Agentic Engineering at Regulated Scale: Building a Post-Quantum, FHE-Native Securities Platform

Zach Kelling — Hanzo Team

Version 1.0 2026

Abstract

We document the engineering capability behind Hanzo's stack by way of its hardest production proof: an agent-driven team built a fully regulated securities venue—an SEC-registered alternative trading system (ATS) with broker-dealer and transfer-agent functions—on a next-generation foundation of production post-quantum cryptography, a high-performance native-GPU blockchain, and fully-homomorphic-encryption (FHE) privacy. The build *consolidated* rather than sprawled—200+ microservices collapsed into 3 compiled binaries, with 3.07M lines of code reduced to 1.06M active (∼3.5M lines of dead code deleted)—reaching production in record time with a small team, and passed FINRA/SEC approval and SOC 2 compliance. This is not template web software; it is among the most compliance-heavy and cryptographically advanced software domains that exists. We describe the throughput of the agentic engineering method that produced it, and we expand on the broad applicability of its FHE layer—computing on data one is not permitted to see—across regulated industries and government, defense included.

Executive Summary. The fastest way to judge an engineering organization is what it has shipped under real constraints. Hanzo's founding CTO, Zach Kelling, was ranked the **#1 user of Anthropic by token count worldwide in 2025** (per public usage tracking at `claudecount.com`)—a marker of an engineer operating agentic AI at the frontier of throughput. That throughput is not spent on brochure sites: the flagship production system built with Hanzo's agentic stack is a **fully regulated securities exchange**—an SEC-registered ATS with broker-dealer and transfer-agent functions—running on **post-quantum cryptography**, a **native-GPU high-performance blockchain**, and **fully homomorphic encryption** for privacy. The build *consolidated* rather than sprawled—200+ microservices collapsed into 3 compiled binaries, 3.07M lines of code cut to 1.06M active—shipped fast with a small team, and cleared FINRA/SEC approval and SOC 2. Regulated financial infrastructure is one of the hardest, most audited software domains there is; clearing it on a cryptographically advanced foundation is the proof that agent-driven engineering is production-grade, not a demo—and that it is exactly the capability needed to modernize the software stacks of regulated enterprises and government alike, defense among them.

Contents

1 The Engineer and the Throughput

Agentic engineering multiplies a strong engineer; it does not replace one. Hanzo's founding CTO works at the frontier of both AI/ML and cryptography—the two disciplines this stack fuses—and operates agentic AI at a volume that placed him, by public token-count tracking (`claudecount.com`), as the top Anthropic user in the world in 2025. The relevant point is not the ranking for its own sake; it is what that volume was spent producing. Raw AI usage that yields disposable prototypes proves nothing. Raw AI usage that yields a regulated, audited, cryptographically advanced production system proves the method works where it is hardest.

2 What Was Built

The flagship system is a regulated securities venue, not a content site. In U.S. securities terms it spans three regulated functions that ordinarily take separate firms years to stand up:

- **Alternative Trading System (ATS)** — an SEC-registered trading venue (Regulation ATS), matching and executing orders.
- **Broker-Dealer (BD)** — FINRA-regulated, with the supervisory, recordkeeping, and capital obligations that entails.
- **Transfer Agent (TA)** — SEC-registered, maintaining the authoritative record of securities ownership and transfers.

These functions are governed by some of the most exacting compliance regimes in software. The notable engineering fact is that the build *consolidated* rather than sprawled: a legacy estate of **200+ microservices** on a public cloud was collapsed into **3 compiled binaries**, and the codebase went from **3.07M lines to 1.06M active** (∼70% smaller, with ∼3.5M lines of dead code deleted; the trading/auth core itself shrank from ∼264,000 to ∼44,000 lines, with 1.3 GB of vendored dependencies eliminated and a single 48 MB binary subsuming 17+ formerly separate services). It reached production **in record time**, with a **small team**, and was carried through **FINRA/SEC approval** and **SOC 2** compliance—leaner than the system it replaced, not larger.

3 One Engineer, a Team's Output

The result that matters most is the ratio of output to headcount. The agentic method let a *single* AI-augmented engineer carry the bulk of a regulated production platform that a conventional organization staffs as a full team—and the platform's own engineering audit documents it. Across an **81-day** window spanning **334 repositories** and **31,058 commits**, the audit attributed **15,294 commits** and **114M raw git-log lines**—framework-wide refactors, dependency upgrades, automation, and production code—to the agentic CTO workflow, measured at **$0.0006 per line** by the audit's methodology. Within that, a focused **10-day remediation sprint** produced **3,694 commits**, removed **∼3.5 million lines of dead code** across 181 repositories, and authored **100% of the critical- and high-severity security fixes**—single-handedly—while consolidating 200+ microservices into three binaries. The security and compliance gaps were closed on the path to approval (a source audit of **93 findings**, 22 critical, plus **780+ dependency vulnerabilities** remediated to zero actionable), through FINRA/SEC approval and SOC 2.

The economic consequence is direct and measured. When one engineer, multiplied by the agentic stack, produces what a team produces, the **cost per line of shipped, audited, production code**

collapses—here, **$0.0006 per line** by the audit's own methodology, orders of magnitude below conventional team rates. For any regulated enterprise or government program (defense included) the implication is the whole argument: the same regulated, post-quantum, FHE-native system, delivered by a fraction of the headcount at a fraction of the cost—and owned, not rented. This is what "agentic engineering" means in practice—not a chatbot writing snippets, but one expert operating at the throughput of a department.

4 The Foundation: Post-Quantum, GPU Blockchain, FHE

What makes the build a genuine engineering flex is the foundation it runs on— each element of which is independently hard:

- **Production post-quantum cryptography.** All three NIST PQC standards (ML-KEM/203, ML-DSA/204, SLH-DSA/205) in a live system with hybrid classical/quantum security—so the chain of custody for securities records is protected against a future quantum adversary.
- **Native-GPU high-performance blockchain.** A chain serving as the authoritative, immutable, post-quantum-anchored record of transfers, with GPU-accelerated cryptographic operations for throughput.
- **Fully homomorphic encryption (FHE).** GPU-accelerated FHE (BFV/BGV/CKKS/TFHE) with threshold decryption, so privacy-sensitive computation runs *on ciphertext*—no single component ever holds both the plaintext and the result.

A team that can compose post-quantum cryptography, a GPU blockchain, and FHE into a system that clears securities regulators is a team that can modernize a high-assurance software stack—in healthcare, finance, critical infrastructure, or government, defense included—to the same foundation.

5 FHE for High-Stakes Computation: Computing on Data You Cannot See

Fully homomorphic encryption is the capability with the broadest reach across high-stakes domains, so it is worth drawing out. FHE allows arbitrary computation directly on encrypted data: the operator performs the computation and returns an encrypted result without ever decrypting the inputs, and only the data owner (or a threshold of keyholders) can decrypt the output. The system doing the work never sees the plaintext. In the securities platform this protects order and ownership data; the same primitive answers a range of problems—in regulated industry, multi-party commerce, and government (defense included)—that have no clean classical solution:

- **Private healthcare and population analytics.** Hospitals and researchers can compute statistics, risk scores, or model inferences over patient records while the records themselves stay encrypted—enabling collaboration on protected health information without exposing it.
- **Multi-party financial computation.** Banks, exchanges, and counterparties can compute a joint result—netting, exposure, fraud signals—over their combined data while each party's books stay encrypted and private, so no participant has to surrender its raw positions to get the answer.

- **Confidential supply-chain analytics.** Suppliers and buyers can reconcile inventory, demand, or provenance across organizations without any party revealing its underlying commercial data.
- **Computation on untrusted infrastructure.** Sensitive workloads can run on shared or third-party cloud infrastructure without ever exposing the data to the host—widening where a regulated workload may legally run.
- **Encrypted ML inference.** A model can classify encrypted images, documents, or signals: the data owner never exposes the input, and the model operator never sees it—useful when the data is sensitive, the model is sensitive, or both.
- **Cross-organization data fusion without disclosure.** Multiple parties can correlate multi-source data while each source stays encrypted, so a breach of the fusion system does not expose the underlying inputs. In a defense setting this is the cross-domain and multi-party (coalition) case—a guard evaluates a release policy against encrypted content, or allies compute a joint result, without ever seeing the data (developed in the cross-domain paper).

FHE's historical objection is speed; the practical answer in this stack is GPU acceleration of the underlying number-theoretic transforms, plus threshold decryption so no single party holds the key. "Compute on data you are not allowed to see" is a capability high-stakes domains have wanted for decades—defense among them; here it is a production component, not a research slide.

6 Why Open Source Was the Accelerant

The obvious question is how a small team shipped a regulated, cryptographically advanced system of this scale so quickly. The answer is open source—specifically, an *owned* open-source stack composed by an agentic engineering method.

Hanzo did not assemble the platform from a patchwork of third-party SaaS. It composed it from coherent open-source building blocks that Hanzo itself authors and controls—identity and access management, key management, the data layer, the gateway and ingress, the ZAP transport, the post-quantum and FHE cryptography, the operator, and the Zen models—each reusable, inspectable, and owned. The agentic stack is the multiplier; the owned OSS foundation is the leverage. You do not rebuild identity, key management, or a gateway for each new system; you compose owned, hardened components and point the agents at the application on top. That is how a regulated platform of this scope reaches production fast—and lean.

For any high-assurance buyer, open source inverts the usual security argument. A closed system asks you to trust a vendor you cannot inspect; an open one lets you read every line, verify it—here, with machine-checked proofs—confirm there are no hidden backdoors, and run it on hardware you control. Auditable beats opaque whenever the stakes are high. Open source is therefore not a cost-cutting compromise: it is simultaneously the source of **velocity** (compose, do not rebuild), of **security** (inspectable and formally verifiable), and of **sovereignty** (own it, modify it, run it anywhere, including air-gapped). The three properties a regulated or government program most wants—defense included—are the same three that made the build fast.

Stated plainly as a hook: *Hanzo builds massive, secure systems quickly because it builds on an open-source stack it owns—so the speed, the security, and the sovereignty all come from the same source.*

7 Why It Matters for Regulated and Government Modernization

The throughput and the difficulty compound. The same agentic engineering method that shipped a regulated, post-quantum, FHE-native securities exchange fast and small is the method Hanzo brings to any regulated-enterprise or government modernization—healthcare, finance, critical infrastructure, and defense alike: take an existing, audited (and, for government systems, already-accredited) stack and modernize it in place—to a sovereign, post-quantum, self-hostable footprint with FHE-grade privacy—faster and cheaper than a traditional rebuild, and owned by the operator. The securities venue is the existence proof that the method holds up under the most demanding regulatory and cryptographic constraints production software offers; every other high-assurance modernization—defense among them—is the same engine pointed at a new mission.