

WHAT IT IS

A number a reviewer can re-run.

Your team computes a reactor-physics number in whatever code it already uses. SMRForge seals that result, the inputs that fed it, and the provenance that produced it, into one SHA-256 manifest with one Ed25519 signature. The bundle is self-contained, so anyone downstream can confirm it offline and re-derive it, **without trusting the model, the vendor, or us.**

TWO WAYS TO CHECK IT, NO NETWORK, NO TRUST IN US

```
$ smrforge verify <bundle>
```

Verify the seal

Recompute the fingerprints and the signature on your own machine, with a checker that imports none of our physics. It confirms the number was not altered since it was sealed, and names the exact section if it was.

```
$ smrforge reproduce <bundle>
```

Re-derive the number

Re-run the open engine on the bundle's own recorded inputs. The answer has to land inside a tolerance the bundle committed to in advance. This is what a signed PDF cannot do. It proves the number is real, not just unaltered.

EVIDENCE BUNDLE

1.001742 ± 0.000285

fidelity: screening-to-reference · advisory

case LEU-COMP-THERM-008 · OpenMC

inputs + provenance sha-256 pinned

signature ed25519 cfb88804...

verify integrity + origin

reproduce re-derived within 3σ

offline · no trust required CHECKED ✓

An example sealed result. Illustrative. The seal proves custody, not correctness. The number is unaltered and re-derives from its own inputs.

THE PHYSICS IS FREE

The engine is **open source (MIT)**. SMRForge does not compute your reactor. It makes the number checkable. The evidence is the product.

IT SITS ALONGSIDE

SMRForge adds a checkable record beside your validated codes and the labs. **It replaces neither.**

HONESTLY GRADED

Results are **screening-to-reference and advisory**, labeled for fidelity, benchmarked on open ENDF/B-VIII data. Not a licensing code.

Not “trust us.” Check us.

Verify a real sealed result in your browser
smrforge.io/check.html

WHY A REVIEWER WANTS IT

Five things a regulator wants from a number.

A reviewer has to defend a number they did not produce. The seal delivers each of these.

01 **FASTER**

Checked in milliseconds, not hours.

A hash check and a signature check, run offline. No cluster, no re-running the pipeline. It names any section that changed.

02 **COST EFFECTIVE**

A laptop, not a cluster.

The costly Monte Carlo run happens once. A reviewer re-runs only the cheap deterministic step on verified upstream constants.

03 **SMARTER**

No trust in the producer.

The tolerance is committed before the run. The seal covers fingerprints, not numbers, so a doctored value is caught while the signature stays valid.

04 **JUST AS SAFE**

Nothing about the safety basis changes.

Your validated codes and V&V still produce the number. SMRForge adds a checkable record beside that process, never the authority over it. It subtracts no rigor.

05 **DURABLE**

Still checks decades out.

Verifies offline years later with just the file and the public key. No original author, no surviving cluster, no copy of the tool that made it.

SAID PLAINLY

The seal proves custody, not correctness. Verify runs offline and imports none of our physics. Reproduce needs the open engine, because a browser cannot run a Monte Carlo eigenvalue solve.

MANY AGENTS, ONE CHAIN

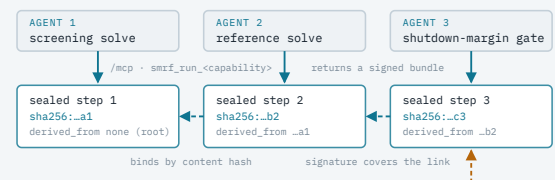
The orchestrator leaves a trail, not just an answer.

\$ smrforge serve • 127.0.0.1:8765

MCP /mcp • REST /v1/run/<capability>

`smrforge serve` runs on localhost and fronts an MCP endpoint, JSON-RPC 2.0 with one `smrf_run_<capability>` tool per capability, plus the same capabilities over REST. An AI agent calls one and gets back a signed evidence bundle as a step in its workflow.

Many agents, one pipeline. Each step seals its result and binds to the one before it by content hash, a `derived_from` link the signature covers. The pipeline becomes a chain a reviewer, human or another agent, walks with `verify --chain`, offline, re-running only the cheap deterministic steps. The orchestrator hands over an auditable trail, not just an answer.



REVIEWER • HUMAN OR AGENT
\$ smrforge verify step-3.json --chain --from ./chain/
walks every link backward, offline, nothing fetched

Hashes illustrative. Each step's seal cites the previous bundle's digest.

LIVE TODAY

`smrforge serve` with MCP and REST on localhost, a signed bundle per call, sealed `derived_from` links, and `verify --chain` walking them offline.

THE DIRECTION, NOT A FINISHED CLAIM

Autonomous orchestration across a whole pipeline end to end, agents re-checking each other. The layer is built for this. It is not shipped as a product today.

AI can produce the number. SMRForge is what makes it **defensible**.