A library-agnostic hybrid classic/PQ OpenSSL provider — built with AI

What a human *actually* does when an AI writes the code

Michael Baentsch · OpenSSL Conference 2026

OpenSSL CONFERENCE

OpenSSL CONFERENCE

Overview

- Background — a sequel to last year
- What got built (and how fast)
- What the human actually *did* to get things right
- Findings: CI is not the safety net
- DO · DON'T · the workflow that worked
- Take-aways
- *(Bonus, if time)* what if Claude had built oqs-provider?

OpenSSL CONFERENCE

Background — a sequel

- Last year, here: *"Implementing oqs-provider — or: how to add experimental PQC to OpenSSL."*
- It ended on a **strawman**: a smaller, provider-agnostic *hybrid-provider* using only the public EVP API.
- This year that strawman is **built** — the **hybrid-provider**.
- The twist:
  - **Written entirely by an AI** (Claude Code — **Claude Opus 4.8**); one human *guided*, did not type code.
  - 13 active days · 22 chat sessions · **107 commits** across ~28 feature branches (~10 PRs).
- Transcripts analyzed. Two questions:

> *1. How much got built, how fast? 2. What did the human actually do?*

OpenSSL CONFERENCE

What got built — in 13 active days

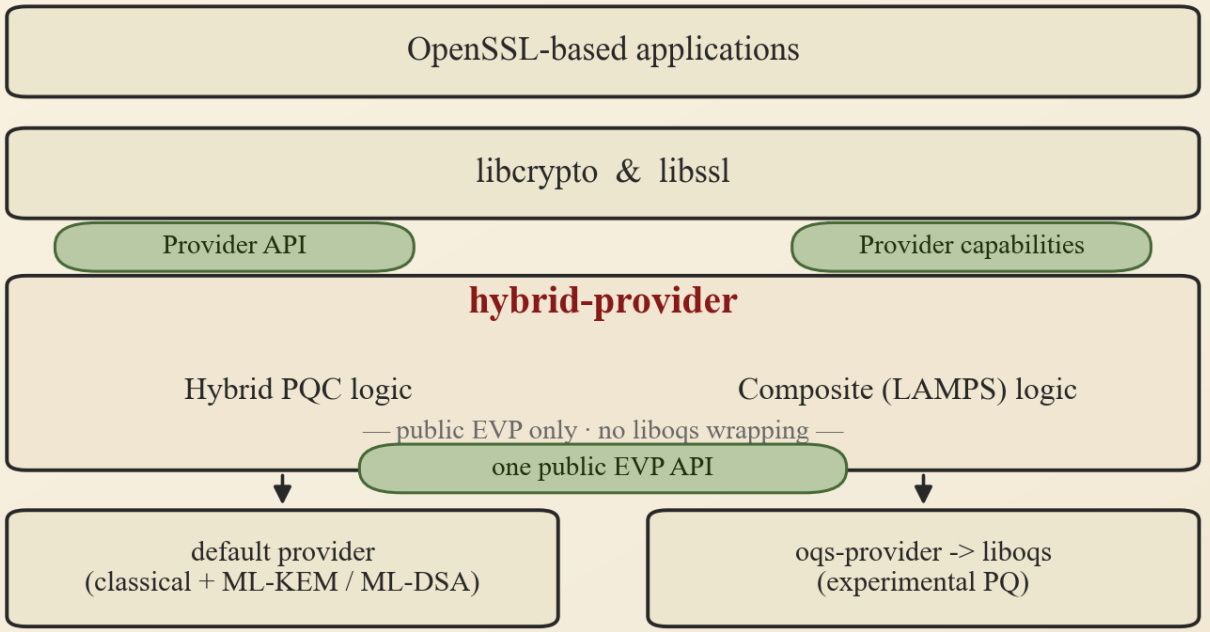
One human guide + one AI. `git` -measured; vendored checkouts excluded.

	hybrid-provider
Active development	**13 commit-days** (11-day sprint) · 107 commits
Lines authored	**~25,000 inserted** (~1,900 / active day)
Core provider code	**9,071 C/H** (47 files) + **6,910** test
Delivers	hybrid KEM + hybrid sig **+ LAMPS composite sig + composite KEM** + enc/dec + TLS caps + drop-in cede

A working, interop-tested OpenSSL provider in two weeks.

*The AI, concretely: **99.6% Claude Opus 4.8** — with a sliver of Sonnet 4.6 (a single session).*

The design — delegation, not duplication



Composes component algorithms through the public EVP API -> ~34% less code, provider-agnostic

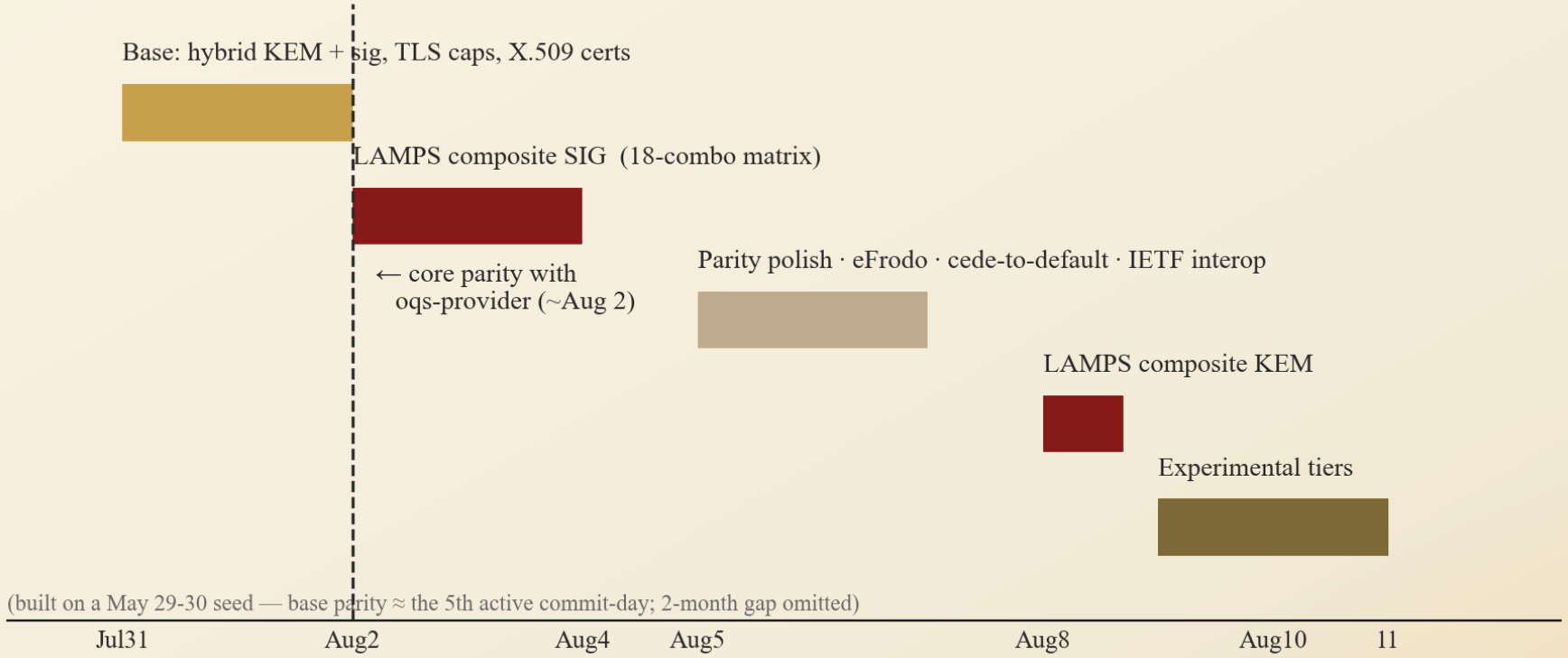
...versus the incumbent

	hybrid-provider	oqs-provider
Who	1 human + 1 AI	multi-contributor
Time	**13 active days**	**~5.5 yrs · 278 commit-days**
Core code	**9.1k** C/H	13.8k C/H
Hybrid KEM + sig	✔ full parity	✔
LAMPS composite	✔ sig + KEM	✖ **none**

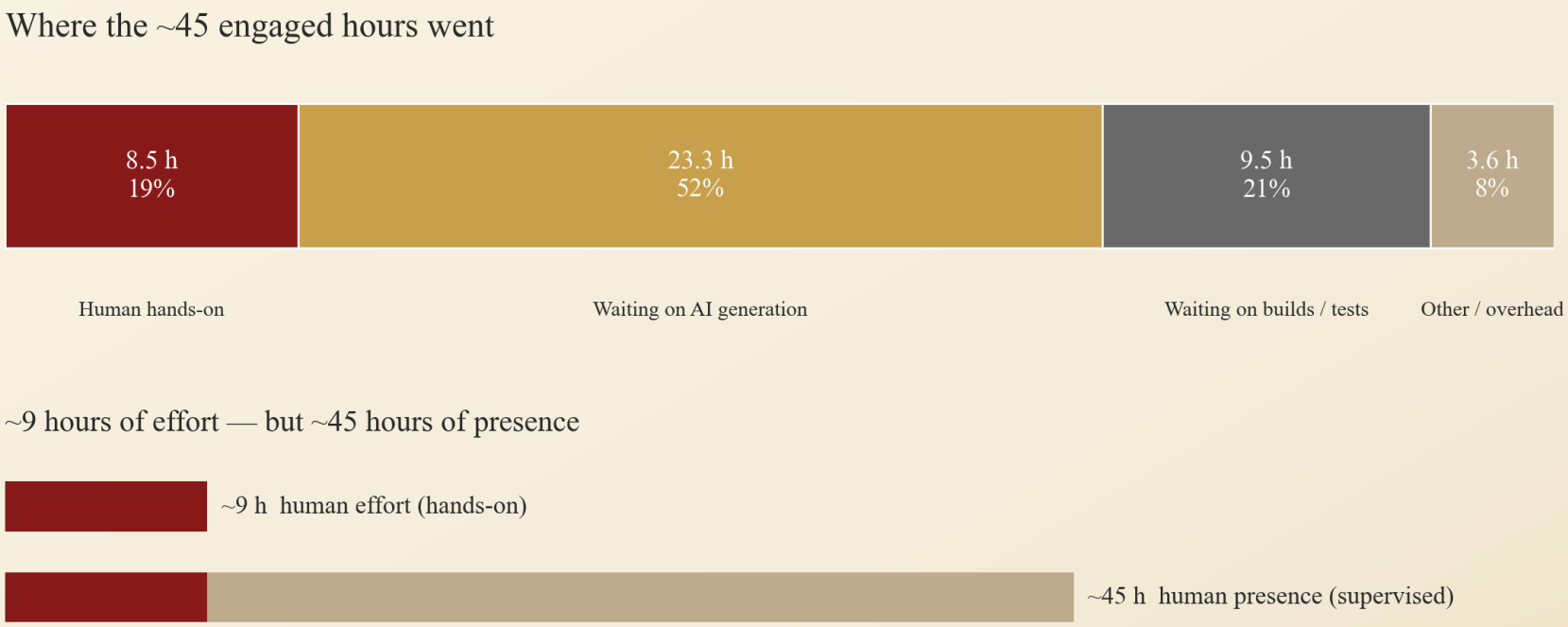
~5 sprint days → functional parity with oqs-provider – same hybrid KEM+sig surface, ~34% less code, cleaner structure.

Beyond it – LAMPS composite sig (18-combo matrix) + composite KEM, neither in today's oqs-provider – two short bursts (~1.5 d + ~1 d).

The build order: base parity → composite sig → composite KEM



"13 days" is calendar time. Real effort was ~45 h.



~9 h hands-on · ~23 h waiting on the AI · ~10 h on builds — but the human was *present* for all ~45 h. That presence is the point, not overhead.

So the productivity story is real. But.

> *A superset of oqs-provider, a third less core code, ~13 days.*

That is the headline. The rest of this talk is the **asterisk**:

What did that one human have to *do* to make "fast" also mean "correct"?

The dataset (so you can trust the *governance* numbers)

- **456** genuine human turns extracted (tool-results / command-echoes filtered out).
- Every turn **hand-classified** (semantic judgement, not keyword matching).
- Split **correction/steer vs. new-task**, each correction tagged:
  - *"would CI or the passing test suite have caught this?"*
- Fully reproducible: `extract.py` → `classify.py`.

Turns	Corrections	New tasks / approvals / bookkeeping
456	211 (46%)	245

≈ every second thing the human said was a correction or a steer.

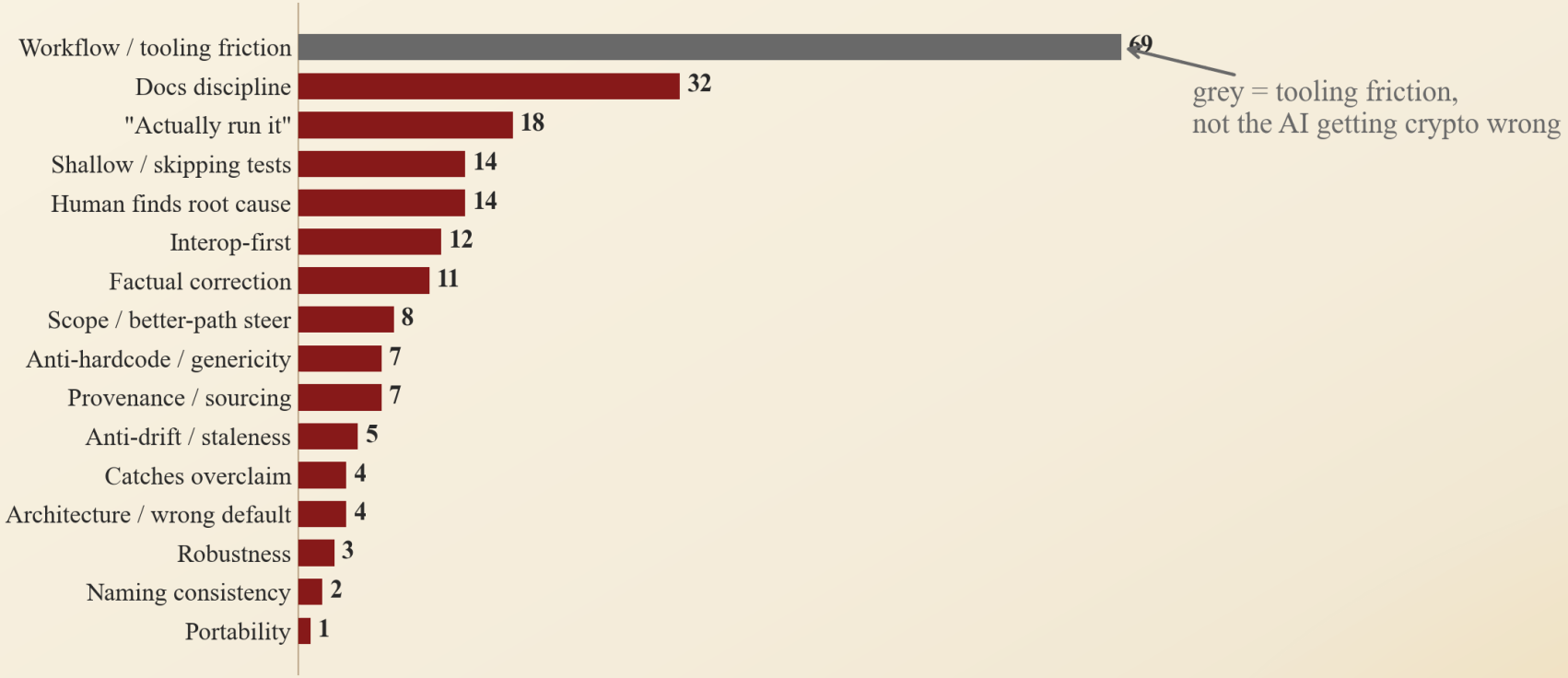
~0 of 211

corrections would have been caught by CI or the green test suite

Every single one was a **judgement call**: design generality, provenance, API-legitimacy, factual floor-knowledge, *"did you actually run it?"*, honesty.

Finding 1 — a *risk profile*, not random noise

The 211 corrections cluster into a few recurring kinds — the same ones for 13 days:



→ A *teachable checklist*, far more useful than "AI makes mistakes."

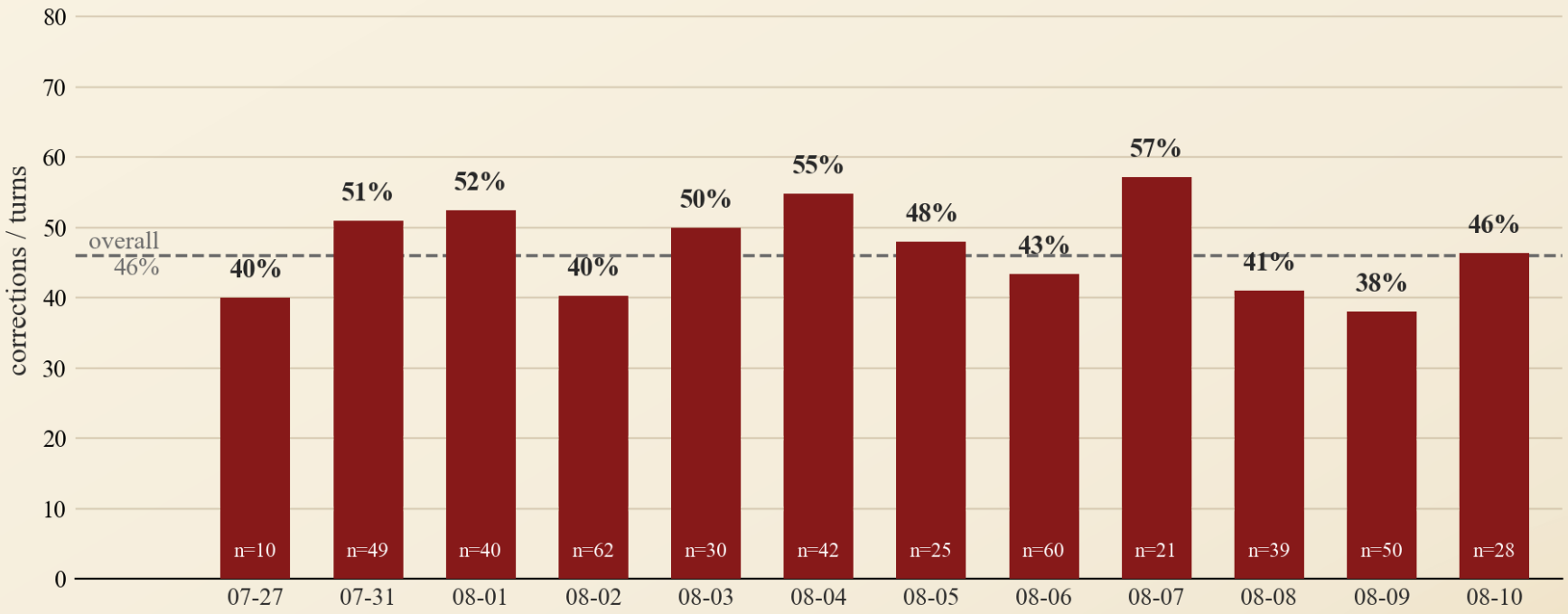
Finding 2 – CI is not the safety net. The human is.

- 18 turns were the human insisting: *"actually run it and show me."* — across **8 sessions, 10 branches.**
- This repeatedly beat green CI:

> *The experimental composite-KEM tier **passed CI** — because the rows were **silently skipping.** Running it locally surfaced two real bugs.*

- *"why can't you execute the tests locally?"*
- *"did you run this against **all** published artifacts?"*
- *"first confirm by name-matching that **all** ceded algs are actually served."*

Finding 3 — guardrails cap old mistakes, not the mistake *class*



Early ~48% → late ~44%. Essentially flat. Memory stopped *exact* repeats; the AI re-made the *same class* of mistake on every new surface. **Plan for continuous review, not a ramp-down.**

† Data per captured chat-day.

Finding 4 – the human's *mental model* drove the architecture

The two most important design properties were **human-reasoned**, against AI defaults:

- **Genericity / substitutability** — enforced repeatedly, sometimes twice in minutes:

  > *"anything in default is ceded, everywhere, incl. future algs" →*
  > *"the CMakeLists still references only the MLX group — did you check everywhere?"*

- **The seed-API 3.5 boundary** — the human reasoned to it first:

  > *"if composite needs the ML-DSA seed — that's not available below 3.5, right?"*

  …which is exactly what later **unlocked the <3.5 and experimental tiers.**

The AI executed brilliantly. It did not *decide* well.

Recurrence — the same steer, again and again

Theme	turns	sessions	branches
"Actually run it" (verify)	18	8	10
Docs discipline	32	10	14
Human supplies the root cause	14	7	6
Interop-first / no self-contained	12	6	7
Factual correction	11	6	8
Genericity / anti-hardcode	8	3	3
Provenance / standards	7	5	5

Corrections are systematic and cross-cutting — you can prepare for them.

In the human's own words

> *"be sure to **not duplicate code per algorithm**, but generalize as much as possible"*

> *"is the empty digest marker a **trick** to get CMS to work, or is it **documented**?"*

> *"do **not** phrase anything favouring ML-DSA — a true independent research perspective"*

> *"the #808 fix is **not a true fix**, it silences the warnings"*

> *"what function name is `inferred_keytype`? that sounds **heuristic / failure-prone**..."*

DO

- Be the **architect & verifier**; let the AI be the fast, tireless **executor**.
- **Make it run.** Observe real output — never accept "CI is green" as proof.
- State the **generality contract** up front; re-check it *everywhere* after each feature.
- Demand **provenance** for every constant, code point, OID, "documented behaviour."
- Keep a **persistent memory** of standing constraints (it stops exact repeats).
- **Front-load your domain knowledge** — version floors, spec traps, key formats.

DON'T

- Don't assume memory/guardrails make review **optional over time** — the rate stayed flat.
- Don't let the AI ship the **specific case as if it were the general one**.
- Don't trust **green CI**, self-contained round-trips, or advertisement-only checks.
- Don't accept *"tracked as follow-up" / "known limitation" / "can't be done"* without pushing — several were real, fixable, or fabricated.

How to proceed — the workflow that worked

- **Commit/push often** on feature branches (this repo survived *two* PC crashes that way).
- **One feature → one branch → one PR**; review the *full* diff before landing.
- Route **expensive matrix/interop tests to weekly CI**; keep push/PR CI fast.
- Add **anti-drift mechanisms** (name-diff test, headline-count guard) so you don't police staleness by hand.
- Keep **benchmarks reproducible-on-demand**, not committed; phrase results **neutrally, per-axis.**

Take it home

The speed is real. So is the asterisk.

- **Output:** a superset of oqs-provider, **~34% less core code, ~13 days.**
- **Cost: 46%** of turns were human corrections — and **~0%** were CI-catchable.
- The correction rate **did not fall** over two weeks.
- The most valuable thing the human did: **insist the code be run and observed.**

AI writes the code fast. You own the judgement that makes it safe.

Green CI plus a human beats green CI alone – and it isn't close.

Thank You!

Participate at github.com/baentsch/hybrid-provider

Contributions — *including AI-authored ones* — very welcome.

Questions? · @baentsch · info@baentsch.ch

PS — a bonus "what if Claude had built oqs-provider?" follows, if we have time.

OpenSSL CONFERENCE

Bonus track

What if Claude had built oqs-provider?

A thought experiment — not part of the build story. *If we have time…*

What if Claude had built oqs-provider?

OpenSSL CONFERENCE

	AI-COMPRESSIBLE	NOT AI-COMPRESSIBLE
YOU OWN THE GATE	**A · AI WINS BIG** • Composite sig matrix • New alg families • OID / code-point grind	**D · YOUR JUDGEMENT** • Reviewing the AI • 211 / 456 = 46% corr. • ~0 CI-catchable
SOMEONE ELSE OWNS THE GATE	**B · THE MIDDLE** • B1 PR #19312 -> 3.2 • B2 NIST FIPS finalising • B2 LAMPS drafts	**C · AI-IRRELEVANT** • Governance #342/#664 • Composite removal #707 • 3.5 native collision

Effort collapses in **A**; **calendar** moves **only in B1** — **B2** (standards) & **C** (governance) set the floor AI can't touch.

OpenSSL CONFERENCE

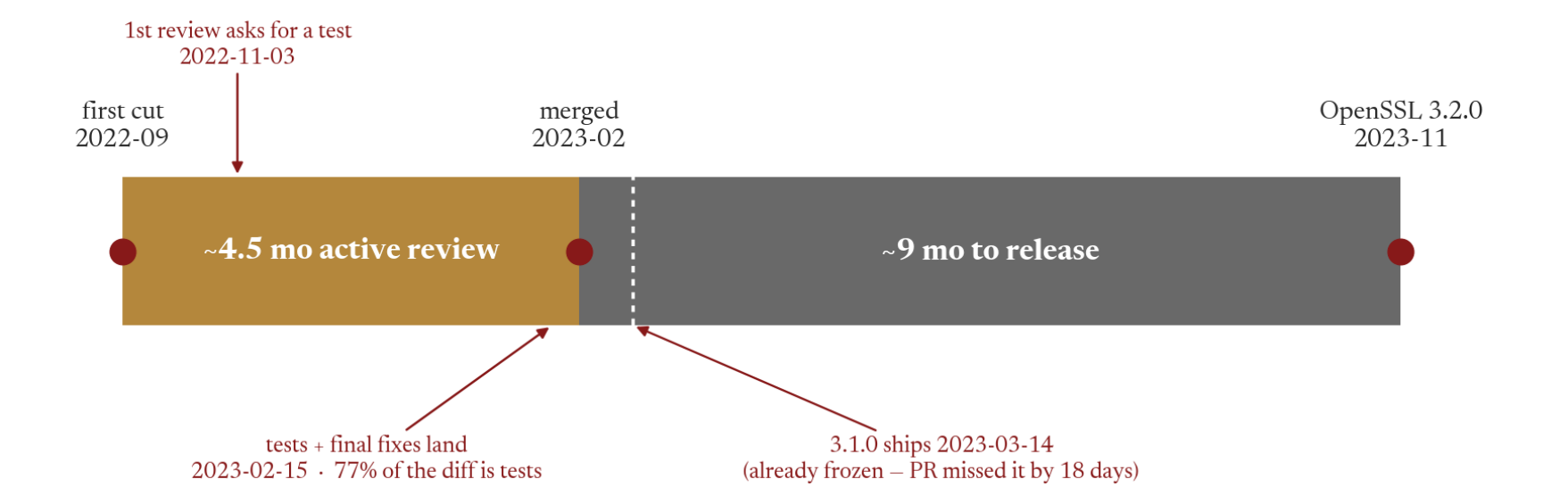
One example per quadrant

Cell	Real example	What AI does	The honest bound
A own · compressible	Composite sigs, ~7 mo matrix/OID/KAT (#317→#549)	Collapses the engineering (here: full 18-combo sig matrix + KAT interop in **~1.5 days**, composite KEM in **~1 day**)	2 of those months were **draft-02 churn** (#454) — a B2 sliver inside an A task
B1 co-owned upstream	**Provider TLS-SIGALG API, PR #19312** → OpenSSL 3.2	Faster drafting → reviewed **sooner** → hybrid sigs ride an **earlier train**	**~5 mo review + ~9 mo to release**; *missed 3.1 by 18 days*. Best case ≈ one cycle
B2 standards-gated	Kyber/Dilithium R3 → ML-KEM/ML-DSA (#603)	Compresses re-coding once the spec lands	Cannot compress **NIST finalising** or an **IETF draft moving**
C community-owned	GOVERNANCE (#342/#664); composite **removal** #707; **3.5 native** collision	**Nothing** — no code decides who maintains, or when upstream ships native PQC	AI *adds* a question: who owns/reviews the AI's PRs?

> ***AI moves the gates you own; it can't move NIST or a review queue — and speeding the writer just exposes the next gatekeeper.***

The one gate AI *could* move (B1)

Hybrid TLS **signatures** needed a provider capability that didn't exist — the author built it and drove it upstream (**PR #19312**, +3397/−238):



Earlier submission ≠ earlier availability. AI speeds the **writer** — not the release cadence, nor the reviewer.