hexagon:2611.01314v1

Adversarial Multi-AI and Independent-Kernel Verification in Frontier Mathematics: An Empirical Study of 31 Numbered Defect Groups Across Four Projects

Robert Kadlec*

Single human author directing a multi-AI campaign across
Antigravity/Gemini, OpenAI Codex, Anthropic Claude, and Lean 4 checkers

November 28, 2026

Abstract

Large-language-model agents now generate thousands of lines of Lean 4 code, high-precision numerical certificates, and research manuscripts at a pace where trust—not generation—is the primary bottleneck. We report an empirical study of a multi-agent mathematical research campaign directed by a single human author (Robert Kadlec) across four projects: formal quantum-circuit lower bounds and synthesis (P1), structural auditing of a million-declaration Riemann Hypothesis archive and post-RH frontier (P2), irrationality exponents of CM abelian periods via separated weights (P3), and arithmetic genus and monodromy of a simultaneous-reality curve (P4). Across Waves 1–3C, a post-T5 genus push, and clean-machine Linux `x86_64` QA, our adversarial pipeline catalogued 31 numbered confirmed defect groups—spanning `def`/`abbrev` reducibility mismatches, wrapped-line axiom-parser omissions, sign and branch-coverage bugs in arithmetic-geometry certificates, self-invalidating hash pins, misattributed literature, and Linux linker/`tmpfs` RSS accounting defects. Cross-tabulating all 31 groups against their primary detecting layers demonstrates sharp complementarity: clean-host Linux gates caught 3 high-severity build, portability, and measurement defects that escaped LLM readers, whereas cross-model AI referees caught 22 groups of mathematical, specification, provenance, and second-order patch regressions outside the kernel's boundary (a share conditional on audit frequency in this single-project case study). We accompany this taxonomy with four open verification tools—highlighted by `KroneckerFpCertV2` and `lean-slice-oss`, whose cross-OS theorem-unfold ablation establishes that disabling theorem unfolding breaks Mathlib checking on `Rat.instEncodable`—seven operational protocols, and a post-v3 release-bundle audit case note motivating an end-to-end packaging gate.

1 Introduction

AI-assisted mathematical workflows now routinely combine computer algebra, arbitrary-precision interval arithmetic, and large-scale formalization in Lean 4 [3] over Mathlib [4]. Experience across recent formalization campaigns shows that two opposite failure modes dominate: an unrefereed single agent frequently overstates its findings or misquotes external work, while even a kernel-checked Lean 4 theorem certifies only the exact formal statement inside the kernel—which can quietly drift away from the theorem advertised in prose [1].

*Masaryk University, Brno, Czech Republic. Email: `kadlecr@mail.muni.cz`. ORCID: 0009-0003-9245-8130. Purely AI-generated research directed and supervised by the author, by whom full scholarly responsibility for the integrity, accuracy, and publication of all reported results is assumed (all mathematical derivations, Lean 4 formalizations, verification scripts, audits, and manuscript prose were executed by AI systems). This paper forms part of a broader research project investigating the respective competencies of humans and large language models.

This paper reports an empirical post-mortem (Version 3, updated through November 28, 2026) of a multi-project research campaign directed by a single human author (Robert Kadlec, Masaryk University, Brno) orchestrating purely AI-executed research under an adversarial triangulation protocol across three distinct model families (Google Gemini/Antigravity, OpenAI Codex, and Anthropic Claude), a five-check LEAN 4 verification pipeline, clean-machine Linux `x86_64` replays, and human editorial oversight. Building on the initial 21-group catalog (Waves 1–3B), we catalog *31 numbered confirmed defect groups* across Projects P1–P4, Tracks T1–T7, the post-T5 Paper 4 genus push, and clean-machine Linux QA of `lean-slice-oss`, map each group to its primary detecting layer under an explicit mechanistic-trigger rule, document four open verification tools, and record post-v3 pre-submission release-bundle audit findings that motivate a dedicated final packaging gate.

Contributions.

1. **Honest multi-checker architecture (Section 2):** an explicit accounting of what a five-check LEAN 4 pipeline actually guarantees, separating distinct codebases from shared kernels and introducing a three-tier epistemic taxonomy (Tiers A, B, C) plus four claim-status tags ([**verified**], [**conditional**], [**simulated**], [**absent**]).
2. **Empirical catalog of 31 numbered defect groups (Section 3):** grouped confirmed defects across Projects P1–P4, Campaign Waves 1–3C (Tracks T1–T7), the post-T5 Paper 4 genus push, and clean-machine Linux `x86_64` QA of `lean-slice-oss`, classified by severity, defect type, and repair status.
3. **Detection-layer matrix (Section 4):** a quantitative cross-tabulation across all 31 groups showing how formal kernels, clean-host Linux gates, cross-model AI referees, blind/external reviews, and human oversight covered complementary error classes, together with the explicit primary-detection-layer attribution rule.
4. **Four open verification tools with Windows and Linux `x86_64` benchmarks (Section 5):** `KroneckerFpCertV2`, `lean_dag_audit`, `stream_export_audit/lean-export-stream`, and `lean-slice-oss` (`SliceFrontendV2_OSS` plus `nanoda_lib` file-backed arena patches and the `Rat.instEncodable` causal theorem-unfold ablation).
5. **Actionable protocols and post-v3 release-bundle audit (Sections 6–7):** seven concrete engineering rules derived directly from the 31 observed failure groups, supplemented by post-v3 pre-submission release-bundle audit findings and explicit conclusions.

2 Adversarial Multi-AI and Multi-Checker Architecture

2.1 Role Separation, Read-Only Boundaries, and Epistemic Tiers

The campaign operated over four frontier projects:

- **P1 (Quantum algorithms, Lean 4 4.35.0):** adaptive `GroverFindAny` lower bounds, unconditional `GroverFindAll` lower bounds (Track T2: fiber Theorem A and Belovs–Yolcu Johnson adversary Theorem B proving $r \geq 1/(16\pi) \approx 0.01989436$, capping speedup over `GRO` at $\leq 3.4973\times$, alongside parametric frozen-spec bridge `T2-m1`), `ArcsinK25` circuit synthesis (366,200 $T \times Q$ vs. 369,800), and fault-tolerant lattice-surgery resource estimates.
- **P2 (Riemann Hypothesis archive audit and post-RH frontier, Lean 4 4.33.1):** dependency-graph auditing, zero-commit compilation (`lean-slice-oss`), sub-polynomial Landau-window obstructions (`SubPolySuperoscSeries.lean`) around a 1.265-million-declaration RH archive, and post-RH non-abelian S_3 Hecke and Rankin–Selberg obstructions (Track T6).

Table 1: Five checker invocations and their actual trust boundaries across campaign receipts.

Invocation	Codebase	Actual guarantee and observed caveat
`lean`	Official kernel (4.33.1 / 4.35.0)	Elaborator plus kernel; primary source of per-target `#print axioms` output (subject to line wrapping on long declaration names, Defect #21).
`leanchecker`	Same official kernel	Independent invocation replaying `.olean` environment; not an independent kernel implementation.
`con-leche`, `con-ron` (`-verified`)	External v4.36.0	Check `lean4export 3.1.0` stream; decline (exit 2) reflexive mutual inductives and exhaust fuel (exit 3) on monolithic > 200 MB exports.
`nanoda_bin`	External Rust v4.36.0 / `0.4.32`	Independent typechecker; enforces 3-axiom hard-error permit list; unpatched 4.33.1 runs emit benign `Unable to print axioms` (repaired in `lean-slice-oss` Patch 2; 4.35.0 prints axioms cleanly).

- **P3 (CM abelian periods and Beta values, Lean 4 4.33.1):** separated-weights $\mu(\omega) = 2$ conditional formalization, full integral-closure multivariate Hensel Lemma 2.9 (`MultivarHenselFullLemma29.lean`), fixed-chart genus-2 Beta period bridges (`Genus2FixedChartBridge.lean`), eigenclass non-annihilation (`EigenclassNonAnnihilation.lean`), and level-$8, 12$ algebraic twist irreducibility certificates.

- **P4 (Simultaneous-reality curve, Lean 4 4.33.1):** arithmetic genus enclosure tightened from $6{,}631 \leq g(C) \leq 16{,}570$ through P4-06 (12,909), Track T5 (9,037, +3,872 units), and the audited post-T5 push to $6{,}631 \leq g(C) \leq 7{,}423$ (302,117/302,909 units certified under Global Inputs (G1)–(G3)), full monodromy S_{269}, and gonality $d \geq 26$ for twin-factor deformations of Salikhov's π-integral [6, 7].

Three operational invariants governed every audit wave: (1) *disjoint authorship and review*—no model family audited code or prose it had written (Gemini-authored artifacts were audited by OpenAI Codex or Anthropic Claude, and vice versa); (2) *strict read-only upstream directories*—frozen releases were hashed before and after every audit; and (3) *explicit epistemic tiering*: **Tier A** denotes machine-checked Lean 4 theorems with zero `sorry`/`native_decide` and standard axioms only; **Tier B** denotes deterministic, clean-machine reproducible Python/FLINT/SymPy computations with pinned digests; and **Tier C** denotes unrefereed human or AI-assisted mathematical prose.

2.2 The Five-Check Lean 4 Pipeline and Its True Independence

Every Tier A module was processed through five sequential checker invocations wrapped by a single-threaded mutex driver (`run_lean_locked.py`, `LEAN_NUM_THREADS=1`, `RAYON_NUM_THREADS=1`) recording source, object, and export SHA-256 digests: (1) `lean` (4.33.1 or 4.35.0, printing `#print axioms` for every target), (2) `leanchecker` (replaying compiled `.olean` terms), and (3–5) export via `lean4export 3.1.0` followed by three external export checkers: `con-leche -verified`, `con-ron -verified`, and `nanoda_bin` [5].

Table 1 states the real independence boundaries read directly from the campaign receipts. Crucially, "five checks" spans at most *four* codebases: `lean` and `leanchecker` share the official C++ Lean kernel, `con-leche` and `con-ron` share an export-checking specification, and on unpatched Lean 4 4.33.1 exports `nanoda_bin` emits zero typechecker errors alongside a benign pretty-printer diagnostic (`Unable to print axioms`, Finding `W2-1`—subsequently fixed in Patch 2 of `lean-slice-oss` with exact tagged `Name` component reporting) while enforcing its strict three-axiom permit list (`propext`, `Classical.choice`, `Quot.sound`) via `unpermitted_axiom_hard_error: true`. Calling this pipeline "five independent kernels" is therefore inaccurate; accurate phrasing is "five checks across four checker codebases."

Table 2: Confirmed defect groups #1–#11 (Projects P1, P2, and P3 early waves). Severity: Crit. = Critical, High, Med. = Medium, Low.

#	Proj.	Defect / Vulnerability	Caught By	Sev.	Status
1	P1	`def` instead of `abbrev` on `weakChosenBatch` caused `ReducibilityHints` mismatch	Upstream Linux `lake comparator`	High	Fixed (`v1.0.1`)
2	P1	Equivalence of `Feasible` with `Grover.lean` spec claimed by "direct inspection" only	Blind cross-model AI PDF review	High	Fixed via `rfl` theorem (`P1-01`)
3	P1	`Track1THC` prose claimed CAR missing upstream; actually present on branch `discoveries`	Codex cross-audit (`P1-06`)	Med.	Scoped in App. B fix
4	P1	Initial FT resource table double-counted `CZ` as $2 \cdot$ CNOT and lacked separate parser receipt	Claude Opus Wave 1 (`C1,C3,C4`)	Med.	Fixed and reproduced (`P1-08`)
5	P2	7 portability bugs in Linux replay scripts (`elan`, `nanoda` pin, 7 missing `.olean` hashes, 18 targets lacking `#print axioms`)	Clean-Linux run by human colleague	High	Fixed (`v2.1.0-pre`, 24/24 match)
6	P2	Thread mix-up placed two P2 versions into release folder and omitted P4 manuscript	Human author + session audit	High	Fixed and verified
7	P2	Unnoticed concurrent literature (Nov 25: `lsernau/grh-lean` 154× compression; `riemann-rev` overlap on Thm. 6.1)	Blind cross-model review + web scan	High	Documented (`P2-03/04`)
8	P2	`UnusedSlot80/33` were not dead namespaces; `HypothesesH.deg` live; `PUnit.{2}` dummy in `CertificateGrounding`	Codex + Claude Sonnet audit	High	Fixed (`v2.0.0`)
9	P3	Misattributed external work [2] as "conditional formalization", whereas its 976 Lean 4 files prove $\mu(\Omega) = 2$ unconditionally	Reviewer + direct audit (`P3-01`)	Crit.	Patched (with fix #10)
10	P3	Patch `p3_01_scope.patch` hunk 2 left dangling pronoun "it", misattributing P3's own engine to external paper	Claude Opus Wave 1 (`A1`)	Med.	Fixed in `p3_rebased`
11	P3	Obstruction (1) heading wrote "$(g \geq 2)$" while range $2 \leq \dim Y \leq g - 1$ is empty at $g = 2$	Reviewer (`P3-08`)	Low	Fixed in diff

3 Empirical Catalog of 31 Numbered Defect Groups

Tables 2, 3, and 4 record all 31 numbered confirmed defect groups caught across Projects P1–P4, O-03, Tracks T1–T7 (Waves 1–3C), the post-T5 Paper 4 genus push, and clean-machine Linux `x86_64` QA of `lean-slice-oss`. Several numbered entries bundle closely related sub-issues (e.g. #5, #14, #16, #26, #30, #31) and are assigned to a single primary detecting layer and primary defect class in Table 5; additional notes outside the 31 numbered rows (including earlier P4-06 edge cases, `reports/O-08.md`, and Finding B1) are summarized below. Every entry corresponds to an archived audit report, diff, or receipt on disk (Appendix A).

Context on Additional Cross-Model Findings Outside the 31 Groups. Between Wave 2 and Track T5, cross-model refereeing of P4-06 (`CROSS_AUDIT_CODEX_P4_06.md`) bypassed the submitted Weierstrass engine via a separate truncated-branch DVR replay across 123 places / 130 closed orbits, reproducing 3,661/3,661 delta units under (G1)–(G3) while catching a Ferrari-

cubic prose slip, proving strict arithmetic non-integrality (branch order -2) at $L_{\pm}, b = \infty$, and flagging five defensive edge cases whose subsequent repair triggered Group #26. Earlier in Wave 1, Claude Opus identified an obstructive universal quantifier ($\forall v_5$, Finding B1) in `Genus2C5AlgebraicData`, later resolved by `Genus2FixedChartBridge.lean` proving `exists_transcendental_nonvanishing_translate` in Track T3. Additional attribution notes are preserved in `reports/O-08.md`.

4 Detection-Layer Matrix and Structural Complementarity

Table 5 aggregates all 31 numbered confirmed defect groups by primary defect class and primary detecting layer. Both the row totals ($\{11, 8, 4, 5, 3\}$) and the column totals ($\{0, 3, 22, 5, 1\}$) sum to 31.

Primary-detection-layer attribution rule. Every numbered group is assigned to exactly one primary detecting layer according to the *mechanistic trigger* that first surfaced the failure, rather than the identity of the operator who initiated the run. Under this rule, Group #5 (seven Linux replay portability bugs discovered during a clean-Linux run operated by a human colleague) is classified under *Linux* rather than *Human*, because the failures were triggered mechanistically by clean-host Linux execution (`elan` setup, tool pinning, missing `.olean` hashes, and `#print axioms` coverage) rather than by human reading or editorial review. Similarly, multi-layer Group #14 is assigned to its primary recomputation layer (*Cross-AI*), and the *Human* column (1/31, Group #6) records only defects caught directly by human organizational oversight outside automated execution or cross-model review.

To make every interior cell uniquely reproducible from Tables 2–4, the complete cell-by-cell ID allocation is:

- *Formal/spec drift & overclaimed Tier A* (11): Linux (#1); Cross-AI (#3, #8, #12, #17, #18, #19, #23, #28); Blind/Ext. (#2, #11).
- *Math/algebra/certificate error* (8): Cross-AI (#4, #13, #14, #15, #22, #24, #29, #30).
- *Literature attribution & concurrency* (4): Cross-AI (#25); Blind/Ext. (#7, #9, #16).
- *Verification script / receipt / parser bug* (5): Linux (#5, #31); Cross-AI (#20, #21, #26).
- *Prose/patch regression & logistics* (3): Cross-AI (#10, #27); Human (#6).

Four structural observations emerge from this expanded sample.

1. **Zero of the 31 post-compilation defect groups were caught by Lean 4's kernel alone.** This neither indicts Lean 4—whose kernel rejected countless proof errors *during* authoring before artifacts entered audit—nor measures kernel recall. Rather, it reflects that residual defects surviving to post-compilation waves (statement drift, missing provenance hashes, algebra/sign errors in Tier B certificates, broken receipt verifiers, Linux linker and RSS-accounting bugs) lie strictly outside the kernel's trust boundary.
2. **Cross-model adversarial AI auditing accounted for 22/31 groups (71.0%), including all 8 math/certificate error groups and every critical regression in Waves 3A–3C.** Notably, cross-audits repeatedly caught *test-blindness bugs* (Group #24: W_3 quadrature checked only even n, hiding a wrong $(-1)^n$ sign on odd n; Group #26: unit tests exercised standalone helpers without running the patched verifier end-to-end, hiding a self-invalidating `script_sha256` pin; Group #30: a sign error in `PQ11R3` and false Case 1 deduction in `PQ13R1`) as well as *second-order patch regressions* (Groups #10, #14, #27). Crucially, adversarial review also generated positive mathematical breakthroughs: repairing Groups #22, #24, and #29 produced the explicit integral cycle $\lambda = (1 - t^4)\gamma$ covering all 28 twists, an all-n proof of $\mu_{\mathrm{eff}}(W_3) \leq 4.9225204$, and `Theorem PQ-All`.

3. **Clean-host Linux x86_64 execution gates caught 3/31 high-severity groups that escaped LLM review.** Alongside Groups #1 (def/abbrev reducibility mismatch) and #5 (7 Linux script portability bugs), Group #31 demonstrated that even after two full multi-AI audit waves (Codex and Claude) on lean-slice-oss, the very first live Ubuntu 26.04.1 x86_64 run immediately exposed four concrete POSIX build/runner bugs (D1–D4) plus a major Linux memory-accounting caveat (X1: an unlinked MAP_SHARED arena on RAM-backed tmpfs /tmp accounts 380.4 MiB under RssShmem, hiding it from RssAnon/RssFile, whereas on disk-backed ext4 /var/tmp 381.2 MiB resides cleanly in reclaimable RssFile).

4. **Checker diversity exposed tooling boundaries and empirical kernel-completeness constraints.** On a 26-non-blank-line Mathlib-free reproducer (PrincipalizationCertRepro.lean), lean, leanchecker, and nanoda_bin accept all declarations while con-leche/con-ron v4.36.0 decline with exit 2 on a reflexive mutual inductive. Simultaneously, cross-OS theorem-unfold ablation in lean-slice-oss overturned a widely tempting streaming optimization (Section 5).

5 Open Verification Tool Suite

To eliminate recurring failure classes documented in Sections 3–4, the campaign produced four open verification tools, summarized with Windows and Linux x86_64 benchmarks in Table 6.

Eliminating native_decide via KroneckerFpCertV2. Polynomial irreducibility and factorization certificates over $\mathbb{F}_p$ (p = 1,000,003, degree up to 269) previously tempted authors toward native_decide (which adds Lean.ofReduceBool to the trusted computing base). KroneckerFpCertV2 packs coefficients into a single GMP natural number at base $B = 2^{64}$, validates all $d + 1$ limb bounds in a single arbitrary-width big-integer bitwise mask operation (Nat.land, whose bit work scales with the packed operand width rather than $O(1)$ bit work—reworded per Group #23), and proves Rabin's general composite-degree irreducibility criterion (rabin_irreducible_general) purely inside the kernel (decide +kernel), passing all five checks with zero warnings across 17 exported targets.

Bounded-Memory Verification and the Rat.instEncodable Theorem-Unfold Ablation. Memory pressure proved to be a first-class verification constraint on 32 GiB workstations, where stock nanoda_bin commits 1,020.07 MiB of Windows pagefile (526.72 MiB Linux RssAnon) on a single 445.01 MiB export (PostRHAbelianClosure.export). Because lean-export-stream--liveness shows that 60.95%–74.65% of exported expressions occur exclusively inside theorem proof terms, permanent post-check deletion of theorem proof bodies appeared attractive. However, live instrumentation of nanoda_lib~0.4.32 measured 94, 258, and 1,211 kernel theorem proof-body unfolds across DVRConductorNoGoingUp, KroneckerFpCert, and PostRHAbelianClosure. A causal ablation (NANODA_DISABLE_THEOREM_UNFOLD=1) executed identically on both Windows and Linux x86_64 confirmed that blocking Declar::Theorem unfolding passes 92,386/92,388 declarations across the three exports (0/8,410 failures on DVR, 0/20,441 on Kronecker) but causes *two hard kernel typechecker panics* (assertion failed: self.def_eq(u, v) at src/tc.rs:1036) on real Mathlib declarations Rat.instEncodable._proof_3 and Rat.instEncodable inside PostRHAbelianClosure.export (473 blocked attempts). Permanent deletion of theorem bodies is therefore *incomplete* for Mathlib exports. Instead, lean-slice-oss's 3-patch series backs ExprStore with an immediately-unlinked file-backed mmap arena (disk_arena_dir), preserving exact kernel semantics across 58/58 unit tests and 6/6 byte-identical cross-OS checker summaries while cutting Windows pagefile commit 4.16×–14.64× and Linux RssAnon 4.10×–8.51×. Notably, Linux run X1 measured the same 8.51× RssAnon reduction and 443.2 MiB VmHWM on both tmpfs (/tmp) and ext4 (/var/tmp); placing disk_arena_dir on a disk-backed filesystem (ext4/XFS, avoiding tmpfs and NFS; Group #31) is specifically what shifts the ∼ 381 MiB arena from non-reclaimable RssShmem (380.4 MiB on tmpfs) into kernel-reclaimable RssFile page cache (381.2 MiB on ext4).

6 Protocols and Recommendations

Seven operational rules directly address the 31 observed failure groups:

1. **Tag every claim with an epistemic tier, status, and pinned provenance.** Separate Tier A (kernel-checked), Tier B (deterministic script), and Tier C (unrefereed prose) explicitly; hash-pin transitive untracked dependency layers and exclude self-invalidating script hashes from result manifests (Groups #26, #28).

2. **Enforce cross-model adversarial review on every artifact *and every fix*.** Treat every patch as a fresh unverified artifact (Groups #10, #14, #27). Require referees to recompute certificates using a *distinct algorithm* and cover both parity regimes (Groups #24, #29, #30).

3. **Inspect cited preprints and upstream licenses at source level.** Never summarize an external formalization or assume license inheritance across distinct archives (Groups #9, #23, #25); audit actual target signatures, axiom logs, and `LICENSE`/`NOTICE` files.

4. **Require self-asserting scripts, end-to-end patch tests, and multiline receipt parsers.** Every numerical generator must assert its own residual and stationarity conditions (Groups #14, #30), patch test suites must execute the patched CLI end-to-end (Group #26), and axiom parsers must handle wrapped multiline `#print axioms` output with exact target-count checks (Group #21).

5. **Report checker codebase independence and completeness evidence honestly.** Distinguish checker invocations from distinct implementations, keep checker declines (exit 2/3) distinct from soundness rejections, and test proposed kernel optimizations by causal ablation (as with `Rat.instEncodable`).

6. **Run clean-host Linux `x86_64` QA before freezing releases.** A clean-machine Linux run catches `def`/`abbrev` mismatches, linker flags, `PATH`-shadowed compilers, missing headers, and `tmpfs` `RssShmem` accounting traps (Groups #1, #5, #31).

7. **Schedule memory-intensive verification sequentially with explicit RAM budgets.** Always enforce single-threaded FLINT/Lean execution (`flint.ctx.threads = 1`, `LEAN_NUM_THREADS=1`, `RAYON_NUM_THREADS=1`) behind a mutex wrapper on 32 GiB workstations and archive full stdout/stderr logs with SHA-256 manifests.

6.1 Post-v3 Pre-Submission Release-Bundle Audit

To preserve the frozen campaign audit boundary, Table 5's 31-group matrix is left unaltered. This final pre-submission release-bundle audit was executed by an additional AI review layer (Anthropic Claude in an interactive chat session directed by the human author) rather than an external human peer reviewer. Its end-to-end review of the assembled multi-paper release bundle surfaced five concrete packaging, provenance, reporting, and prose-proof defects that slipped past per-project campaign waves:

1. **Missing core C source in Paper 06 (`lean-slice-oss`):** packaging dropped `c_lean/ReadonlyArchive_portable.c`, breaking the `Makefile` build and reducing internal `SHA256SUMS` integrity to 64/65.

2. **Stale reproducer receipts in Paper 06:** all three Mathlib-free Lean reproducers (`ConLecheMutualRepro`, `ConLecheNestedRepro`, `PrincipalizationCertRepro`) had been edited after verification, causing their recorded `source_sha256` digests to disagree with the packaged files.

3. **Author-metadata drift and missing AI disclosures:** companion papers exhibited an inconsistent author first name across drafts (unified to single human author Robert Kadlec),

missing affiliation/ORCID headers, and omitted AI-provenance disclosures in Papers 05 and 08 and the Paper 08 cover letter.

4. **Undisclosed spurious Siegel poles in Prime Spectral Lab:** the shipped `README.md` omitted real-axis false positives (`spurious_siegel_poles_real`) already flagged inside the JSON output, an unstated $\beta \geq 0.8$ sensitivity floor, a 100k vs. 200k parameter mismatch, and a hardcoded Windows output path.

5. **Minkowski vs. Hermite prime-bound proof gap in Paper 08:** invoking Minkowski's convex-body area argument yielded $\lceil (2/\sqrt{3})\sqrt{p} \rceil$ rather than the stated floor $\lfloor (2/\sqrt{3})\sqrt{p} \rfloor$, subsequently closed in two lines via Hermite's constant $\gamma_2 = 2/\sqrt{3}$.

These five post-v3 findings provide direct empirical evidence that per-artifact verification waves do not suffice on their own: any multi-paper AI-assisted release requires a dedicated *final release-bundle audit layer* that rebuilds every archive from scratch in a clean directory, checks internal manifests and receipt hashes against shipped files, executes every published script, audits cross-paper authorship and AI-disclosure uniformity, and verifies every prose proof step before deposit or submission.

7 Conclusions and Limitations

Across 31 numbered campaign defect groups and five post-v3 release-bundle findings, the central empirical conclusion is structural complementarity: formal kernels eliminate proof-step invalidity inside formalized statements, clean-machine Linux execution catches build, portability, and memory-accounting failures invisible to text-based review, cross-model AI auditing exposes mathematical, specification-drift, and second-order patch defects outside the kernel's boundary, and a dedicated end-to-end release-bundle audit catches packaging omissions, stale receipts, and metadata drift across assembled archives. Crucially, the headline observation that cross-model AI audits caught 22/31 (71.0%) of the campaign defect groups is *conditional on cross-model AI audits having been executed most frequently* in this single-project, single-author case study with no control group; it should be interpreted as realized detection share under our deployment cadence rather than an intrinsic per-run recall advantage over clean-machine testing or human review.

Limitations. This study documents one intensive research campaign directed by a single human author (Robert Kadlec) without a controlled experimental baseline; none of the underlying mathematical manuscripts has yet completed traditional human journal peer review. Our counts reflect defects discovered by our audit layers, not an unknown ground truth of all latent prose errors.

A Campaign Receipt and Audit Index

All 31 defect groups and tool benchmarks above are grounded in archived read-only reports and JSON receipts:

- Master defect registry and cross-audit waves: `02_SLEPE_CESTY_BARIERY_A_CHYBY.md` (Version 5, 31 groups), `reports/O-08.md`, `reports/CROSS_AUDIT_CLAUDE_WAVE1.md`, `reports/CROSS_AUDIT_CLAUDE_WAVE2.md`, `reports/CROSS_AUDIT_CLAUDE_WAVE3B.md`, `reports/CROSS_AUDIT_CODEX_WAVE1.md`, `reports/CROSS_AUDIT_CODEX_WAVE3A.md`, `reports/CROSS_AUDIT_CODEX_P4_06.md`, `audits/AUDIT_CODEX_T3_T4_T7.md`, `audits/AUDIT_CLAUDE_T1_T2_T6.md`, `audits/AUDIT_CODEX_T5_DVR.md`, and `p4_genus_push_20261128/AUDIT_CODEX_P4_PUSH.md`.

- Tool and cross-OS receipts: `wave2_verify/WAVE2_MASTER_SUMMARY.json`, `wave2_verify/receipts/T4_KroneckerFpCertV2_WAVE2.json`, `lean_slice_oss_20261128/receipts/o04_cross_archive_audit.json`, `lean_slice_oss_20261128/receipts/theor`

em_unfold_ablation_windows.json, and lean_slice_oss_20261128/receipts/linux_20261128/REPORT_EN.md (including linux_theorem_unfold_ablation.json and linux_posix_mmap_runtime.json).

References

[1] K. Buzzard, *Mathematical reasoning and the computer*, Bull. Amer. Math. Soc. **61** (2024), 211–224.

[2] G. V. Chudnovsky, *The irrationality exponent of the lemniscate period: A determinant proof with a Lean 4 formalization*, arXiv:2610.26539, October 2026.

[3] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in Automated Deduction – CADE 28, LNCS 12699, Springer, 2021.

[4] The mathlib Community, *The Lean mathematical library*, in Proceedings of CPP 2020, ACM, 2020.

[5] A. Bailey, *nanoda: An independent typechecker for Lean 4 exports*, 2025–2026.

[6] V. Kh. Salikhov, *On the irrationality measure of* π, Russian Math. Surveys **63** (2008), 570–572.

[7] D. Zeilberger and W. Zudilin, *The irrationality measure of* π *is at most* 7.103205334137…, Moscow J. Combin. Number Theory **9** (2020), 407–419.

Table 3: Confirmed defect groups #12–#21 (Projects P3, P4, O-03, and Waves 1–3B).

#	Proj.	Defect / Vulnerability	Caught By	Sev.	Status
12	P3	`MultivarHenselLemma29.lean` proved normalized rational core, not full $\mathcal{O}_{\mathbb{L}}$ denominator or truncated primitive G_i of Lemma 2.9	Codex Wave 1 audit (**FAIL** for full L2.9)	High	Closed in T3 (`MultivarHenselFullLemma29`)
13	P3	`Genus2BetaPeriodIdentity.lean`: sign discrepancy on C_5^+ ($1-4x^5$ vs. $4x^5+1$); single-sheet ray on C_5^- missing oval factor 2	Codex Wave 1 audit	Med.	Explicit ray/oval labeling
14	P3	Divergent Flint–Hills example ($a=b=2$, later $a=b=3$) in prose; $\sqrt{2}$ error in unflagged $B(3/8,3/8)$ certificate script	External + Codex/Claude recomputation	High	Fixed in `v2.2.0`
15	P4	Special-fiber branch filter mistaken for arithmetic integrality over $\mathbb{Z}_{(p)}$ + hardcoded `holdout_verified = True` flag	Codex + ChatGPT 6 Pro pre-`v1.0.0`	Crit.	Honest enclosure + P4-06/T5 proofs
16	P4	Unused `conic_*` lemmas in `DVRConductorNoGoingUp`; overblown abstract; missing OpenAI $\mu(\pi)=2$ citation and companions	Blind cross-model review (`P4-01/02/03`)	High	Fixed in master upgrade
17	P4	Lemma 3.2 general-p remark omitted primes 17 and 41 dividing leading coefficients $-272, -164$ (not the resultants)	Claude Opus Wave 2 (`P-C3`)	Med.	Added $\{2,3,5,11,17,41\}$
18	P4	`LocalClusterLemmas.lean` omitted $\mathbb{Q}(C)$-descent and $\vartheta/(R-25)\vartheta_{25}$ integrality while report called Lemma 3.3 "complete"	Claude Opus Wave 2 (`P-C1,P-C2`)	Med.	Tier A table narrowed
19	O-03	Benchmark files `BenchB350.*` deleted mid-audit; standalone `KroneckerFpCert` lacked a concrete end-to-end instance	Claude Opus Wave 2 (`K1`)	Med.	Closed in T4 (`KroneckerFpCertV2`)
20	P3/P4	Per-target `#print axioms` stdout transcripts not archived in JSON receipts for `P3-04(a,b)`	Codex Wave 1 audit	Low	Archived in T1 D5
21	P1	Single-line regex parser in `verify_p1_lean5.py` dropped 1 of 21 targets (long name wrapped over 3 lines; raw stdout clean)	Claude Opus Wave 3B (`P1-R1`)	Med.	Multiline parser in T1 D2

Table 4: Newly catalogued confirmed defect groups #22–#31 (Campaign Waves 3A–3C across T1–T7, post-T5 Paper 4 genus push, and clean-machine Linux `x86_64` QA of `lean-slice-oss`).

#	Proj.	Defect / Vulnerability	Caught By	Sev.	Status
22	P3 (T3)	Wrong integral lattice $(1-\zeta_8)^2\mathcal{O}\beta$ (index 4 vs. 2/8) in §B.5; refuted general two-scale weight impossibility via $E \times E$ counterexample	Codex Wave 3A (`T3-B4,T3-B5b`)	High	Fixed ($\lambda = (1-t^4)\gamma$ cycle, 28 twists; §B.4 retracted)
23	P2/P4 (T4)	Overclaimed `SqrtM23ClassGroup3` as $h = 3$ (proves $3 \mid h$); inaccurate closure counts (57 mods); `Nat.land` $O(1)$ bit-work claim; missing license grants	Codex Wave 3A (`T4-04..06`)	Med.	Fixed in roadmap/audit; Apache-2.0 pilots added
24	P4/Meta (T7)	Wrong sign $(-1)^n$ on odd n in W_3 integral (quadrature tested only even n); misassigned single-branch saddle $t \approx 0.2675$ to Paper 4 curve	Codex Wave 3A (`T7-W1,W2,S1`)	Crit.	Fixed (all-n proof $\mu_{\text{eff}}(W_3) \leq 4.9225204$; saddle retracted)
25	Meta (T7)	`prime_spectral_oss` README claimed "34 consecutive zeros" (actual indices 1–33, 36); `O-08` misattributed two authors to [2]; `OrbitWindows` Table 1 $L = 6, 24$	Codex Wave 3A (`T7-P2,M2,O2`)	Med.	Fixed across README, PDFs, and SHA-256 ledger
26	P4 (T1)	**BLOCKING** `T1-B1`: hardened P4 verifier asserted pre-patch `script_sha256` invalidated by the patch itself; strict non-`None` omitted in 2 scripts	Claude Wave 3B (`T1-B1,M3,M4`)	Crit.	Fixed (separate manifest, 7 strict files, 9 tests incl. 2 E2E PASS)
27	P1/P4 (T1)	False "i.e." claiming prime 3 divides the 9 displayed constants (only denominators $24, 6$); invalid 7th JSON key `_draft_comment`; stale TeX patch base	Claude Wave 3B (`T1-M2,M5,X1`)	Med.	Fixed (reworded, 6-key JSONs, 3 rebased patches)
28	P1/P2 (T2/T6)	Untracked `Track2Grover/` semantic layer omitted from T2 receipts (`T2-M1`); `T6_REPORT` wrote in present tense about unelaborated 4.35↔4.33 application	Claude Wave 3B (`T2-M1,T6-M1`)	Med.	Fixed (157-file `dependency_sha256`; T6 scoped; parametric `T2-m1` bridge)
29	P4 (T5)	Only 2 of 54 finite `R2/R3` centres admit both regimes (34 R2-only, 18 R3-only); standalone `branch_orders` can drop a $y = 0$ branch on reduced nodes	Codex Wave 3C (`F16,F17,F22`)	High	Logged in `T5_REPORT` §10 (3,872 units intact; led to `PQ-All`)
30	P4 (Push)	Wrong sign $-1/(25-z)$ vs. $+1/(25-z)$ in `PQ11R3`; false Case 1 deduction in `PQ13R1` ($k = -20/9$ counterexample); Hensel helper accepted duplicate branches	Codex P4-Push audit	Crit.	Proofs, coverage & $g(C) \leq 7{,}423$ verified; stricter check in `verify_p4_push.py` (prod. helper unrefactored)
31	Tool (`OSS`)	4 Linux build/runner bugs (`D1` `cp -a` index, `D2` `leanc` missing `-lleanshared`, `D3` PATH shadowing `clang`, `D4` missing `lean.h`) + `X1` unmeasured `RssShmem` (380.4 MiB on `tmpfs` vs. `RssFile` on `ext4`)	Clean-machine Linux `x86_64` QA (Ubuntu 26.04)	High	Fixed in `run_linux_tests.sh` (`D1`), `build.py`, `Makefile`, `bench_nanoda_ab.py`, & docs

Table 5: Detection-layer matrix across all 31 numbered defect groups (rows: primary defect class; columns: primary detecting layer). Every row and column sum is exact.

Defect Class	Kernel	Linux	Cross-AI	Blind/Ext.	Human	Total
Formal/spec drift & overclaimed Tier A	0	1	8	2	0	11
Math/algebra/certificate error	0	0	8	0	0	8
Literature attribution & concurrency	0	0	1	3	0	4
Verification script / receipt / parser bug	0	2	3	0	0	5
Prose/patch regression & logistics	0	0	2	0	1	3
Total catalogued (31 groups)	**0**	**3**	**22**	**5**	**1**	**31**

Table 6: Verification and auditing tool suite with Windows and Linux `x86_64` benchmarks.

Tool	Tier / Lang	Primary Function	Measured Benchmark
`KroneckerFpCertV2` (`O-03/T4`)	Tier A / Lean 4	Pure-kernel $\mathbb{F}_p[X]$ arithmetic, Brent–Kung composition, general composite-degree Rabin irreducibility (`0 native_decide`)	2,822 code lines, 17/17 targets, 0 warnings, Wave 2 5/5 PASS (22.9 s `lean`, 21,282 `nanoda` decls; concrete deg 6 & 12 irreducibles); V2 engine also powers `GeometricMonodromyAndRigidityCert` (deg 269/267, 26/26 targets); earlier V1 engine certified `MonodromyS269Cert` in 43.2 s `lean`.
`lean_dag_audit` (`O-04`)	Tier B / Python	Source-tree import DAG, Cooper–Harvey–Kennedy dominators, nested-comment forbidden-token scan	Chudnovsky [2] (976 mod., 114,709 lines, 0 forbidden tokens, 225-mod. blowup subcone); `lsernau/grh-lean` (`grh-compact`: 794/794 mod., 112,827 lines; `grh`: 803/803 mod., 477,124 lines).
`stream_export_audit` & `lean-export-stream`	Tier B / Python + Rust	Single-pass NDJSON `lean4export 3.1.0` strict schema, back-ref, tagged-axiom auditor + `-liveness` model	Python: 6.738 MiB Win / 6.750 MiB Linux traced heap on 34.38 MiB export (12/12 tampers); Rust: 88.9–95.9 MiB/s Win, 64.9–69.9 MiB/s Linux EPYC (32/32 tamper checks; 60.95%–74.65% proof-only exprs).
`lean-slice-oss` (`SliceFrontendV2_OSS` + `nanoda_lib` patches)	Tool / Lean 4, C, Rust	Portable Windows/POSIX zero-commit `.olean` mapping, native `LeanXRayDump`, and file-backed `mmap` `ExprStore` + `W2-1` fix	Live Linux `x86_64` QA (`gcc 15.2`, `clang 21.1`, 4/4 POSIX `mmap` tests, 58/58 `cargo test`); cuts `nanoda` Win pagefile commit 14.64× (1,020.07 → 69.70 MiB) and Linux `RssAnon` 8.51× (526.72 → 61.86 MiB, 381.2 MiB reclaimable `RssFile` on `ext4`).