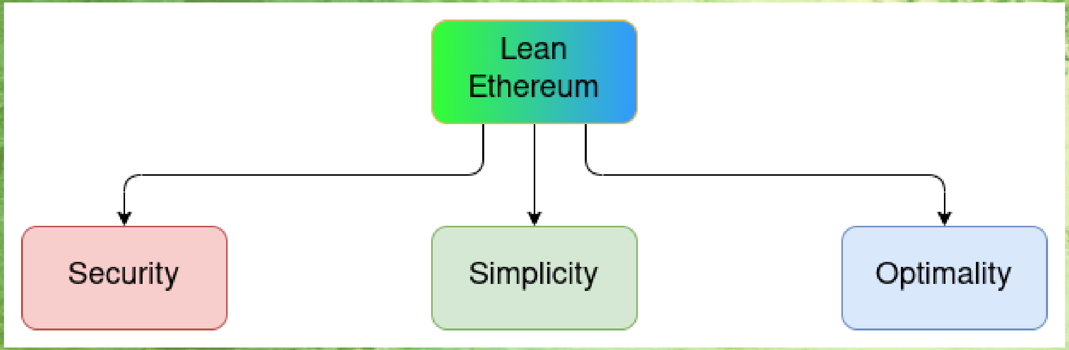
Lean Ethereum

Lean Ethereum: Ethereum for the long term



Lean Ethereum

Security

Simplicity

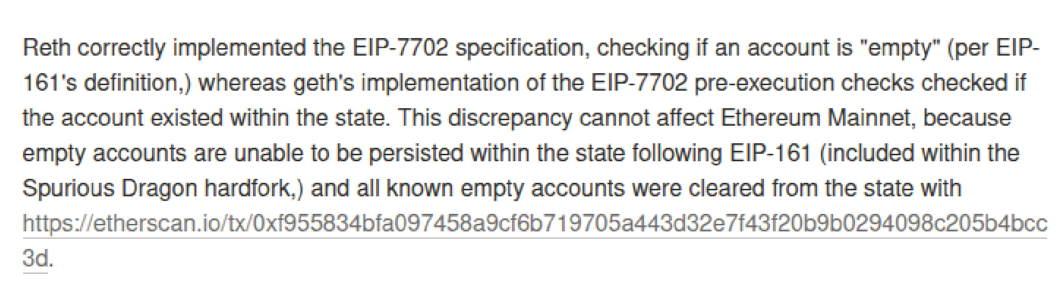
Optimality

Why security?

- Ethereum L1 is the World Ledger
- Fast chains that break are a dime a dozen. Zero-downtime chains where your assets and records are safe are the rare thing
- The ability of everything else (L2, apps...) to function depends on L1 being safe

Why simplicity?

- Maximizes number of people who understand the protocol and feel empowered to contribute
- Forcing function toward ossification
- Feeds into security



Reth correctly implemented the EIP-7702 specification, checking if an account is "empty" (per EIP-161's definition,) whereas geth's implementation of the EIP-7702 pre-execution checks checked if the account existed within the state. This discrepancy cannot affect Ethereum Mainnet, because empty accounts are unable to be persisted within the state following EIP-161 (included within the Spurious Dragon hardfork,) and all known empty accounts were cleared from the state with https://etherscan.io/tx/0xf955834bfa097458a9cf6b719705a443d32e7f43f20b9b0294098c205b4bcc3d.

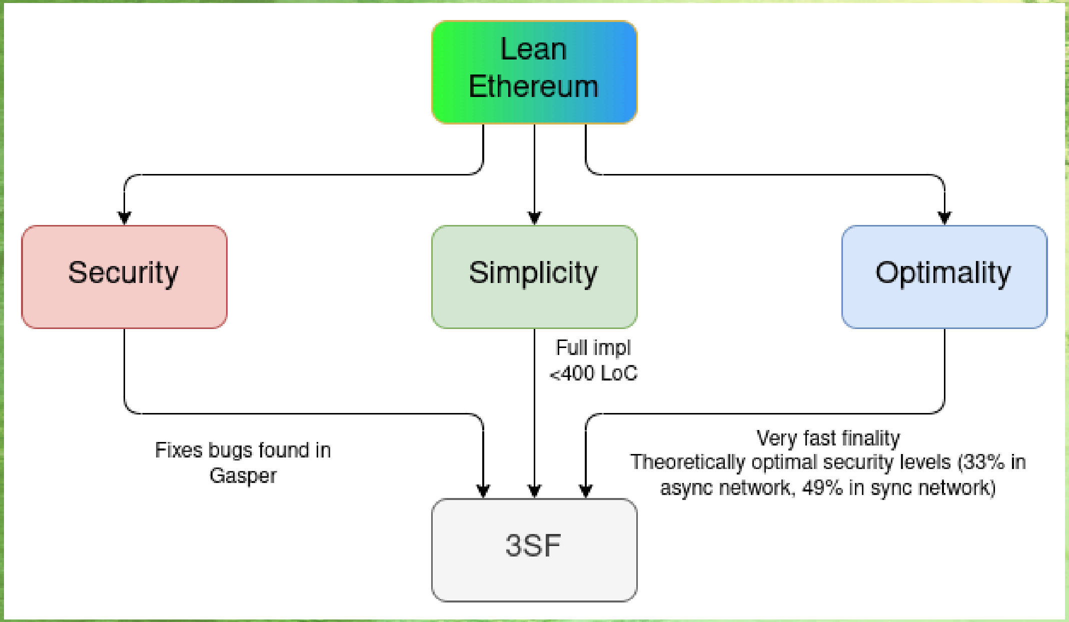
Why optimality?

- We want to scale and minimize latency
- Making Ethereum to be as efficient as possible is the only way to maximize scale and minimize latency without tradeoffs
- Feeds into ossification (if it's optimal, no reason to change it again)

Why now?

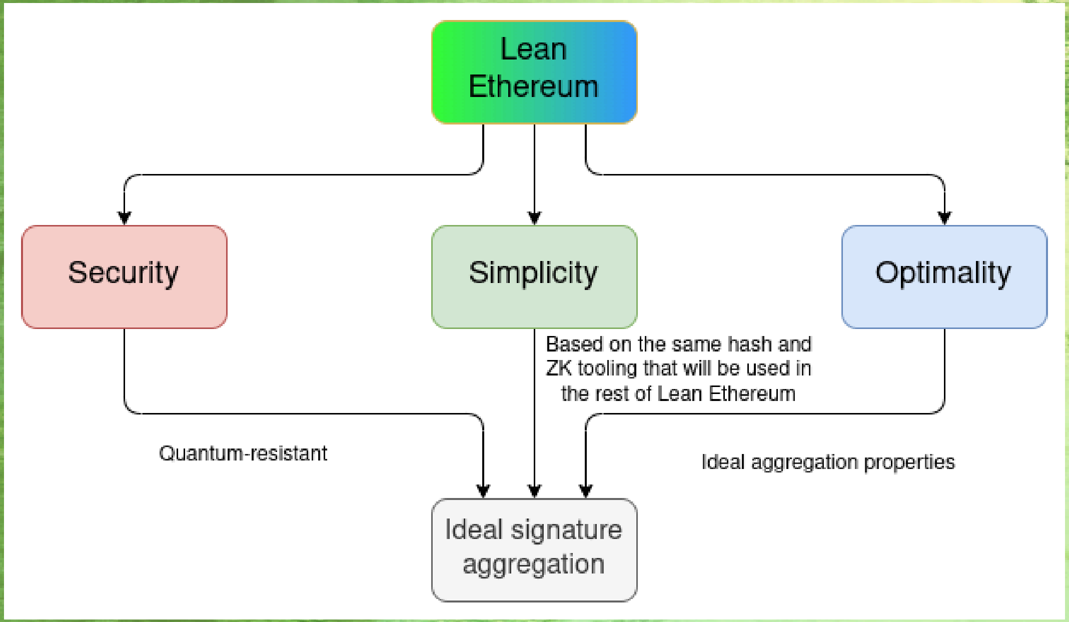
- Consensus research is relatively mature
- ZK is more well-understood
- We have learned a lot from our mistakes
- The risk that we will miss a 10x gain from making the wrong choice is *much* lower than before

Example: 3SF

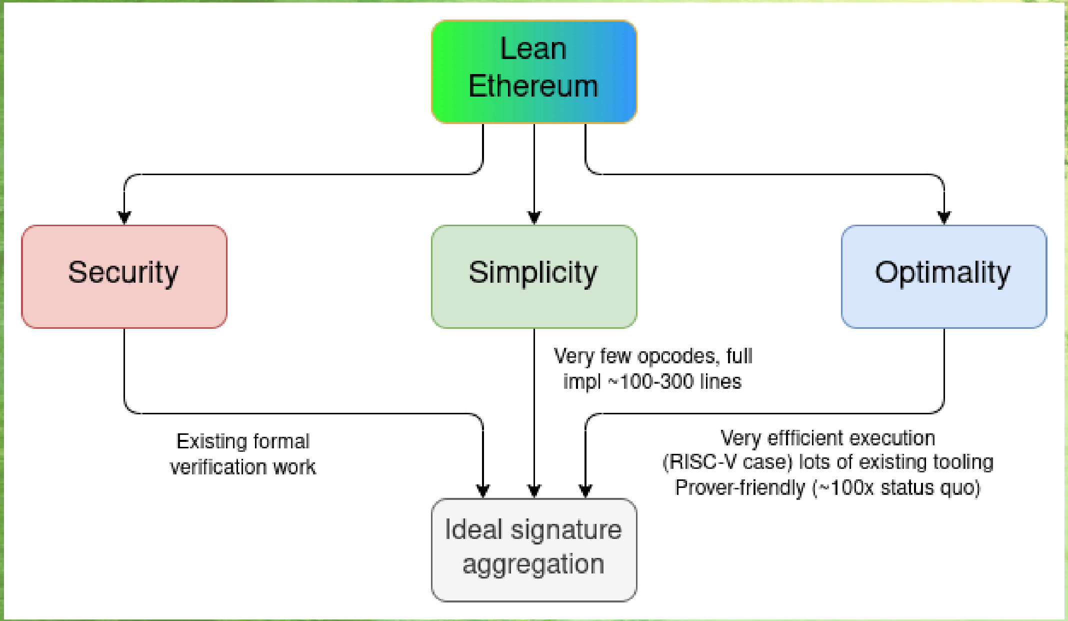


Lean Ethereum
Security
Simplicity
Optimality
Full impl
<400 LoC
Fixes bugs found in Gasper
Very fast finality
Theoretically optimal security levels (33% in async network, 49% in sync network)
3SF

Example: ideal signature aggregation


Lean Ethereum
Security
Simplicity
Optimality
Based on the same hash and ZK tooling that will be used in the rest of Lean Ethereum
Quantum-resistant
Ideal aggregation properties
Ideal signature aggregation

Example: ZK-VM (RISC-V or Cairo-like)



Lean Ethereum

Security

Simplicity

Optimality

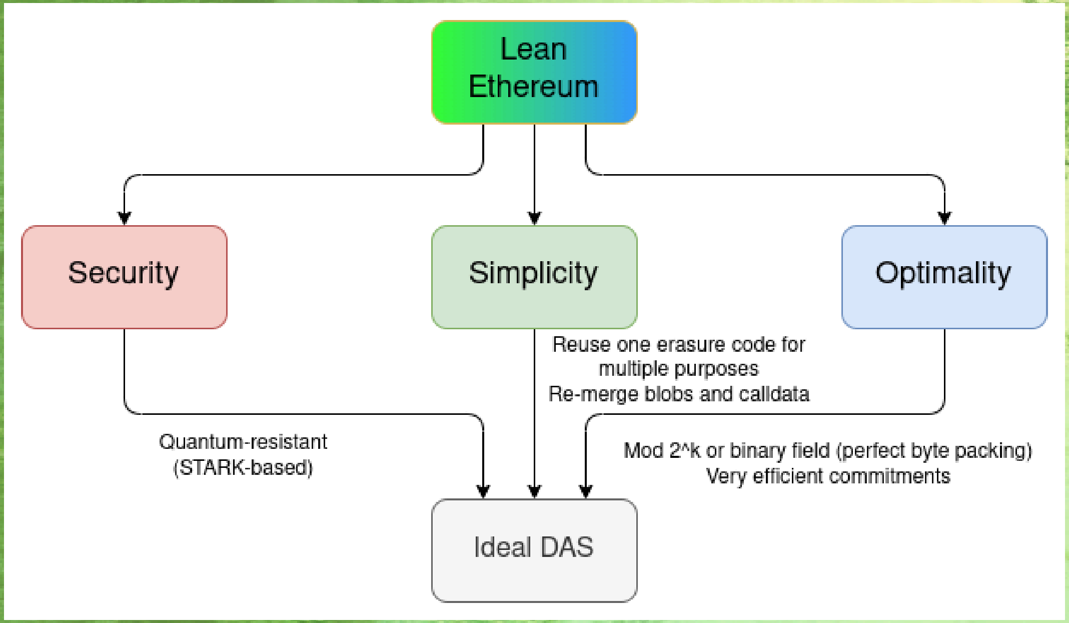
Existing formal verification work

Very few opcodes, full impl ~100-300 lines

Very effficient execution (RISC-V case) lots of existing tooling Prover-friendly (~100x status quo)

Ideal signature aggregation

Example: Data layer



Lean Ethereum

Security

Simplicity

Optimality

Reuse one erasure code for multiple purposes
Re-merge blobs and calldata

Quantum-resistant
(STARK-based)

Mod 2^k or binary field (perfect byte packing)
Very efficient commitments

Ideal DAS