etheorem

machine-checked proofs for Ethereum's consensus spec, in Lean 4

Ivan Anishchuk & Raj Gill · EPF cohort-seven · mentor & etheorem author: Leo Lara

run · test · prove

etheorem is Ethereum's consensus spec, written in **Lean 4** as one artifact you can **run**, **test**, and **prove**.

Executable	**Conformance-green**	**Machine-checked**
real Lean 4, not pseudocode	passes the official vectors (Fulu · Gloas · Heze in review)	growing proofs of the properties clients depend on

Two fellows, split by layer: Raj (@irajgill) on SSZ, Ivan (@IvanAnishchuk) on consensus. With Mouz (@Mouzayan) on the Gloas proofs.

"Lean 4" here is the theorem prover, not the lean chain / Beam.

▓▓ Tested, not proven

Today

- consensus spec, test vectors, clients
- trust rests on *passing tests*
- tests only hit the cases someone generated

Prior proofs

- Dafny / ConsenSys, Phase 0, *archived 2026*
- K + Coq / Runtime Verification, *abstract* model
- **Lean 4, current forks, nothing**

Mainnet stopped finalizing **twice in May 2023** (four epochs, then nine) when a surge of valid old attestations overwhelmed the majority client.

> Never proved anywhere: **serialization total over the codec, merkle-branch completeness, a real shuffle, merkleization over a verified hash.** That is where we work.

▓▓ The plan: two layers, one roadmap

Raj · SSZ layer

the three central theorems, made **total over the codec**: round-trip, injectivity, size bound

Ivan · consensus layer

rising difficulty: arithmetic + structural, merkleization, **the shuffle**, fork choice / FFG

They meet at **merkleization**: the consensus proofs lean on SSZ's guarantees. Two sides of one project.

- **Conformance underneath.** etheorem stays green as the spec moves, and the same suite is the **oracle** every proof is checked against.
- A small experimental cross-check of a real Rust client (**moonglass**): differential run clean so far, **zero divergences**, plus a prototype extraction proof.

The SSZ layer: closing the arms

```
theorem decode_encode : ∀ {s : SSZType}, SSZType.BasicSupported s →
    ∀ (x : s.interp),
      SSZType.deserialize s (SSZType.serialize s x) = .ok (x, ...)
  | _, .uintN8,          x => decode_encode_uintN8 x
  | _, .bitlist,         x => ...   -- merged
  | _, .containerVar fs, x => ...   -- in review
```

Three central theorems, each a **match on SSZ's type constructors**: round-trip · injectivity · size bound.

> **A missing arm is a type the spec serializes and nobody has proved anything about.**

Merged: bitvector / bitlist, wide `uintN` 128 & 256. **In review:** mixed fixed/variable-field containers, the shape `BeaconState` takes, and the hardest arm. **Behind it:** `list` / `vector` over variable-size elements, then the codec is fully covered.

▒▒ The consensus layer: the worked example

Already proved: an honest merkle branch always verifies, down to etheorem's *pure-Lean* SHA-256:

```
-- statement simplified for the slide; full form in the appendix
theorem isValidMerkleBranch_complete
    (h : HonestProof leaf branch index depth root) :
    isValidMerkleBranch leaf branch depth index root = true
```

> Trust footprint: the Lean kernel **alone**, zero new axioms, zero `sorry`. The proof reduces symbolically through `Sha256Spec`, etheorem's pure-Lean SHA-256.

The rest of the roadmap follows this shape, up to the flagship: **the validator shuffle is a real permutation**.

Roadmap, ordered so partial progress still counts

	Raj · SSZ	**Ivan · consensus**
Jul (5–7)	container arms land	Heze, merkle proof, roadmap
Aug (8–12)	variable-element `list` / `vector`, **codec fully covered**	invariants; cached-tree merkleization
Sep (13–16)	theorems on the real schemas, `BeaconState` and `BeaconBlockBody`	**shuffle = real permutation** *(likely slip)*
Oct (17–20)	decoder soundness, one accepted encoding per value; Heze containers	fork-choice invariants; FOCIL on Heze
Nov (21–22)	*joint:* axiom audit · pipeline · report	*(buffer)*

Success is tiered: **minimum / target / stretch**. Whatever lands up the ladder, stands.

▓▓ Already landing

Raj · SSZ

- #10 bitvector / bitlist arms *(merged)*
- #18, #21 wide `uintN` 128 / 256 *(merged)*
- #28/#29 mixed-field containers *(review)*

Ivan · consensus

- #5 Gloas containers, #4 build fix *(merged)*
- #6 Heze fork layer, FOCIL *(in review)*
- merkle-branch proof, roadmap *(pending)*

> **Method, one line:** tool-assisted, machine-drafted candidates, every step checked by the Lean kernel, zero `sorry`, axioms audited. The assistance changes nothing about the guarantee.

First Lean 4 proofs of the consensus spec Ethereum actually runs, already landing.

github.com/etheorem/etheorem · @IvanAnishchuk · @irajgill · review welcome

▒▒ Appendix: the worked example

`isValidMerkleBranch` completeness: feed a perfect combine-tree its honest opening at `index`, and the check always accepts. Specialised to the verified pure `Sha256Spec`, so nothing is assumed about the hash.

```
theorem isValidMerkleBranch_complete_sha256Spec
    {n : Node} {depth : Nat} (hp : IsPerfect n depth) (index : Nat) :
    letI : HasherTag := pureHasherTag
    isValidMerkleBranch (honestLeaf n depth index)
        (honestBranch Sha256Spec n depth index)
        depth index (bytesToRoot (Node.merkleRoot Sha256Spec n))
      = true
```

`#axioms` → `[propext, Classical.choice, Quot.sound]`: the kernel trio, **zero new axioms**, zero `sorry`. The `combine`-returns-32-bytes fact is a discharged hypothesis, not an axiom. (`native_decide` known-answer witnesses live in a separate test lib and carry `Lean.ofReduceBool` on themselves alone.)

Conformance at re-pin: Fulu · Gloas green, Heze in review (v1.7.0-alpha.11). *Re-verify `#print axioms` and the #6 CI run against the branch before presenting.*

▒▒ Appendix: the three central theorems

```
theorem decode_encode : ∀ {s : SSZType}, SSZType.BasicSupported s →
    ∀ (x : s.interp),
      SSZType.deserialize s (SSZType.serialize s x)
        = .ok (x, (SSZType.serialize s x).size)

theorem serialize_injective : ∀ (s : SSZType), SSZType.BasicSupported s →
    ∀ (x y : s.interp),
      SSZType.serialize s x = SSZType.serialize s y → x = y

theorem encode_size_le_max : ∀ {s : SSZType}, SSZType.BasicSupported s →
    ∀ (x : s.interp),
      (SSZType.serialize s x).size ≤ SSZType.maxByteLength s
```

Injectivity falls out of round-trip. The in-review arm, `containerVar`, is hard because the decoder recovers field boundaries from an **offset table** the encoder wrote: the proof walks that table and shows the recovered regions tile exactly the encoder's output. Its preconditions carry the 32-bit offset bound the encoder silently assumes. One arm remains behind it, `list` / `vector` over variable-size elements, the same offset walk element-wise.

Re-verify the axiom audit on all three against the branch before presenting.