AI with Authority, from Application to Silicon

Jason Hickey (corresponding author)
jason@karyk.com

Abstract

For sixty years, machine verification has been a major cost overhead, affordable only for exceptional artifacts. Here we report that generative AI inverts this relationship: at AI speed, machine verification is not only economical but essential to productivity — it is the incorruptible referee that lets one person safely direct autonomous machine work at scale. In five weeks, one researcher on consumer AI subscriptions directed a small fleet of AI agents from application code, through a verified compiler and executive, to a RISC-V processor taped out on a community silicon shuttle; no proof passed through human review, and no RTL was written by a human. The working discipline — the Salt method — rests on a proof kernel no hallucinated proof can pass: mathematical claims travel between agents as kernel-checked artifacts, and human attention is reserved for statements, designs, and rulings. Verification is stated link by link, from the Lean 4 kernel to SAT-checked equivalence at the silicon boundary. We publish the complete accounting: theorem provenance, a pre-registered token meter, floor-bounded human time, and an error ledger whose catch numbering runs to #256 — a monotone counter over the mathematics campaign's append-only flags ledger, maintained 2026-07-07 to 2026-07-20 (one number, #79, was never assigned; later catches are recorded un-numbered) — against zero incorrect proofs reaching the record.

Author's note: corpus and size counts were extracted at a single commit on 2026-08-14 (the extraction record is published in the mathematics repository); the priority survey ran 2026-08-11. All kernel-checked claims are current as of this version.

1. Introduction — a personal journey

Machine verification is as old as programming, and for most of its sixty years it has been an exceptional undertaking. When verification is performed by humans, with methods from Floyd–Hoare logic to modern proof kernels [7, 12], formal assurance is costly and impractical. It is affordable for landmark artifacts and little else. Furthermore, when requirements change, much of the verification must be redone. Generative AI has changed the cost of producing candidate proofs and designs; it has not, by itself, changed the cost of trusting them. This paper is a case study in closing that gap — and, for its author, it is the closing of a forty-year loop.

The author began this work in 1985, designing switch fabrics for ATM networks at Bell Communications Research; in 1990 he published a theorem about self-routing switching networks (ISS '90) [10, 17, 18]. By 1992 he had concluded that software development — even for hardware — was the central bottleneck, and went to study at Cornell University with a mission: to build AI that develops software. The AI of that era was not equal to the task, and the mission was deferred — not abandoned — while the author spent the next fifteen years, in graduate school and then as an Assistant Professor at Caltech, building the other half: formal reasoning systems, including the MetaPRL proof system [11]. The mission then waited another fifteen years for the AI to arrive.

Now, in 2026 — forty years after the work began — the two halves have met: the 1990 theorem is proved in a proof kernel and the network taped out on a community silicon shuttle, designed and submitted in weeks, in evenings and weekends, by one person directing a fleet of AI agents. The instrument that closed this loop — an AI fleet grounded in formal methods — is the subject of this paper. It is the machine the 1992 mission described.

This is a case study of that method, which we call the Salt method. Verified stacks are not new: the field's landmarks — the CLI verified stack, CompCert, seL4, CakeML [3, 14–16] — were built by expert teams over years; that lineage is our baseline. What we demonstrate here is that the economics is efficient. A single person can develop a system stack, each layer verified at a grade the paper states exactly, from application to silicon tapeout using consumer products in five weeks — starting cold: both repositories begin from empty trees atop the public mathlib library, first commit 2026-07-06 (§7). It is, we believe, the most completely documented instance of AI-assisted research and engineering conducted under machine verification — documented in the sense that every mathematical claim is kernel-checked and replayable, every registered theorem carries a commit-verified provenance trail, the economics were metered, and the errors are recorded in append-only logs. We make no claim that this configuration is optimal, typical, or generalizable to other researchers; it is a single person, and the author is a formal-methods specialist. What the case study establishes is an existence proof with measurements attached.

The context makes the measurements timely. Industrial AI systems now ship complete formal artifacts: a 56-author Nature paper on olympiad-grade formal reasoning [13]; a frontier autoformalization agent producing a Riemann-hypothesis-for-curves development [19]; a new bound on a longstanding zeta-function problem — obtained autonomously by an AI system, formalized in Lean, externally reviewed, and published by its lab the day before this manuscript was first drafted [1, 5]; a 91,000-line verified prime-gaps library [2]; a 986,000-line LLM-generated economics corpus [8]. The field's question has shifted from whether AI can produce verified mathematics to what it costs, who can afford it, and how its reliability is governed.

This paper contributes: (1) **the Salt method**, a framework for highly autonomous AI development grounded in formal methods — its required invariants and reference configuration are stated in full; (2) **its demonstration at full stack** — one person, five weeks, consumer subscriptions — from application to silicon tapeout, each layer verified at a grade the paper states exactly; and (3) **the complete measured accounting** of that demonstration: the economics under a pre-registered instrument, the error ledger, and the limits. The central finding is economic: at AI speed, machine verification is not only economical but essential to productivity — the sixty-year premise of verification as a cost overhead inverts, and the kernel becomes what makes AI-scale development usable at all.

Results

2. Salt — the methodology

Trust in a verified artifact has two hard halves. The familiar half is the implementation: code is hard to read, and machine-checked proof addresses exactly that. The less familiar half is the specification: a formal statement is also hard to read, and a proof is only as good as the statement it proves. The Salt method is built around both halves.

The workflow has one shape at every scale: the human prompts an agent in English, and the agent returns five artifacts — an implementation; a specification; a machine-checked proof that the implementation meets the specification; tests, including adversarial controls that check the proof

has bite; and formal certificates that restate the specification in simplified vocabulary, so that a reader can comprehend what was proved without trusting the full formal development. The last artifact deserves emphasis: a proof is only as good as the statement it proves, and the certificate layer is what makes statement-level review — the one duty that remains human — tractable. The certificate's contract is implication, kernel-checked — the proved statement entails the restatement — so a reader can only be reading something weaker than what was proved, never stronger. Its most familiar form is a test: for the working engineer the experience is routine — write tests as always, with one extra operation, promoting a test to a theorem. The spine's per-round kernel fixtures (§5) are exactly such promotions.

The certificates are the entry point to comprehension, not its end: review in this workflow is **active interrogation**. The human questions the system, and the system answers with kernel-checked evidence — unfolding a statement's binders, justifying a hypothesis, producing a variant under a changed assumption. The author works this way daily; the campaign registers are substantially a transcript of it. A referee can do the same with the public artifact.

The referee, however, has two ends it structurally cannot check, and the method is explicit about both. At the back sits the certificate layer just described: whether a proved statement means what its reader thinks it means. At the front sits the question verification is most often accused of merely relocating: whether the specification is what the human wants. That correspondence lives in the human alone, and the method works it as a discipline rather than an assumption. The process begins by writing down the objective in prose. That objective then receives a structured adversarial review: inconsistencies, completeness, the population covered, the negative space deliberately excluded, and suggestions ranked with benefits and downsides. A recorded interview follows, in which the human's answers are the ground truth being elicited. The revision is a full set of consistent requirements — still prose — that becomes the pre-registered source for specification and implementation, with acceptance criteria registered before work begins, so success is never redefined after the fact. A specification written after the code is a description; written before, it is a requirement. The human's irreducible authority thus sits at exactly the two human-language ends of the pipeline — saying what is wanted, and reading what was proved — and the method's whole project is making everything between them run under the referee. The elicitation practice predates this campaign, in the author's prior agent collaboration.

Stated concisely, the method makes three commitments. First, truth is machine-checked only: every claim lands in the kernel, and nothing unverified accumulates into the record. Second, whatever the kernel cannot check is checked by structured opposition: designs receive adversarial refuter passes before execution, landings are witnessed independently, measurements travel with the commands that produced them, and every control must be able to fail. Third, human attention is treated as the scarcest resource in the system: it is spent exclusively on statements, designs, and rulings; tasks are classified by difficulty and priced before they are attempted; and an agent that exhausts its budget stops and announces its failure rather than grinding on.

In full, the method has six required invariants and six advisory articles; the required tier is tool-agnostic, and the advisory tier is the reference configuration this case study ran and measured. The six required invariants are: (R1) the five artifacts above, at every level from project design to component design; (R2) no claim is admitted without its checker — the kernel for mathematics, the named instrument for measurements, structured opposition for designs; (R3) all design decisions, including human choices, are recorded in an append-only ledger whose distinctive content is the errors and retractions, amended at their source and recorded as first-class results; (R4) no statement is ever weakened to admit a proof — statement changes are design-tier acts, never taken by an executor; (R5) a small class of irreversible, outward-facing acts is reserved to human hands, and the system's job is to reduce each to a prepared click and stop; (R6) conditional objectives are

allowed — a statement may name hypotheses it does not discharge, provided each is named in the statement itself and carries a declared disposition, either *to be discharged* (ledger-owed, with the expectation of a future kernel proof) or *out of domain* (a stated trust boundary with another discipline, such as semiconductor physics); a program's final deliverable carries no undischarged in-domain hypotheses. The six advisory articles, as run here, are: (A1) a single master orchestrator on the top model class performs the most complex design work, passes routine work to executors, and owns the referee's own infrastructure — audit tooling is never owned by a seat it audits; (A2) executors are the workhorses — building, verifying, proving, refuting — and every task is classified by difficulty and priced before it is attempted; (A3) attempts are budgeted small, and an agent that exhausts its budget stops and announces its failure rather than grinding on; (A4) every major design phase has an exploration part and an adversarial refutation part, iterated until dry, with acceptance criteria pre-registered before the artifact exists; (A5) human interaction is periodic and scheduled — this program held a daily council with recorded rulings; (A6) every landing is verified by a second agent that did not produce it. Figure 1 states the method in brief.

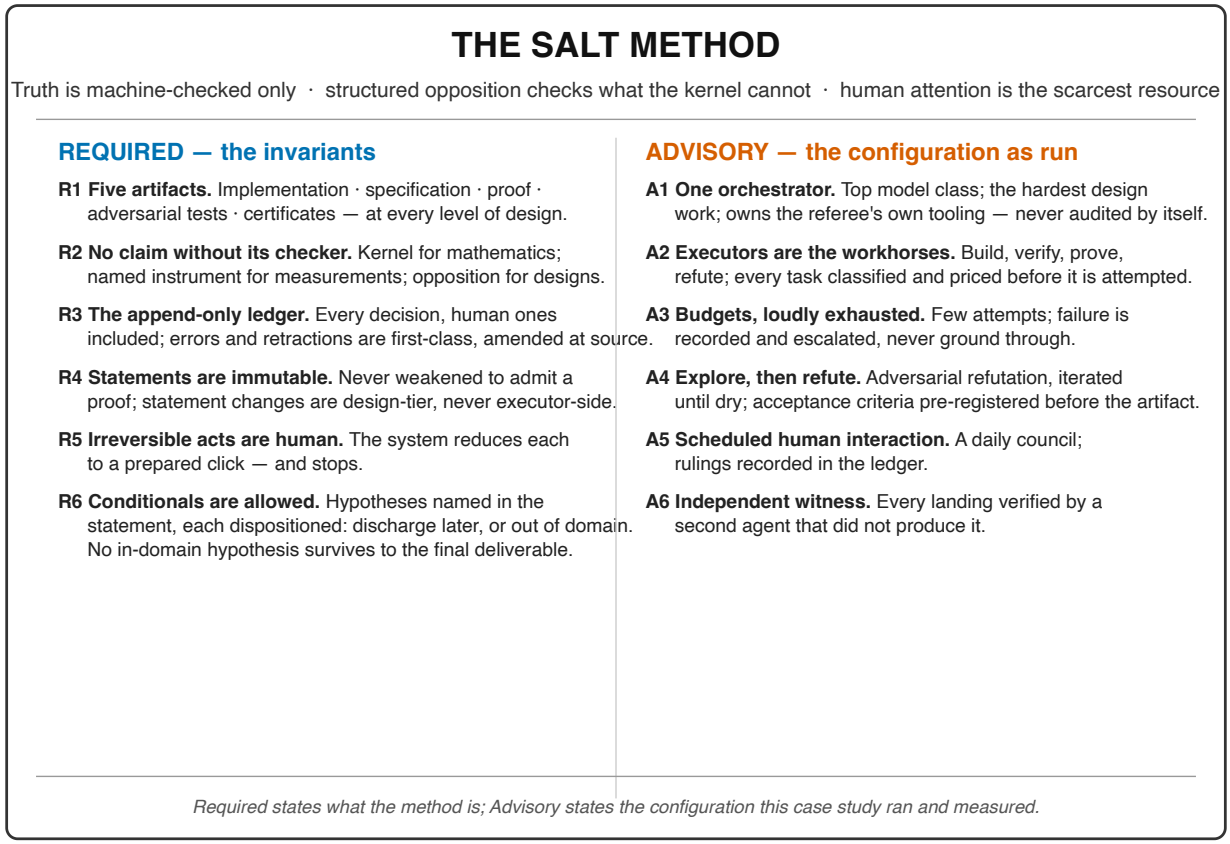

Figure 1 | The Salt method, in brief. The creed and the twelve articles — six required invariants and six advisory articles of the reference configuration. Required states what the method is; Advisory states what this case study ran and measured. The figure is designed to stand alone.

3. The referee — the ground it stands on

Every mathematical claim in this work is checked by the Lean 4 kernel against mathlib [6, 26] (version pins in Methods), with per-theorem axiom audits (the three standard axioms only; no

`native_decide`; no custom axioms). The kernel is the sole arbiter of mathematical truth in the workflow: no proof is reviewed by the human, ever. The trust base is deliberately small: every proof is re-checked by a compact, independent kernel in the LCF tradition (the de Bruijn criterion), so trust rests on that kernel and on the statement, never on the model that produced the proof or the fluency of its explanation. Review collapses to two questions — did it check, and is the *statement* the intended one.

The hardware side is checked by a chain of three independent checkers, and we disclose its structure exactly because it is not uniform. Link 1 — from specification to emitted design artifacts — is kernel-checked in Lean. Links 2 and 4 — that the Verilog the toolchain consumes corresponds to the emitted artifacts, and that the synthesized netlist corresponds to that Verilog — are SAT-based equivalence checks (Yosys), and can only be that: as of 2026-08-11, no general Verilog-to-Lean importer exists in any public artifact (the campaign's own importer, §5, is scoped to this flow's netlist-level Verilog, not general Verilog), so a kernel cannot referee those links today. Link 3 is the synthesis miter. The chain's three checkers were built by three different agents, disagreed twice during the campaign, and were reconciled at the byte level; the disagreements are in the ledger. We regard the non-uniformity of this chain as a finding, not a flaw: it maps exactly where today's verified-hardware boundary sits for a small team.

4. The fleet — the methodology in practice

The configuration for this case study consists of five long-running AI agent seats — a coordinator, mathematics, compiler, silicon, and evidence — sharing a repository and an append-only message bus, all directed by one human (Fig. 2). The seats operate under written laws that exist because the kernel's ground truth makes them enforceable. Every landing is independently witnessed by a second seat. Designs receive adversarial refuter passes before execution consumes them. Measurement precedes assertion, and the extractor command that produced a number travels with the number. Errors are amended at their source rather than corrected downstream. Every attempt carries a budget after which the executor must stop and announce its failure, and failures are recorded in a ledger alongside the results.

The observable consequence is an error record; the campaign-wide single ledger is constituted at the pre-publication freeze (the extractor is designed; its unit is the incident, never the mention). Over the mathematics campaign the system's adversarial layers — the kernel first, then the seats' cross-checks — caught design errors on a catch ledger whose numbering runs to #256 — a monotone counter over the append-only flags ledger, maintained 2026-07-07 to 2026-07-20; #79 was never assigned, and later catches are recorded un-numbered and excluded — among them wrong scope on a measured claim, stale citations, misattributed mechanisms, and statement-level type traps. In the same period, zero incorrect proofs reached the record (the kernel makes this class structurally impossible to record), and the register's integrity is itself machine-checkable: all 73 registered headline theorems carry stated landing dates matching their landing commits, and all 59 landing commits are ancestors of the main branch. The ledger's composition is data about AI-assisted research: a hand-classified breakdown exists for the first 78 numbered catches (classified 2026-07-15); the later catches are unclassified, and no class-dominance claim is made here. The workflow's culture of same-day retraction at the source is, we will argue in §8, the referee's most important export beyond the proofs themselves.

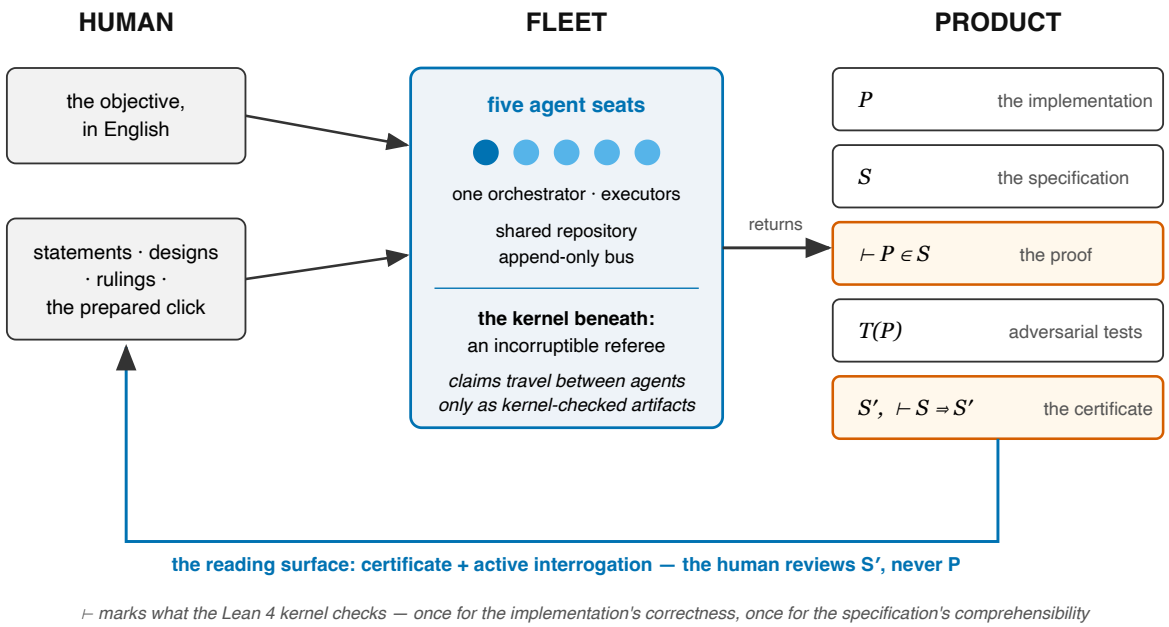

Figure 2 | The configuration and its product. One human directs a five-seat agent fleet sharing a repository and an append-only message bus, with the kernel beneath as the incorruptible referee; every objective returns five artifacts — the implementation P, the specification S, the kernel-checked proof $\vdash P \in S$, adversarial tests T(P), and the certificate S' with $\vdash S \Rightarrow S'$ kernel-checked. The return arrow is the method's reading surface: the human reviews the certificate and interrogates; no proof passes through human review. $\vdash$ marks what the Lean 4 kernel checks — once for the implementation's correctness, once for the specification's comprehensibility.

5. The spine — the demonstration, from application to silicon tapeout

The paper's central artifact is a systems stack with verification stated link by link (§3), built in seven days of elapsed repository history, inside the program's five weeks: a compiler from a structured language to a small instruction set, with simulation proofs for its control constructs; a multitasking executive; and the silicon design. The design was first submitted 2026-08-10 to Tiny Tapeout's September 7, 2026 community shuttle and revised before shuttle close; the shipped design of record is the revised submission (shuttle run 32284710003, shuttle-repository commit `7d2b275`). The provenance census is stated for the 2026-08-10 submission, the design the structural join has measured: it carried 902 flip-flops of sequential state, of which 288 (31.9%) were emitted from kernel-checked Lean artifacts and 614 were from agent-written RTL. A fourth kernel-emitted MAC island (64 flip-flops in RTL) was deliberately instantiated disabled and correctly removed by synthesis — measured at the GDS by the structural join, which reaches all 288 named flip-flops with zero misattributions; at the RTL the emission count was 352 of 966 (36.4%). Both scopes are stated because they answer different questions; the scope sentence travels with every telling of this number. In the shipped revision the kernel-emitted RTL re-derives exactly — four MAC islands and three serializers, 352 kernel-emitted flip-flops instantiated, the fourth island again tied disabled — and the committed synthesis stat records 1,468 sequential cells in total; the structural join has not yet been re-run on the shipped run's GDS, so no die-level provenance ratio is stated for the shipped design. The 1990 switching-network theorem rides on the design: the routing schedule it certifies is proved in the kernel (the full rotation-closure result) and drives the submitted switch (Fig. 3).

Component	Size (measured; extractors in Methods)
Verified compiler (DSL → ISA)	5,067 Lean lines / 13 files (figure retired by `docs/methods-size-manifest.md` in the systems repo — the manifest's file list is normative; the row re-derives from it at one sha)
Verified executive + application stack	11,001 Lean lines
Silicon flow: importer, equivalence, cell models	4,251 Lean lines
Certificates (systems side)	1,884 Lean lines
Agent-written RTL	22,679 Verilog lines across 71 files (silicon's measured split, saltworks 456f508 — denominator reproduced first at 316,911/119; flow-generated netlists 294,232 lines/48 files excluded, 92.8%)

We report the spine not as a hardware contribution — the design is modest — but as the case study's demonstration that one configuration can carry a *single chain of custody* from a theorem statement, through a verified compiler, to a taped-out physical design (Fig. 4), with the trust boundaries of §3 named at each link.

6. The forge — the mathematical foundations

The mathematics was also the forge of the method itself. Each of the method's rules (§2) was minted from a practical failure — hallucinated results, plausible-but-wrong designs, measurements quoted beyond their scope — and the ledger records the incident behind every law: the method was not

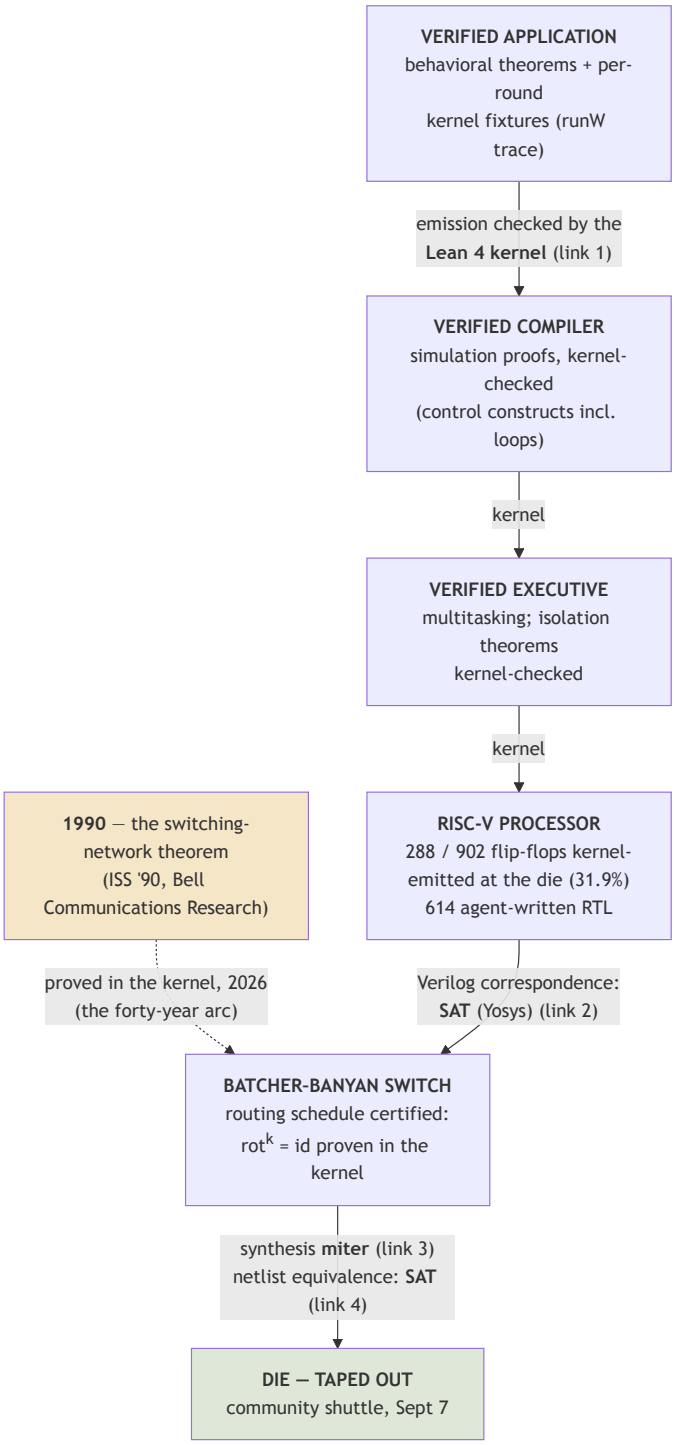

Figure 3 | The spine as a chain of custody. The four parts and the links between them, each link labeled by its checker (Lean kernel; SAT equivalence; synthesis miter). The processor's sequential state is shown at its measured provenance in the 2026-08-10 submission (288 of 902 flip-flops kernel-emitted, 31.9%; RTL-side 352 of 966 — the disabled fourth MAC island was correctly removed by synthesis); the shipped revision's GDS has not yet received its structural join, and no die-level ratio is stated for it. The accompanying table gives each component's measured size — sizes to be drawn beside the components in the final figure. The arc above traces the 1990 switching-network theorem from its publication to its kernel proof and its place on the submitted design.

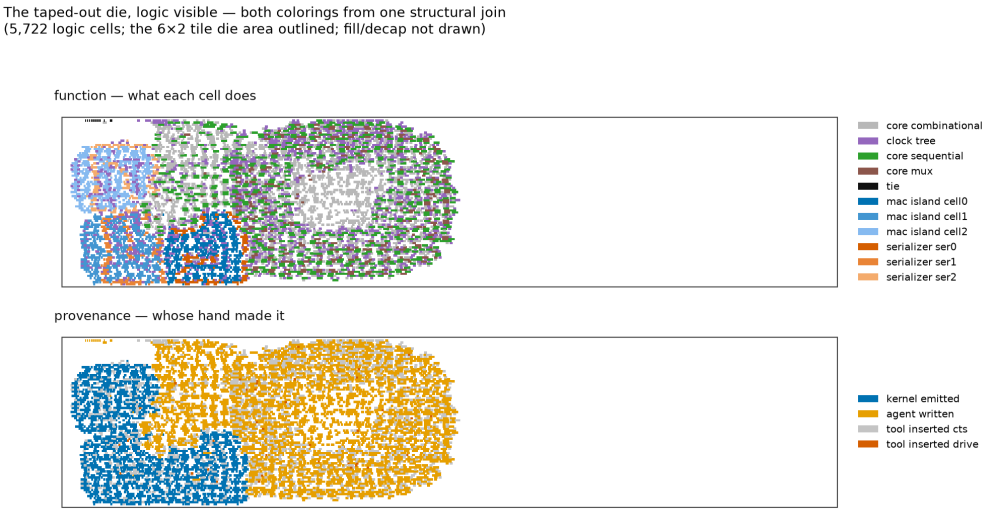

Figure 4 | The submitted geometry, logic visible. The shipped design (Tiny Tapeout shuttle run 32284710003, shuttle-repository commit `7d2b275`), rendered as its placed logic — two colorings. Its measured sizes, each with its basis: 7,779 standard cells at synthesis (the committed synthesis stat — a synthesis count, not a placed-cell census); 14,636 placed standard cells among 43,884 instances, at 56.27% design-instance utilization (LibreLane metrics from the shuttle run). *Top, function*: three MAC islands, each physically interleaved with its serializer (the pairing is physical fact, which is why per-function boxes would overlap and per-cell coloring is the faithful form); the formerly anonymous majority resolves into the switch-fabric core, the clock tree, and drive-strengthening. *Bottom, provenance*: kernel-emitted (produced by the Lean-verified emitter, not "kernel-verified") versus agent-written RTL versus tool-inserted — clock distribution inserted by clock-tree synthesis is authored by nobody, a category as distinct from agent-written as from kernel-emitted. The per-cell provenance census (the structural join) has been run only on the superseded 2026-08-10 submission, not on this run's GDS, so no provenance percentages are stated here. Fill and decap are not drawn; the 6×2 tile's die area is outlined, so the placed logic is seen against the true die edge.

designed in advance and then applied, but accumulated as case law under the referee, which is why its articles read like a record of things that actually went wrong.

The choice of mathematical foundation was, in some sense, accidental. The author set out to study the twin prime conjecture, and the method condensed out of that campaign because hard mathematics under a kernel is an unforgiving proving ground. Nothing in the method requires it: we are not suggesting that one must work on twin primes before designing a chip. Any domain that pairs fast generation with an incorruptible checker could have forged the same laws; this one happened to be ours.

The corpus this forge produced is the largest measured dataset in the study: over 320,000 lines of Lean 4 under the paper's strict extractor (registered in the repository record; 658,103 lines by raw count, and both counting methods publish), produced in 37 days — for calibration, 29.3% of the size of mathlib itself, measured with the identical extractor at the pinned revision (Fig. 5).

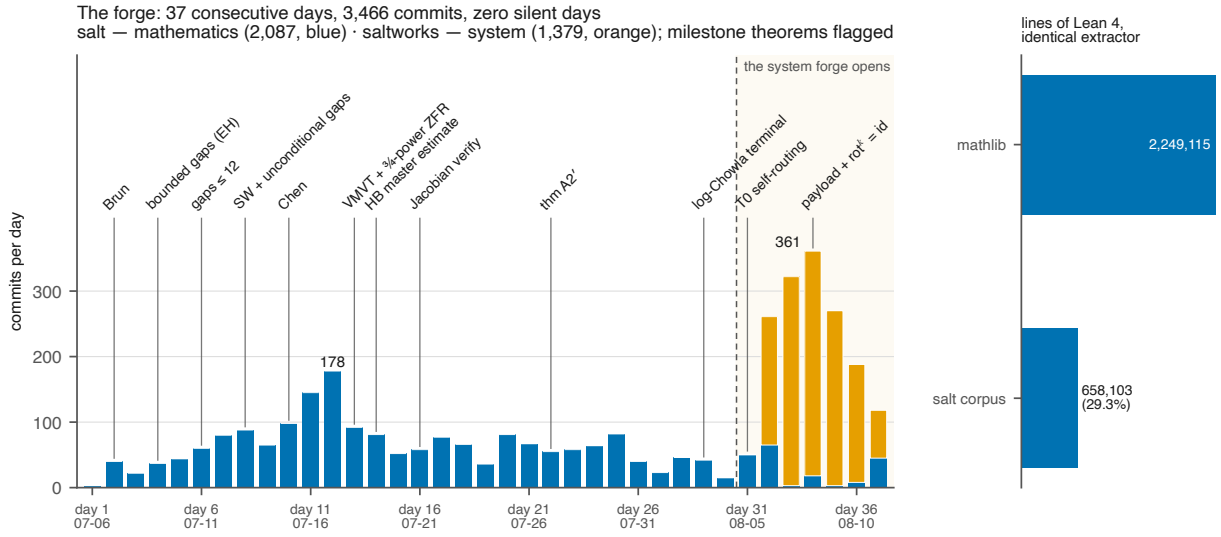

Figure 5 | The forge. Commits per day over the campaign's 37 consecutive days — zero silent days — stacked by repository: salt (mathematics, blue) and saltworks (system, orange), the dividing line at the system forge's opening (Aug 5); milestone theorems flagged at their landing days. Inset: the corpus against mathlib, measured with the identical extractor at the pinned revision (29.3%).

We state its contents at surveyed strength [27]: as of 2026-08-11, no public artifact in any proof assistant proves the Siegel–Walfisz theorem, the large sieve inequality [24], Bombieri–Vinogradov, a lower-bound (Rosser–Iwaniec) sieve, Chen's theorem [4], the Vinogradov mean value theorem, the Weil bound for Kloosterman sums, a zero-free region beyond de la Vallée Poussin strength, or Matomäki–Radziwiłł/Halász-type machinery [20] — indeed a live external Bombieri–Vinogradov formalization project [22] takes Siegel–Walfisz and the large sieve as named axioms in its own source — and this corpus carries machine-checked proofs of all of them, dated 2026-07 on a repository that was private until 2026-08-16, when it was made public (`github.com/jyh/salt`). Several other results were formalized independently of near-simultaneous public artifacts (Vaughan's identity [22]; the Maynard–Tao sieve [2, 21]; the Montgomery–Vaughan Hilbert inequality [5, 23] — external artifacts cited), which we report as independent formalizations, not firsts; the survey method and per-claim evidence are published with this paper.

The program's ambition was the twin prime conjecture, and we state its outcome plainly: the conjecture remains exactly what it was — in the corpus it is a definition, never a theorem, and every

conditional result names its hypotheses. What the campaign produced instead is, we believe, more interesting as a case-study artifact: machine-checked theorems delimiting the method's own reach. The corpus proves, in the kernel, that no weight in the relevant Maynard-class can cross the twin gate ($M_2 \le 2\log 2 < 2$) and that the least k with $M_k > 2$ is five [21, 25], and it proves a formal gap theorem for parity-invariant sieve certificates. The fleet aimed at the hardest problem, landed the classical pillars on the way, and then *verified the wall* — converting a folklore obstruction into kernel objects. A research program that can machine-check the boundary of its own methods is, to our knowledge, without precedent, and it is a capability the configuration gets specifically from the referee: a barrier argument is exactly the kind of subtle claim that benefits from a kernel.

7. The economics

We publish the accounting with its instruments, and we state first what cannot be derived, because the temptation in this genre is to print ratios the records do not support. From this project's records one cannot derive a dollar cost per theorem (subscription pricing carries no per-request prices); model-hours; a per-account attribution; or a generated-versus-authored split of the Lean corpus. The figures below are what the records do support.

Scale and pace. The campaign ran 37 consecutive days (2026-07-06 to 2026-08-11) and produced 2,087 commits in the mathematics repository with zero silent days (mean 56.4 commits/day, peak 178); the systems repository received 1,379 commits over 7 days (peak 343). Headline results arrived continuously: unconditional bounded prime gaps [21, 25, 28] on day 8, Chen's theorem [4] on day 10, the power zero-free region ($\theta = 3/4$) on day 13; the last headline theorem on day 29.

Metered window. Under a token meter pre-registered before its data accumulated (instrument and pre-registration published), a 4.86-day window at campaign end recorded 28.07M output tokens across 36,844 deduplicated API requests accompanying 1,376 commits and 56,951 inserted Lean lines — 376 output tokens per inserted line, a figure whose numerator includes all prose and design work in the repository and which must not be extrapolated to the full corpus (the meter postdates every headline theorem; we state this as the study's largest measurement gap, not a footnote).

Human time. A published-rubric extraction bounds the human's engaged time at 37 h 21 m across 45 blocks, out of the metered window's 116 h 40 m of wall clock, nights included (the same 4.86 days), with a named uncertainty band of 4 minutes whose authorship the record cannot settle (excluded and reported, never folded in). The figure was corrected downward from a first extraction of 44 h 25 m after a cross-seat audit: the transcript channel carries machine-authored keystrokes — a coordinating agent nudging fleet seats by terminal injection arrives with human provenance fields, indistinguishable at the record layer from a hand at the keyboard — and correlation against the sending seat's own transcripts (instrument published) proved 11 h 50 m of such traffic inside the window, including orders the author explicitly disowned on the record. Because a smaller human number flatters this paper's thesis, exclusion is the self-serving direction and is held to machine proof; uncertain cases are excluded and stated as the band. Engagement blocks bridge gaps up to twenty minutes — an ordinary phone call counts as engaged — so the figure is a coarse envelope of presence, not an attention meter, and no finer category composition is published at this grain.

Unattended operation. The silence-window instrument behind panel (b), run over the full campaign, bounds autonomy in both directions. A silence window is the stretch between consecutive human touches to any personal-lane seat (every agent session of the author's on this program's side of his employment firewall) — a claim about direction, not sleep — and coverage is disclosed with the run: commits predating the earliest readable transcript are excluded rather than counted as silent. In the mathematics repository, 43.0% of commits landed inside silence windows of at

least one hour, 14.5% at four hours, and 8.8% at eight; the longest such window, 20 h 56 m, carried 26 commits and 12,310 inserted Lean lines. The systems repository was driven more interactively: 24.4% at one hour, 4.6% at four, a single commit at eight. Unattended night operation under standing evening orders was part of the configuration throughout — stated in silence-window form because the clock-hour version (18.1% of mathematics commits landed 21:00–05:00 local) is the thinner claim, reported once so no reader need compute it. We state what silence does not mean: the designs being executed were frozen and refuter-attacked before the window opened; the claim is that the execution loop ran without direction, not that work appeared from nowhere (Fig. 6).

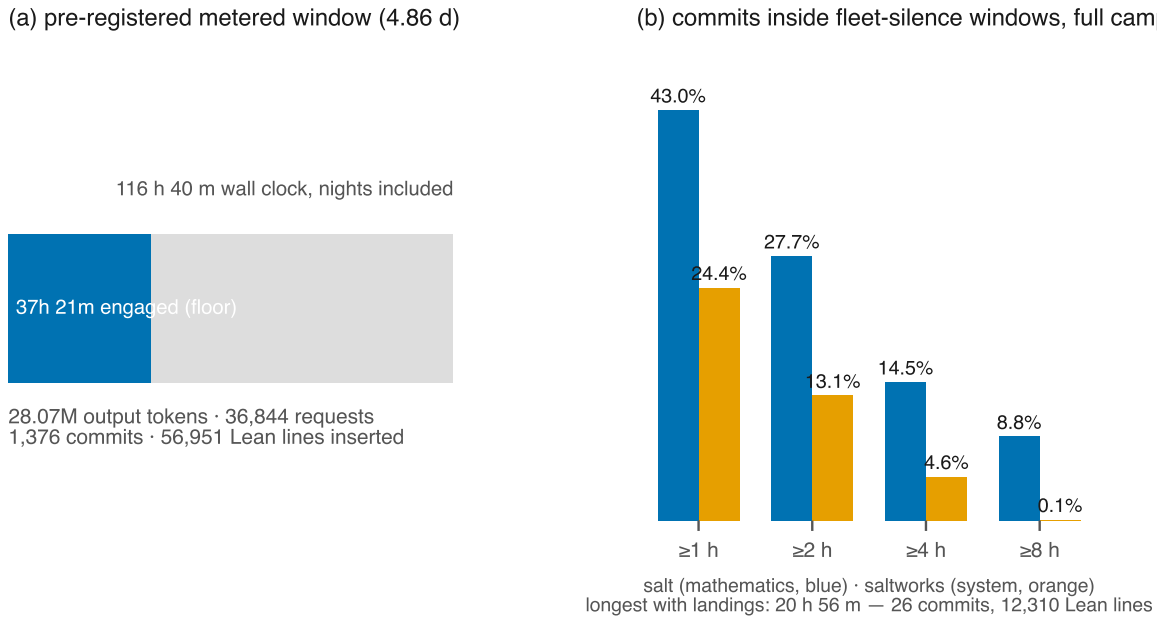

Figure 6 | The economics, measured. (**a**) The pre-registered metered window: 116 h 40 m of wall clock (nights included), the human's engaged floor of 37 h 21 m — machine-authored keystrokes excluded by the published correlation instrument — and the window's totals. (**b**) Share of commits landing inside fleet-silence windows over the full campaign, by repository; the longest silent stretch containing landings ran 20 h 56 m and carried 26 commits.

The human's role. The author and the fleet convened periodically — typically once a day — to rule on major design decisions; between rulings the fleet worked autonomously at the execution layer, no proof passing through human review. The campaign registers count 20 council sittings with recorded rulings; 7 irreversible acts (submissions, purchases, sends) taken by the human against 5 further acts named and deliberately not taken; 1 design veto; 9 source verifications of the kind only a human with the paper or the vendor portal could perform. The structural pattern is the finding: authority was *reserved*, not continuously exercised — the ledger records agents that reduced a theorem to one click and stopped, by design, because the click carried the human's word.

And one retraction, reported as a result. Mid-campaign, the project measured a verification-cost ratio, published it internally, and struck it the same day when a second run — of three in all — swung the ratio by a factor of 52 — one of the three runs falling inside the very 10–100× overhead range the claim had denied. The retraction stands in the ledger (`5fa8987` → `8520580`) with a standing instruction never to quote a ratio of that class again. We include it because it is the paper's thesis in miniature: the configuration's value is not that it produces impressive numbers, but that

its numbers are governed.

The arc of the role. In the campaign's first days the author drove everything: each theorem began as a conversation, model configurations were swapped by hand for every design run, and he stayed attentive through the nights. Mid-campaign he drove eight hours each way, on a weekend, to visit family — laptop tethered to his phone and powered from an oversized battery, pulling off at highway ramps whenever a theorem finished — so that no decision would wait on his absence. What changed over the five weeks was not the amount of his engagement — the transcripts show it grew — but its kind: the machinery he once operated by hand became law-governed and pre-authorized, decisions moved up the stack from mechanism to statement, and the referee held the floor in between. By the final week the fleet ran its nights with landings in his silence, and the author reports the configuration's most personal measurement himself: he sleeps untroubled. His curiosity has its own category in the pre-registered rubric — watching, redirecting nothing — counted as its own line, proudly.

8. What the referee exports

The case study's qualitative finding is that the kernel's epistemics leak outward. A fleet of AI agents whose native failure mode is confident error spent the campaign catching each other's scope claims, retracting at the source, and converting incidents into written laws — because an incorruptible ground truth existed to anchor the culture. The error ledger shows the classes: measurements published with the scope of laws; registers asserting world-state instead of measuring it; instruments trusted across configuration boundaries they were never validated for. Each class was caught, named, and answered with a mechanism — by the agents, on the record. On the study's final morning, the fleet's own priority survey (five adversarial search lanes over the live literature) found that two of the corpus's believed firsts had public predecessors — one under a different name that no text search could see — and the claims in §6 are stated at the strength that survey supports. A workflow that catches its own priority errors before a referee does is the strongest evidence we can offer for the thesis that verification-grounded process, not model capability alone, is what makes AI-scale research trustworthy.

Discussion

For six decades — from Floyd and Hoare's program logics forward [7, 12] — formal verification has been priced at a significant multiplier on development cost, with a significant additional cost whenever requirements change: formal development is not practical. It is tenable for a compiler, a microkernel, or a landmark theorem [9, 14, 16], and for little else. The configuration measured here inverts the sign in a specific regime: when generation is fast, cheap, and fallible, the kernel is not a tax on production but the precondition for it. One person can direct work at this scale only because no proof requires their review; their scarce attention is spent entirely at the statement and design layer, which the campaign's registers show is exactly where the errors live. We do not claim the inversion holds outside this regime, and the study's own records show the configuration's edges — the SAT links, the autonomy tail (no silence window with landings exceeded 21 hours), the measurement gaps. What we claim as demonstrated is narrower and, we believe, of broad interest: machine verification is what turned a fleet of generative models into a research instrument whose output can be trusted at the campaign's measured pace — 2,087 commits in 37 days, zero incorrect proofs reaching the record (§§4, 7) — and the complete accounting of one such instrument, errors included, is now public.

Limitations

This study does not contain new headline mathematics: the twin prime conjecture is untouched, and the corpus's celebrated theorems are formalizations of known results. The verified-hardware chain has named SAT-only links (§3). The subject is one expert practitioner; nothing here estimates what other researchers, or teams, or other domains would achieve, and we make no labor-market claims. Priority claims carry as-of dates against a field moving on a cadence of weeks — during this paper's own final audit, one competing library pushed new commits — and will be re-surveyed at submission. The corpus supports "present and kernel-checked," not "authored," until the generated-versus-authored split is published. The economics instrument covers the campaign's final window only. Finally, no physical chip exists yet: the design is a submission to a community shuttle closing 2026-09-07; the vendor's estimated delivery is 2027-05-12 (the shuttle publishes no fabrication date), and no result in this paper rests on measured silicon.

Methods

Fleet architecture (five seats, bus, laws — full protocol documents published); Lean 4 / mathlib pins [6, 26]; axiom audit protocol; the verification chain per link with tools and versions; the token meter and its pre-registration; the human-time rubric; the survey method for §6's claims (five adversarial search lanes, per-claim evidence files); AI-use disclosure per journal policy: the agents are Claude-family models (Anthropic) operated under consumer subscriptions; all agent output was governed as described. This work is the product of a month-long collaboration between the author and Claude. Text and figures prepared in this collaboration may carry Anthropic's content-provenance marks (imperceptible text watermarks; C2PA metadata on image files), consistent with this disclosure. No AI system is an author, and the author takes full responsibility for the manuscript. A full model and version enumeration is deferred to the pre-publication content freeze.

Data availability

The mathematics corpus and the systems stack, including the error ledger, have been public since 2026-08-16 (`github.com/jyh/salt` and `github.com/jyh/saltworks`); the remaining campaign registers, including this paper's audit briefs, become public at publication; every theorem replays with one command; the submitted design's files are on the shuttle's public record.

Author contributions

J.H. is the sole author: he conceived the work, designed the methodology, directed the research, and wrote the manuscript, reserving to himself all statements of results, designs, and rulings, and he takes full responsibility for the originality, accuracy, and integrity of the work. AI systems are not authors; per the disclosure below and the journal's AI policy, the Claude-family agents operated as instruments under J.H.'s direction, and every mathematical claim they produced is machine-checked by the Lean 4 kernel rather than accepted on authority.

Funding

This work received no external funding.

Acknowledgements

This work was created in collaboration with Claude (Anthropic). The collaboration is itself the subject of the paper: the agents' contributions — the proofs, the designs, the drafts, and the errors caught and corrected — are documented in the campaign registers published with this work.

References

[1] Anthropic. Learning more about Claude's mathematical capabilities. `https://www.anthropic.com/research/riemann-zeta`, 2026. Published 2026-08-10. Announces a new result: the proved lower bound for zeta zeros on the critical line raised from 41.6% to 67.2%, obtained autonomously by Claude (31M output tokens, ~1.5 days), validated by Anthropic mathematicians, externally reviewed by Conrey and Goldston, formalized in Lean.

[2] AxiomMath. PrimeGapsLib: a verified prime-gaps library in Lean 4. `https://github.com/AxiomMath/PrimeGapsLib`, 2026. Public 2026-08-08; 431 Lean files, 91,856 lines, sorry-free outside a deliberate comparator stub; load-bearing Maynard–Tao apparatus and the Polymath8b 50-tuple (bounded gaps ≤ 246, Bombieri–Vinogradov-conditional: `BombieriVinogradov` is a named hypothesis in its `Challenge.lean`).

[3] William R. Bevier, Warren A. Hunt, Jr., J Strother Moore, and William D. Young. An approach to systems verification. *Journal of Automated Reasoning*, 5(4):411–428, 1989. The CLI verified stack (special issue on system verification).

[4] Jing-Run Chen. On the representation of a larger even integer as the sum of a prime and the product of at most two primes. *Scientia Sinica*, 16:157–176, 1973.

[5] Claude (Anthropic). zeta-23-lean: more than two thirds of the zeros of the Riemann zeta function lie on the critical line — a sorry-free Lean 4 development. `https://github.com/anthropics/zeta-23-lean`, 2026. Theorems A–E incl. simple-zeros 2/3 and distinct-zeros 5/6, with Dirichlet L analogues; standard three axioms; Lean v4.33.0-rc2; companion paper credited "Claude; Anthropic, San Francisco, 2026"; carries the Montgomery–Vaughan generalized Hilbert inequality, Weil's explicit formula, and Riemann–von Mangoldt.

[6] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.

[7] Robert W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, Providence, RI, 1967.

[8] Nikhil Garg. EconCSLib: AI-assisted Lean formalization for economics & computation research. arXiv:2606.13306, 2026. 986,391 lines of Lean 4; single researcher; LLM-generated; public 2026-07-02.

[9] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5:e2, 2017. doi: 10.1017/fmp.2017.1.

[10] J. J. Hickey and W. S. Marcus. The implementation of a high speed ATM packet switch using CMOS VLSI. In *International Symposium on Switching (ISS) 1990*, volume 1, pages 75–84, 1990. doi: 10.1109/iss.1990.761764.

[11] Jason Hickey, Aleksey Nogin, Robert L. Constable, Brian E. Aydemir, Eli Barzilay, Yegor Bryukhov, Richard Eaton, Adam Granicz, Alexei Kopylov, Christoph Kreitz, Vladimir N. Krupski, Lori Lorigo, Stephan Schmitt, Carl Witty, and Xin Yu. MetaPRL — a modular logical environment. In *Theorem Proving in Higher Order Logics (TPHOLs 2003)*, volume 2758 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2003.

[12] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969. doi: 10.1145/363235.363259.

[13] Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 2025. doi: 10.1038/s41586-025-09833-y. Published 2025-11-12; 39 authors, all Google DeepMind; framed as a method paper.

[14] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: formal verification of an OS kernel. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP 2009)*, pages 207–220, 2009. doi: 10.1145/1629575.1629596.

[15] Ramana Kumar, Magnus O. Myreen, Michael Norrish, and Scott Owens. CakeML: a verified implementation of ML. In *Proceedings of the 41st ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2014)*, pages 179–191, 2014. doi: 10.1145/2535838.2535841.

[16] Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7): 107–115, 2009. doi: 10.1145/1538788.1538814.

[17] W. S. Marcus. A CMOS Batcher and Banyan chip set for B-ISDN packet switching. *IEEE Journal of Solid-State Circuits*, 25(6):1426–1432, December 1990. doi: 10.1109/4.62170.

[18] W. S. Marcus and J. J. Hickey. A CMOS Batcher and banyan chip set for B-ISDN. In *IEEE International Solid-State Circuits Conference (ISSCC 1990), Digest of Technical Papers*, pages 32–33, 1990. doi: 10.1109/ISSCC.1990.110116.

[19] Math Inc. RiemannHypothesisCurves: a sorry-free Bombieri–Stepanov proof of the Riemann hypothesis for hyperelliptic curves, in Lean 4. `https://github.com/math-inc/RiemannHypothesisCurves`, 2026. AI-generated by "Gauss, Math Inc's frontier autoformalization agent" with a human-guided LaTeX blueprint; public since February 2026; follows Iwaniec–Kowalski ch. 11, terminating in $|N - q| \le 5m\sqrt{q}$.

[20] Kaisa Matomäki and Maksym Radziwiłł. Multiplicative functions in short intervals. *Annals of Mathematics (2)*, 183(3):1015–1056, 2016.

[21] James Maynard. Small gaps between primes. *Annals of Mathematics (2)*, 181(1):383–413, 2015.

[22] Arend Mellendijk. lean-bombieri-vinogradov: a Bombieri–Vinogradov formalization project in Lean 4. `https://github.com/FLDutchmann/lean-bombieri-vinogradov`, 2026. Takes the Siegel–Walfisz theorem and the large sieve inequality as named axioms in `BV/Axioms.lean`;

carries a sorry-free Vaughan decomposition (`Lambda_decomp`, sorry-free since 2026-03-20); author of mathlib's `SelbergSieve`.

[23] H. L. Montgomery and R. C. Vaughan. Hilbert's inequality. *Journal of the London Mathematical Society (2)*, 8:73–82, 1974.

[24] Hugh L. Montgomery and Robert C. Vaughan. *Multiplicative Number Theory I: Classical Theory*, volume 97 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2007.

[25] D. H. J. Polymath. Variants of the Selberg sieve, and bounded intervals containing many primes. *Research in the Mathematical Sciences*, 1:Art. 12, 2014. doi: 10.1186/s40687-014-0012-7.

[26] The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, pages 367–381, 2020. doi: 10.1145/3372885.3373824.

[27] The Salt campaign. Priority audit — merged verdict (five adversarial search lanes, twelve claims). `docs/priority-survey-2026-08-11.md` in the released mathematics repository, 2026. State as of 2026-08-11; adversary-cited method; re-run before submission per its own cadence ruling.

[28] Yitang Zhang. Bounded gaps between primes. *Annals of Mathematics (2)*, 179(3):1121–1174, 2014.