstrapping



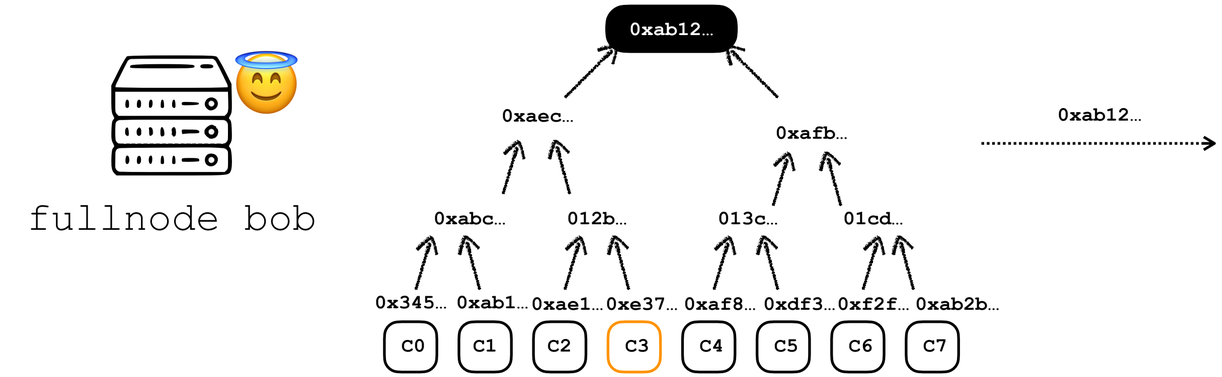
fullnode bob



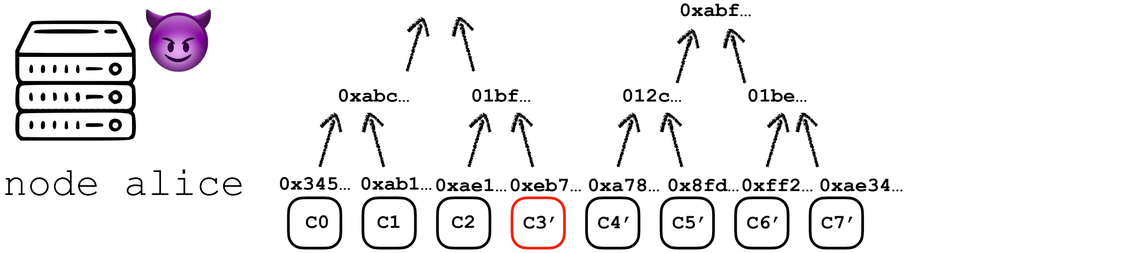
Interactive bisection game



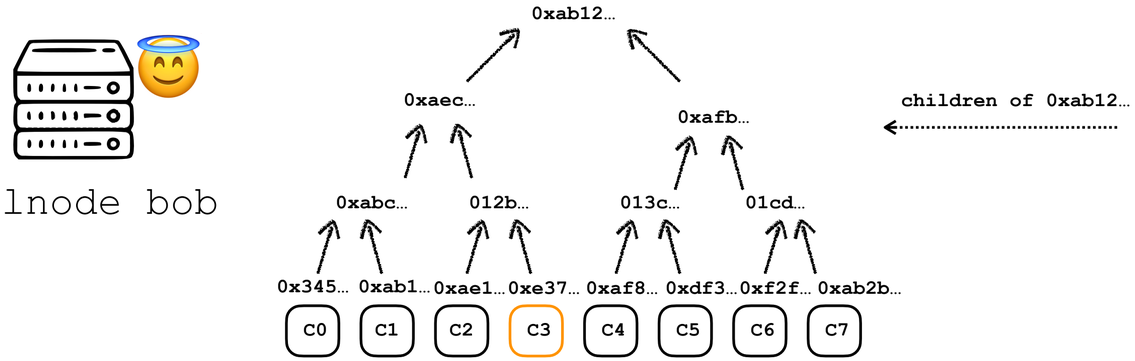
superlight client bootstrapping



Interactive bisection game



fullnode alice



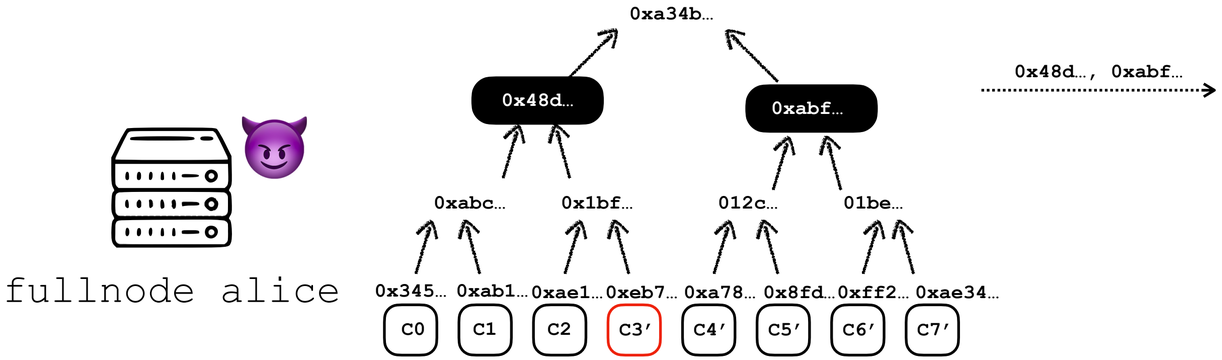
fullnode bob

superlight client bootstrapping

0xa34b...

0xab12...

Interactive bisection game

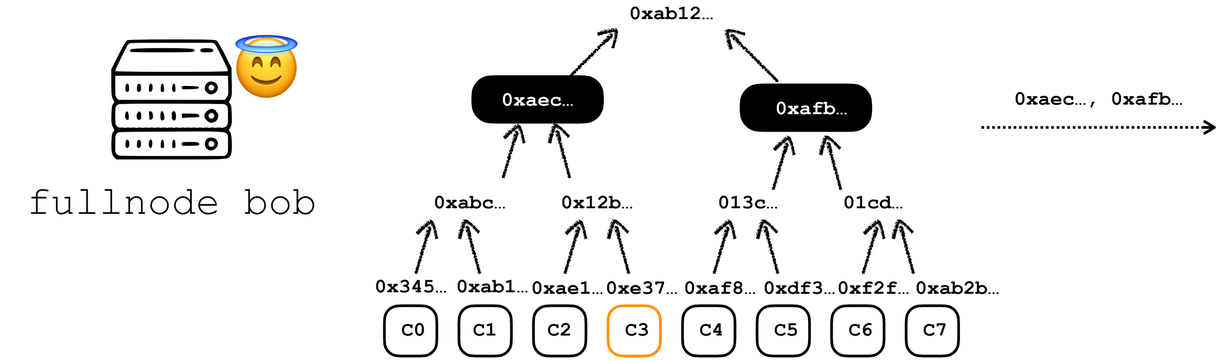


0x48d…, 0xabf…

superlight client bootstrapping

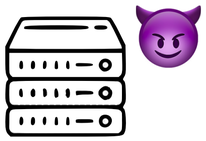
0xa34b…

0xab12…

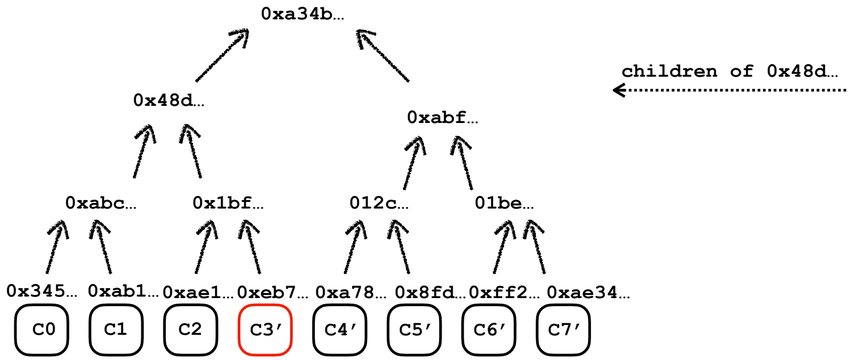


0xaec…, 0xafb…

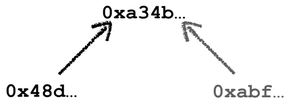
Interactive bisection game

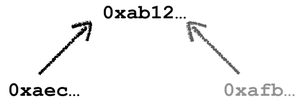


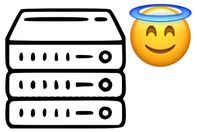
fullnode alice



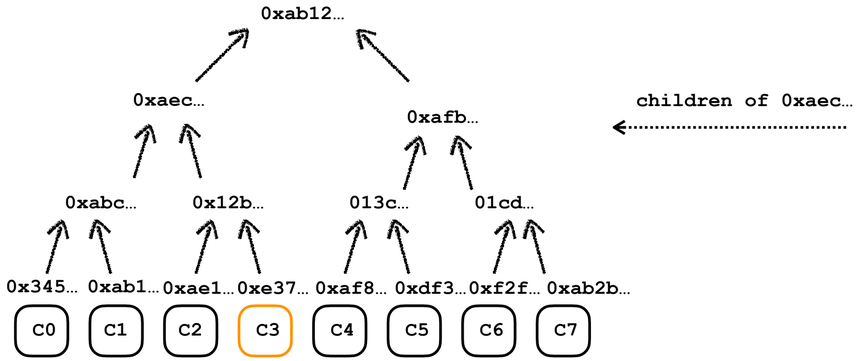
superlight client bootstrapping



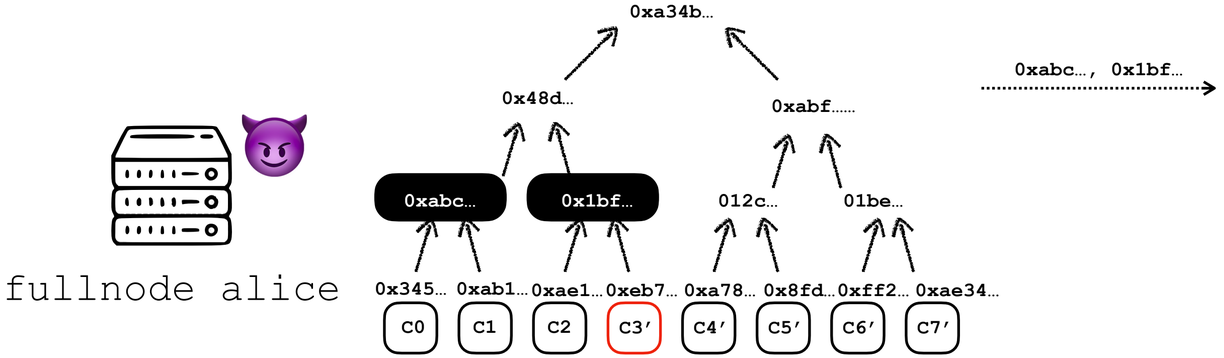


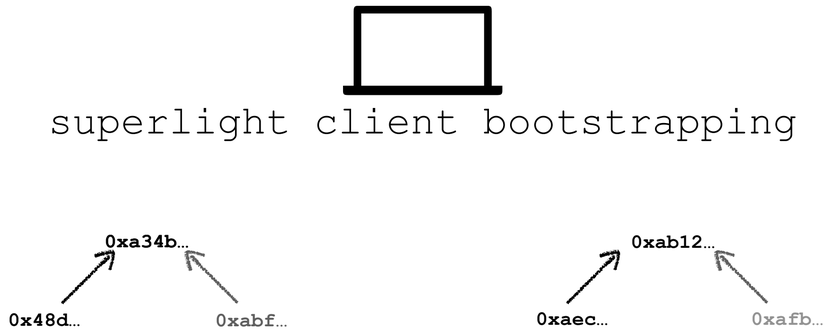


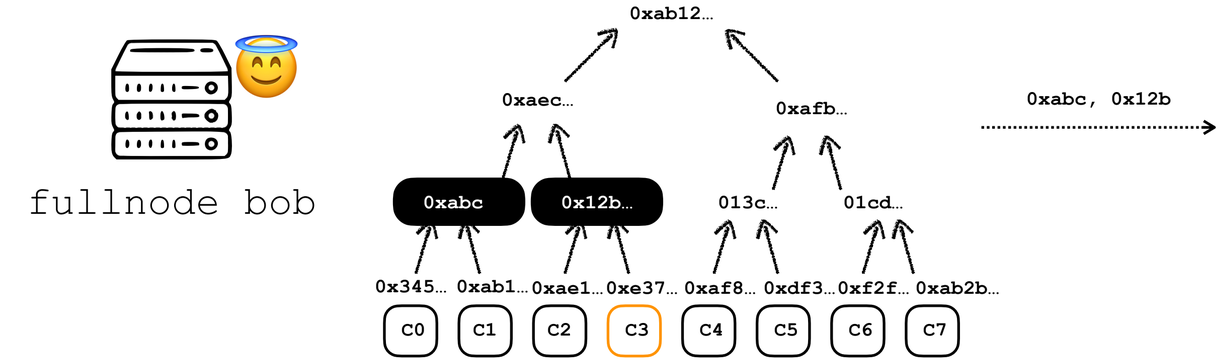
fullnode bob



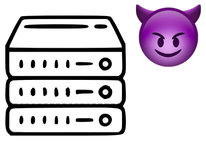
Interactive bisection game



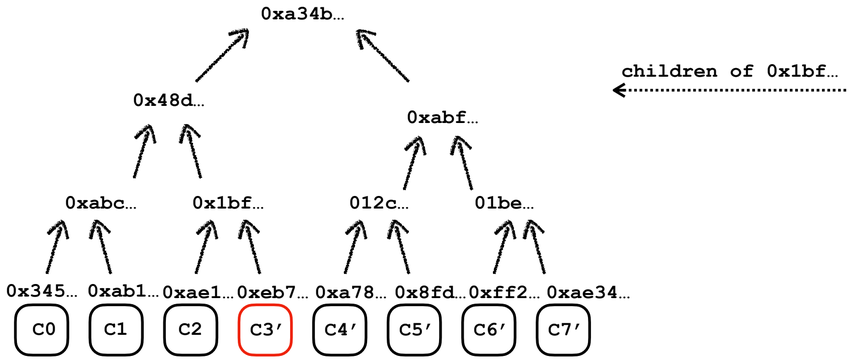




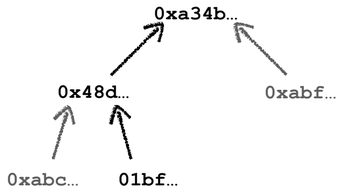
Interactive bisection game

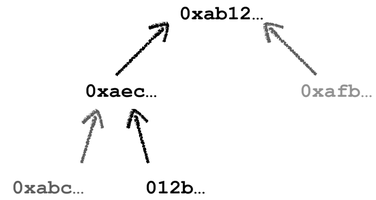


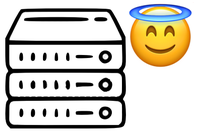
fullnode alice



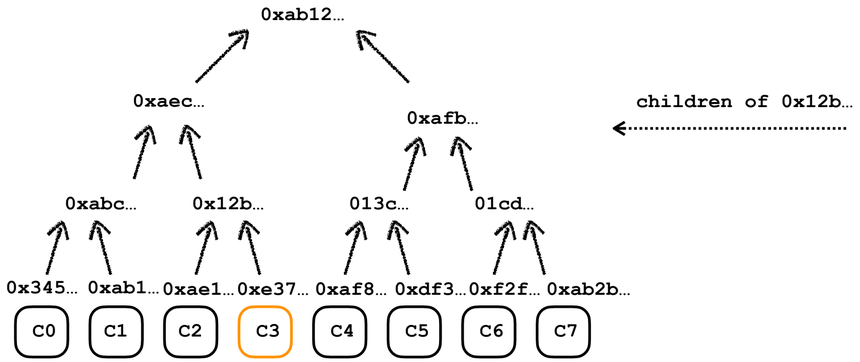
superlight client bootstrapping



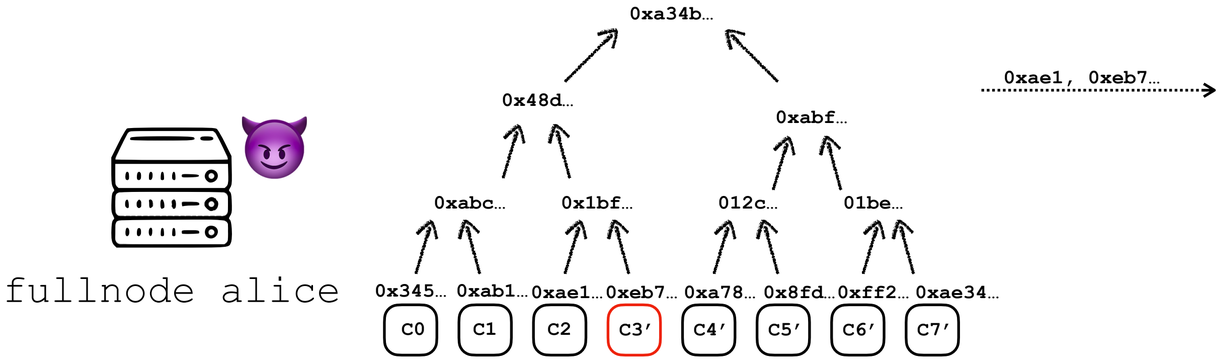




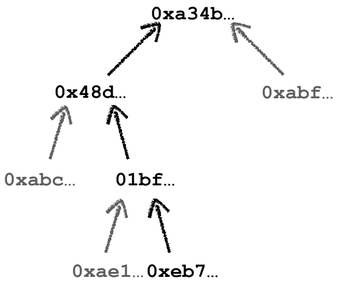
fullnode bob

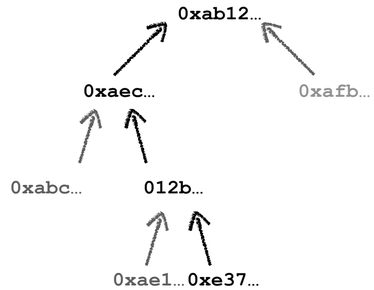


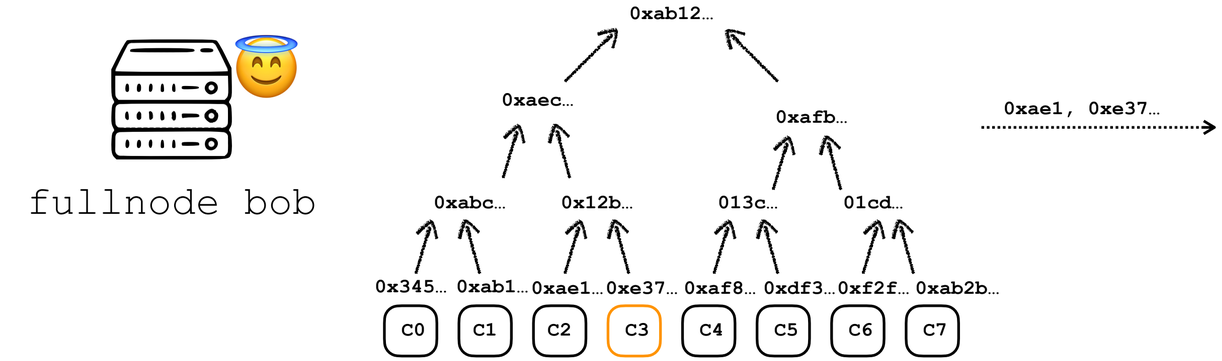
Interactive bisection game



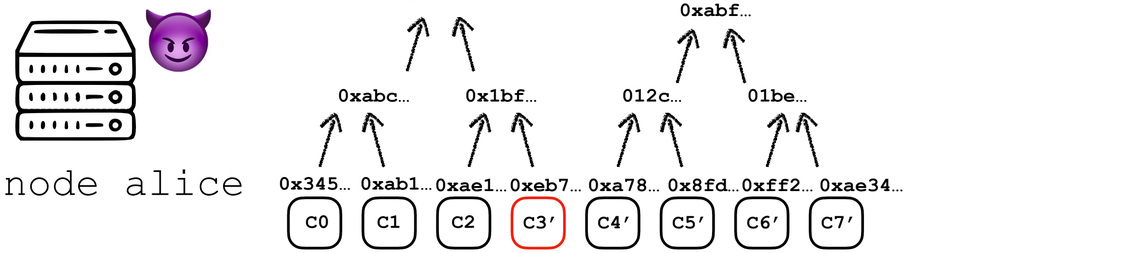
superlight client bootstrapping



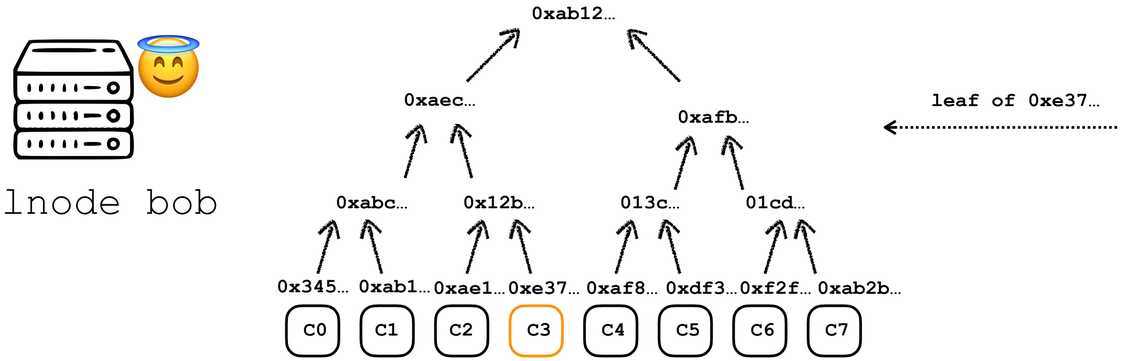




Interactive bisection game

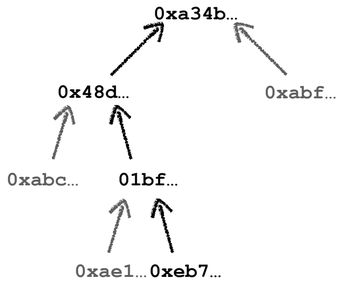


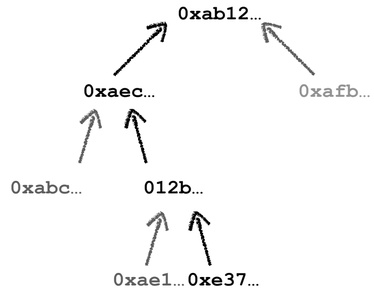
fullnode alice



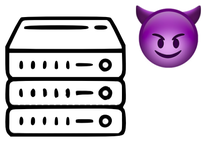
fullnode bob

superlight client bootstrapping

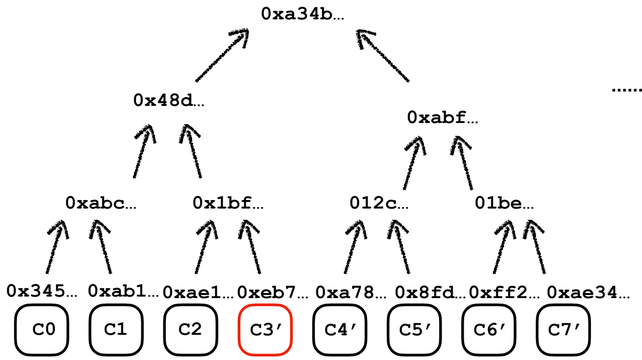




Interactive bisection game

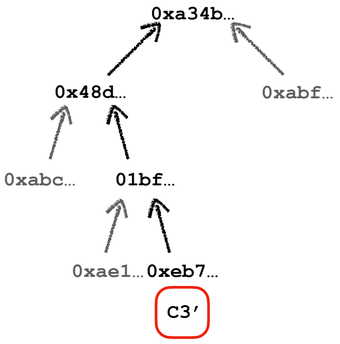


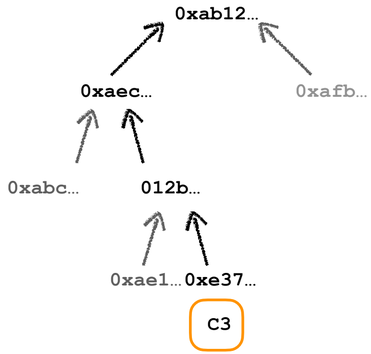
fullnode alice

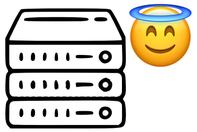


C3′

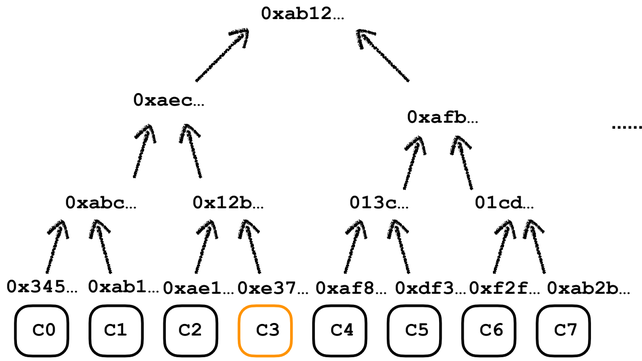
superlight client bootstrapping





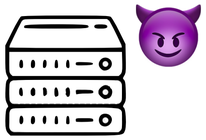


fullnode bob

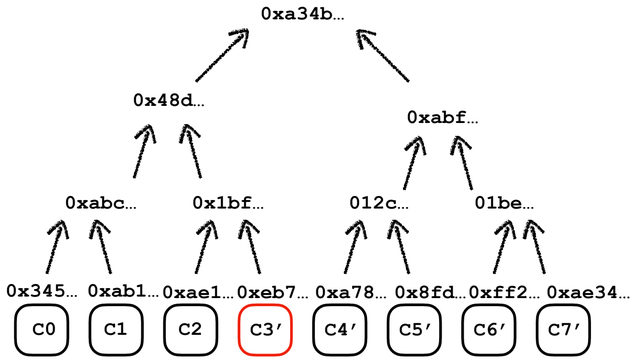


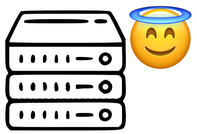
C3

Interactive bisection game

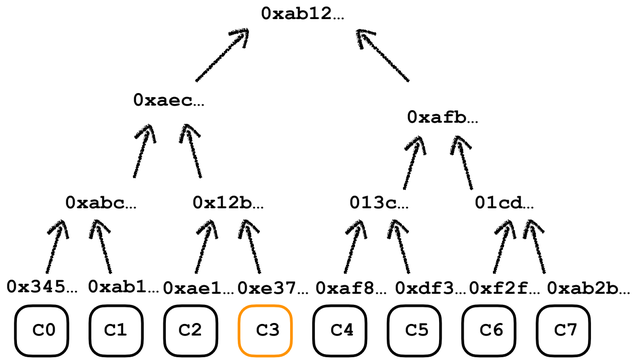


fullnode alice

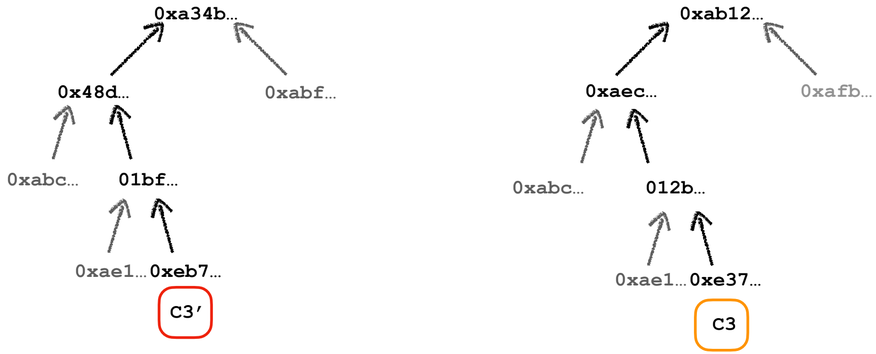




fullnode bob

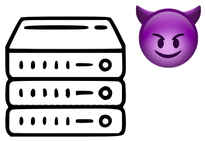


superlight client bootstrapping

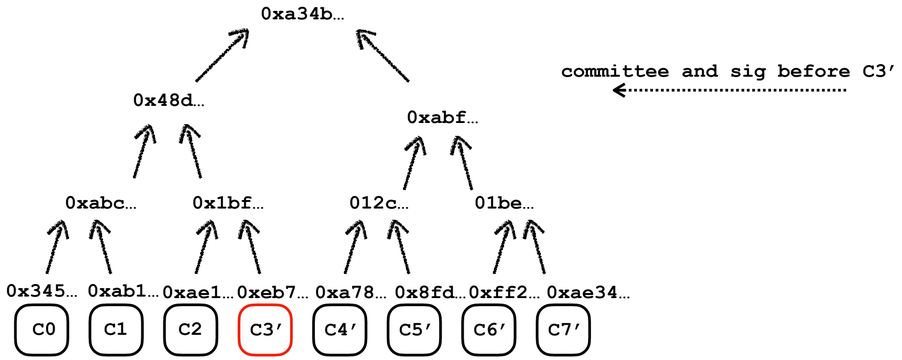


first point of disagreement!!

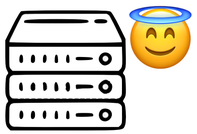
Interactive bisection game



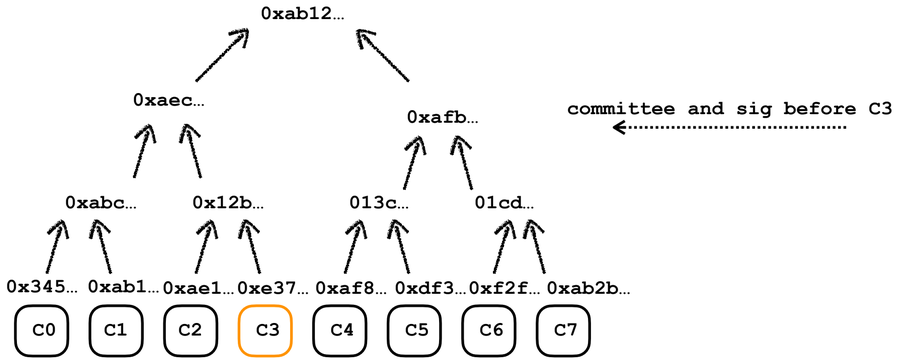
fullnode alice



committee and sig before C3′

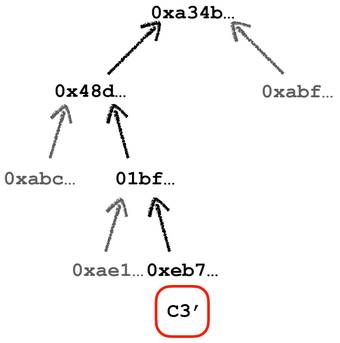


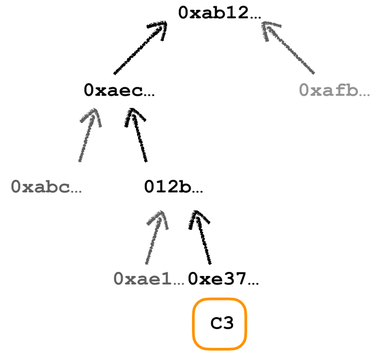
fullnode bob



committee and sig before C3

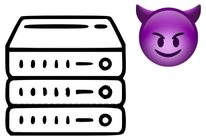
superlight client bootstrapping



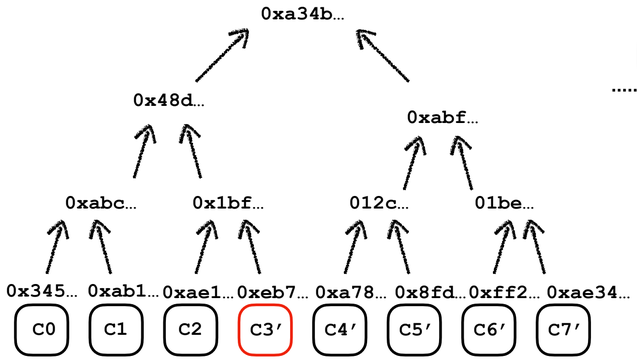


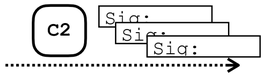
Interactive bisection game

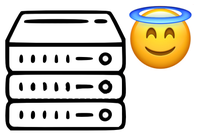
Check sync committee signatures



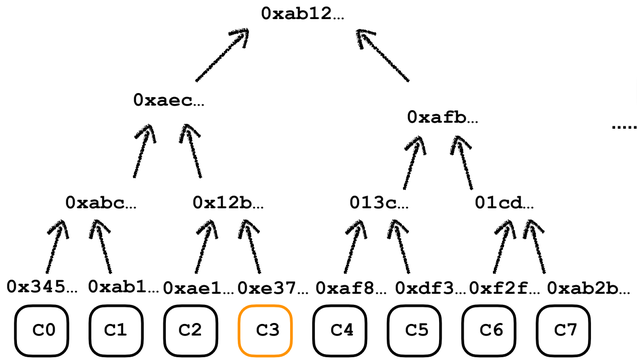
fullnode alice

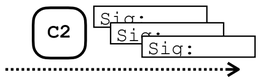




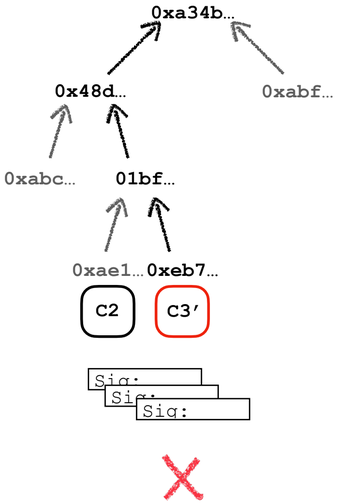


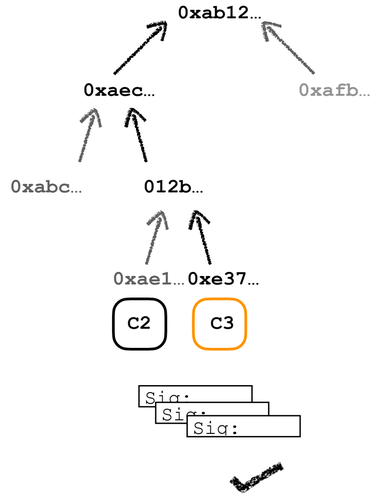
fullnode bob





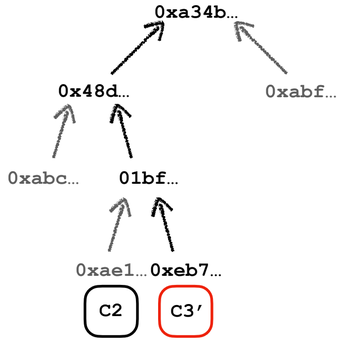
superlight client bootstrapping





Time & space complexity of Superlight Client sync

$$O(log|n|)$$ 🥳



$log|n|$

$|n|$ is number of blocks

How to extend it to multiple provers?

Honest prover will always win!

Implementation: https://github.com/shresthagrawal/eth-pos-superlight-client

Benchmark setup

- 14 provers out of which 13 malicious provers and 1 honest prover
- prover servers deployed to heroku free tier dynos with shared CPU and 512mb RAM
- clients running on MacBook Pro 6-Core Intel Core i9 Processor 32 GB RAM with 40 Mbps internet speed
- for each setup 10 trials were made
- every malicious prover would override the sync committee of the honest chain starting from some period chosen randomly on boot

Drum rolls…

Benchmark results

Beacon Chain data with 155 sync periods and 8 provers

Implementation	Time to Sync	Data downloaded	Interactions
Light Client	110.72 ± 7.00s	16.83 ± 0.21MB	159.60 ± 1.96
Superlight Client	62.63 ± 0.85s	2.29 ± 0.00MB	163.00 ± 0.00

Benchmark results

Dummy Chain data with 1024 sync periods (~2.8 years) and 8 provers

Implementation	Time to Sync	Data downloaded	Interactions
Light Client	803.42 ± 3.69s	106.58 ± 0.22MB	1027.60 ± 2.11
Superlight Client	92.95 ± 1.97s	2.27 ± 0.00MB	205.00 ± 0.00

Didn’t cover in the presentation

- Details about Ethereum PoS
- Using Merkel Mountain Ranges to have periods which are not exact power of tree dimension
- Multiparty tournament in details
- Discuss the original PoPoS construction and how this construction is different
- Details about official light client specification by Ethereum
- Assumption about security and liveness of sync committee
- Implementation details, how it is very close to production implementation
- Benchmarks with increasing chain sizes and increasing provers

Next steps…

- Ethereum Academic Grant
- Work with Stanford team to publish PoPoS paper
- Build a secure wallet for Ethereum PoS

Special thanks to Dr. Dionysis Zindros, and Prof. Sören Petrat

References

[1] Dionysis Zindros. "Proofs of Proof of Stake in sub-linear complexity”

[2] Harry Kalodner et al. “Arbitrum: Scalable, private smart contracts

[3] Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka. “Proofs of Proofs of Work with Sublinear Complexity”

[4] Ertem Nusret Tas et al. “Light Clients for Lazy Blockchains”

[5] Vitalik Buterin. Ethereum PoS Concensus github

[6] Vitalik Buterin et al. “Combining GHOST and Casper”.

Remark: This is not the complete list of references used for the thesis. This is just the main references. For the complete list check the thesis paper