OpenSSL CONFERENCE

Dr. Tokenlove or: How Bouncy Castle Learned to Stop Worrying and Love the LLM

David Hook, Legion of the Bouncy Castle/Keyfactor

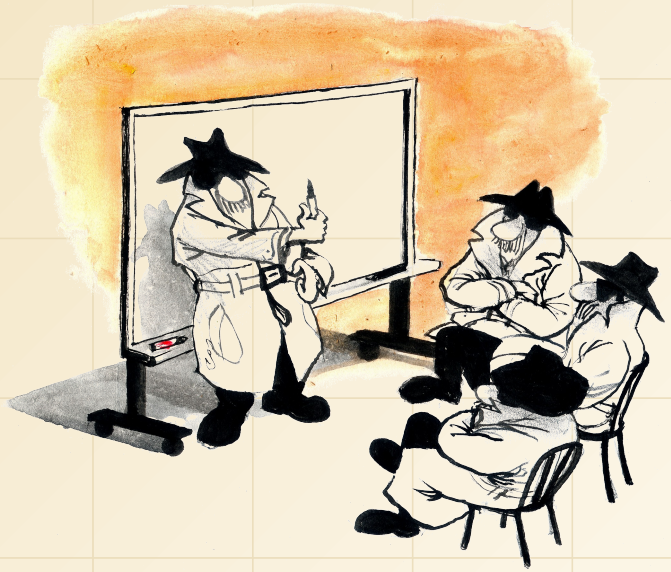
§ PRIVACY AS A FUNDAMENTAL HUMAN RIGHT

14.10.2026

Bouncy Castle in one slide

OpenSSL CONFERENCE

- 25+ years of open-source cryptography — Java and C#
- Enormous API surface: JCE provider, TLS/JSSE, CMS/PKIX, OpenPGP, S/MIME, PQC…
- Enormous Lines of Code: 775K BC Java, 321K BC C#.
- Multiple editions: mainline, LTS (native JNI acceleration), FIPS
- Maintained by a very small team relative to that surface
- Legacy reach: we still ship a Java 1.4 build.
- And 25+ years of accumulated technical debt…



How we started

OpenSSL CONFERENCE

- Seeing OpenAI Chat-GPT 3.5 (around middle 2023)
- Quite basic compared to now, but noticed it could produce “stack overflow“ quality answers on the main Bouncy Castle APIs, but perhaps not so good on FIPS.
- Started a support project for training an LLM and seeing if we could develop a “Support-bot“
- Support-bot project really kicked off in 2024, finally ready to ship early 2026.
- However, from 2025 to early 2026, LLM coding tools went from
  { “perhaps promising”, “annoying” } → { “enormously useful” }
- Not a gradual slope — a step change when tools became agentic and started shipping with a meaningful context (for us this meant Opus 4.6 and its trial 1 Meg of context)
- This talk: what we did, what worked, what broke — and how to tell the difference

An introduction to Coding Tools

OpenSSL CONFERENCE

- Suspicion that “something had changed“
- Tried a couple of things, mostly disasters
- Changed the approach a bit, some progress.
- Refined approach and “Oh my!“
- First introduction to “power“, issue with Native GCM decryption in LTS API.
- First reminder of “reality“, user question about EdDSA curve usage.

The velocity chart (and why it's the wrong metric)

OpenSSL CONFERENCE

- bc-java commits per quarter:
  - 2024 Q3 – 2025 Q1: 277 / 427 / 435
  - 2025 Q2 – 2026 Q1: ~186–227 per quarter
  - 2026 Q2: 877
- Raw commit count is a vanity metric...
- ...but the composition changed: hardening, tests, docs — work that never used to make the cut, retiring technical debt.



Teaching the tool the repo

OpenSSL CONFERENCE

- 4 May 2026 — initial CLAUDE.md (3ef319876c)
- May — repeated refinement: bans, conventions, build rules
- 29 May 2026 — outgrew one file → docs/claude/ (2c2ade336e):
  - architecture · build-and-test · build-jdk14 · conventions · upstream-lts-ports
- Then .claude/skills/ — parameterised, repeatable recipes
- Treat the tool like a new hire: onboarding docs, real information, not vibes
- Use the tool like you're pair programming – keen and (mostly) knowledgable typist, but a tad irresponsible if not watched

Institutional memory as executable docs

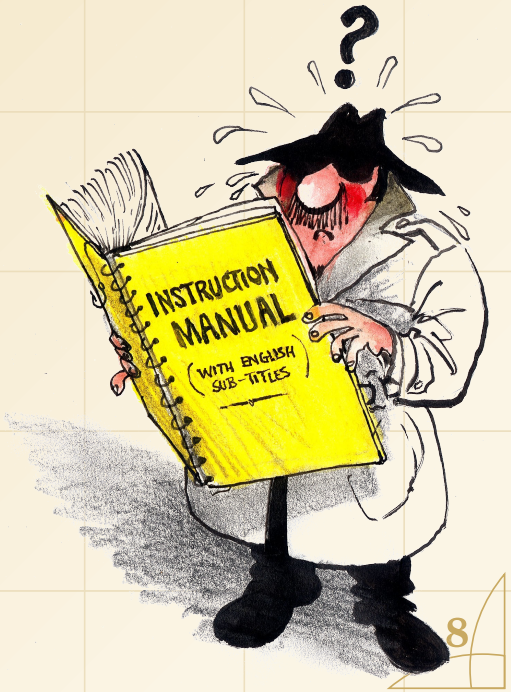
OpenSSL CONFERENCE

- Try to make every AI failure a documented guard. Actual commit messages:
  - “Record the false-green Gradle run and moving-HEAD traps, the base64-in-Java-source parsing hazard…” (596112dd35)
  - “test cannot run against signed jars because build-test always re-jars bctest” (b674527055)
- Tool falls in a hole → we fence the hole → it never falls in again
- Documents don't leave, get sick, or distracted.
- LLMs actually read them as well!
- Like code documentation, these documents are “living documents“.

Skills: recipes for repeatable jobs

OpenSSL CONFERENCE

- There's a surprising amount of regularity in our world
- Example: .claude/skills/bcpqc-add-signature-algorithm — the “add a PQC algorithm” playbook
- Case study: New PQC Signature Algorithm Process
  - BCPQC provider wiring, OID arc placement
  - exposure conventions, library conventions
  - javadoc conventions
- Time spent focusing on core implementation – not on making it fit.
- Socialisation of conventions much easier to carry out.



Win #1: your test suite is lying to you

OpenSSL CONFERENCE

- bc-lts-java, one audit sweep:
  - “Register 25 test classes that no task could reach” (c32e030a)
  - “Run the five bcpg packet tests that were present but never registered” (c1f12d56)
  - “Make the GCM blocks-remaining tests run and be able to fail” (58e12a24)
- Tests that exist ≠ tests that run ≠ tests that can fail
- LLMs are relentless at exactly this cross-referencing drudgery

Win #2: edge-case archaeology in native code

OpenSSL CONFERENCE

- bc-lts-java native/JNI layer — found by LLM audit:
  - CTR counter carry lost when a 16-byte IV's low lane wraps (752e66da)
  - ChaCha skip used a signed comparison where unsigned was needed (c8ede20d)
  - skip(Long.MIN_VALUE) silently moved nothing (3748e9ba)
  - Native CTR getPosition reported the counter, not the offset (e887b541)
- Common shape: boundary arithmetic, overflow, sign — where humans glaze over

Win #3: systematic hardening sweeps

OpenSSL CONFERENCE

- LTS sweep, one theme: never trust a wire-declared length
  - “Bound the BMPString allocation to the content that arrives” (d6d84167)
  - “Cap the bcrypt round count from an encrypted OpenSSH private key” (9f468ecb)
  - “Bound the depth of nested multipart content in both S/MIME canonicalisers” (06934838)
  - “Fix unbounded loop if HW entropy source fails” (23adedff)
- One class of bug, hunted exhaustively across the whole surface

The Mythos report

OpenSSL CONFERENCE

- Coordinated vulnerability disclosure: frontier-model (Claude Mythos-class) security analysis of Bouncy Castle Java and C#
- Based on a scan of the published GitHub repositories.
- 88 finding bundles → ~49 unique defects (6 high, rest medium)
- Status at triage close: 48 fixed, 1 open (by design, documented)
- Findings arrived with affected paths, attack scenarios, RFC citations, severity rationale
- These were on code bases that have been relentlessly scoured for decades

Two bytes, N years

OpenSSL CONFERENCE

- RSA PKCS#1 v1.5 verify, NULL-omitted DigestInfo branch (RSADigestSigner):

```
-  for (int i = 0; i < hash.length; i++)
+  // compare the whole trailing OCTET STRING - tag, length
+  // and every hash byte: hash.length + 2 bytes
+  for (int i = 0; i < hash.length + 2; i++)
   {
       nonEqual ⊨ (sig[sigOffset + i] ^ expected[expectedOffset + i]);
   }
```

- Last two hash bytes never compared — 2^{16} off forgery work
- The sibling JCA class had the correct bound — the model diffed the siblings (fix: ea5970ea9b)

It reopens old wounds, too

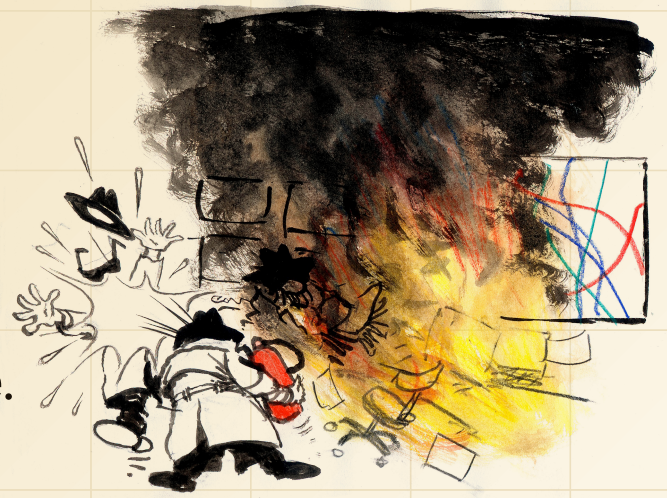
OpenSSL CONFERENCE

- CVE-2018-5382 (BKS 16-bit MAC key): the 2018 fix gated creating v1 stores — the load path was never gated. Completed 8 years later (faf5daa6e9)
- CVE-2023-33201 (LDAP injection): fix reached main + jdk1.4 CertStore — but the jdk1.4 build overwrites the patched file with an unpatched variant. Published bcprov-jdk14 never got the fix (27c468af54)
- "Fixed" is a claim about paths, not files — models are good at checking all the paths

Over the edge

OpenSSL CONFERENCE

- STE (standard technical English) mostly okay.
- API design is not a strong point
- Starting from scratch, in general, not so strong either.
- Case study: Merkle Tree Certificates
  - First pass: no tools, hand-coded, dealt with ASN.1 structures, minor utilities.
  - Second pass: Opus, generated API and tests based on a later draft, used previous work. API took 3 days to refactor.
  - Third pass: Fable, updated API based on later draft, review was effortless.
- Difference between the 3 passes, more context available each time.



Fix, revert, refix: 36 hours of IESEngine

OpenSSL CONFERENCE

- Finding (high): stream-mode IES puts the MAC key at a length-dependent KDF offset → one known plaintext forges any shorter message
- 21 Jun 18:40 — fix v1: refuse static-key stream mode entirely (2ffafa2de9)
- 21 Jun 21:28 — REVERT: static-static stream ECIES/IES is a supported mode with real users (6fb4898a3d)
- 22 Jun 11:23 — fix v2: derive the MAC key from a fixed KDF prefix — closes the forgery, keeps the mode (6d9e4bbaee)
- The model found the bug. It could not know our compatibility contract.

Case study: the diff read backwards

OpenSSL CONFERENCE

- Reconciling LTS against upstream (ESTService, Jul 2026): the model read the diff direction backwards
- Its recommendation: keep LTS's unbounded drain — i.e. preserve a thread-hang defect and reject upstream's fix
- Delivered fluently: a rationale, a trade-off analysis, a confident “leave it”
- What caught it: not “are you sure?” — a specific technical question: “can you suggest a better solution for the narrow case?”
- Same session: the proposed fix contained dead code (available() ≡ 0 on the wrapper); the first regression test passed against the broken code
- Of four significant errors: zero caught by the model reviewing its own output

How to tell the difference

OpenSSL CONFERENCE

- Trust more when:
  - mechanical + cross-referencing: test registration, sibling-impl diffs, bounded-allocation sweeps
  - a failing test proves the bug; you're hunting instances of a known class
- Trust less when:
  - the fix touches a supported-use contract; “correct” depends on who your users are
  - the green tick came from a build you haven't audited; severity / by-design judgments
- Scrutinise hardest: “no change needed — we're ahead” — cheapest to conclude, costliest to be wrong about, leaves no artifact to review
- Probe with specific, forward-looking questions — “are you sure?” gets the same answer, more confidently
- Universal rule: every claim gets a test that fails on the old code — run it against the unfixed state

Our working rules today

OpenSSL CONFERENCE

- 1. Docs first: write more down, if the tool gets it wrong, fix the docs, then the code
- 2. Skills for anything we'll do twice
- 3. Tests must run and must be able to fail — audited, not assumed
- 4. Every AI-assisted change is human-reviewed; provenance is recorded
- 5. Documented AI Policy in CONTRIBUTING.md
- 6. Developers are responsible for what the AI produces and own it
- 7. Try not to include failures in anything we'll do twice!



What this means beyond Bouncy Castle

OpenSSL CONFERENCE

- Mythos-scale CVD is coming to every significant library — ready or not
- Maintainers: expect duplicate AI reports; have a triage story; write the reachability facts down before you need them
- Dependents: the long tail of dormant bugs is getting shorter — on both sides of the fence. Update cadence matters more
- The gap that matters now isn't finding bugs, or even maintainers — it's the capacity to judge and ship fixes
- The tools are enormously useful, but everyone having access to them means there's no room to rest.

Questions?

OpenSSL CONFERENCE

