Confidential Computing by Default: An FHE-Native Chain and Agent Runtime

Zach Kelling — Hanzo Team

Version 1.0 2026

Abstract

Every system that processes sensitive data has a moment where it holds the data in the clear—and that moment is the insider risk, the cross-domain leak, and the blast radius of every breach. Fully homomorphic encryption (FHE) removes that moment: computation runs directly on ciphertext, and only a threshold of keyholders can decrypt the result. This paper describes Hanzo's design for making FHE the *default* mode of computation across the two layers that normally demand plaintext—a blockchain's state and rules, and an autonomous agent's actions—so that state stays encrypted on-chain, policies are evaluated over ciphertext, and agents act on data they are never able to read. We give the cryptographic basis (BFV/BGV/CKKS/TFHE with threshold decryption), the GPU acceleration that makes it practical, the post-quantum anchoring that protects it for the long term, the applications it unlocks across regulated industry, multi-party commerce, and government (defense included), and an explicit, honest assessment of which components are production today versus an active research frontier.

Executive Summary. Normally, to compute on data you must first decrypt it—so somewhere in every system there is a component holding the secret in the clear, and that component is what an adversary, an insider, or a subpoena goes after. FHE lets the computer do the work *while the data stays encrypted*: it takes in ciphertext, produces a ciphertext answer, and never sees the plaintext. Hanzo pushes this idea through the whole stack. The blockchain stores encrypted state and can enforce rules on encrypted data; agents can act on encrypted data and return encrypted answers; and only a quorum of keyholders can ever decrypt—no single party holds the key. This is directly useful wherever the stakes are high: hospitals can compute over patient records without exposing them, financial counterparties can compute a joint result without sharing their books, and sensitive workloads can run on infrastructure you do not fully control without exposing the data to the host. The same shape serves government and defense—allies can compute a joint answer without sharing raw data, and a cross-domain guard can decide whether something is releasable without seeing its content. FHE's historical objection is speed; the practical answer here is GPU acceleration plus running only the sensitive part of a computation under encryption. The result is a stack where confidentiality is the default state of data in use, not just data at rest or in transit.

Contents

1 The Plaintext Gap

We protect data in two of its three states well and the third state badly. Data *at rest* is encrypted on disk; data *in transit* is encrypted on the wire. But data *in use*—the moment a program actually computes on it—is almost always plaintext in memory. Every database query that filters records, every policy engine that decides access, every model that runs inference, every smart contract that checks a balance, does so on cleartext. That cleartext moment is the structural weakness behind a large share of real incidents: the compromised process, the malicious insider, the memory-scraping implant, the cross-domain guard that has to read the data to rule on it, the cloud host that can see its tenants' workloads.

Confidential computing has tried to shrink this gap with hardware enclaves (trusted execution environments). Enclaves help, but they relocate trust rather than remove it: you must trust the silicon vendor, the attestation chain, and the absence of side channels, and the data is still plaintext *inside* the enclave. Fully homomorphic encryption attacks the gap directly. With FHE the data is never decrypted to be computed on at all—not in the enclave, not in memory, nowhere. The computation is performed on ciphertext and yields a ciphertext result. The plaintext gap closes.

2 FHE in One Page

Fully homomorphic encryption is an encryption scheme with an extra property: there exist operations on ciphertexts that correspond to operations on the underlying plaintexts. Concretely, given encryptions of a and b, anyone—without the key—can compute an encryption of $a + b$ and of $a \times b$. Addition and multiplication are a complete basis, so in principle any computable function can be evaluated homomorphically by expressing it as an arithmetic (or boolean) circuit. The party doing the computation learns nothing; only a holder of the secret key can decrypt the output.

The scheme families and what each is for.

- **BFV / BGV** — exact integer arithmetic. Right for counts, sums, comparisons, exact matching, and policy predicates over integers.
- **CKKS** — approximate arithmetic over real/complex numbers, with SIMD "batching" that packs thousands of values into one ciphertext. Right for statistics, signal processing, and machine-learning inference.
- **TFHE** — fast boolean gates and programmable bootstrapping. Right for branching, comparisons, and arbitrary logic, including data-dependent control flow.

Bootstrapping. Each homomorphic operation adds noise; after enough operations the noise would corrupt the result. *Bootstrapping* refreshes a ciphertext to a low-noise state and is what makes encryption "fully" homomorphic—it permits unbounded computation depth. Bootstrapping is the expensive step, and it is the principal target of the acceleration discussed in §7.

Threshold decryption. The secret key need not exist in one place. With *threshold* FHE the key is split across n parties so that any t of them can jointly decrypt but no coalition smaller than t can. Combined with FHE this gives a system in which *no single party ever holds both the data and the ability to reveal it*—the computation runs blind, and revealing the answer requires a quorum. This property is what makes the design defensible for multi-party and adversarial settings.

3 Confidential State: An FHE-Native Chain

A blockchain is normally the *opposite* of confidential: every validator re-executes every transaction against shared plaintext state, which is precisely why public chains leak everything. An FHE-native chain inverts this. State is stored as ciphertext; transactions carry encrypted inputs; and the state-transition rules are evaluated homomorphically, so validators advance the ledger *without ever seeing the values they are agreeing on.*

- **Encrypted state.** Balances, ownership records, order books, classifications—whatever the application holds—live on-chain as ciphertext. The authoritative record exists, is immutable and auditable, and is unreadable to anyone without a decryption quorum.
- **Rules on ciphertext.** A transfer rule such as "permit only if the sender's balance covers the amount" is a comparison; under BFV/TFHE that comparison is evaluated on the encrypted balance and encrypted amount, yielding an encrypted yes/no that gates the state update. The rule is enforced without the rule-evaluator learning the balance.
- **Threshold-gated disclosure.** When a value must be revealed—a settlement, an audit, a court order, a cross-organization release decision—a threshold of keyholders cooperates to decrypt exactly that value, and the decryption itself is recorded. Disclosure becomes an explicit, quorum-authorized, logged event rather than a default.
- **Post-quantum anchoring.** The chain's signatures and transport are post-quantum (ML-KEM/ML-DSA), so the encrypted record and its provenance survive a future quantum adversary—essential because ciphertext recorded immutably today is exactly the "harvest now, decrypt later" target (see the full-spectrum cyber and ZAP papers).

The chain thus provides a tamper-evident, post-quantum, immutable audit trail *of encrypted facts*: you can prove what happened and in what order without ever exposing what the facts were, until a quorum decides otherwise.

4 Confidential Action: FHE-Native Agents

The second layer that normally demands plaintext is automation. An autonomous agent that triages intelligence, screens transactions, routes logistics, or enforces a policy has to "see" the data to act on it—and so the agent runtime becomes another plaintext concentration point, now with the additional risk that the model itself might memorize or exfiltrate. FHE breaks this too.

An FHE-native agent operates on encrypted inputs and emits encrypted outputs. It can evaluate the encrypted state of the chain, apply encrypted rules, and produce an encrypted decision or an encrypted action—all without the agent operator (or the model host) ever seeing the plaintext. Where the agent must run a learned model rather than a fixed rule, CKKS-based encrypted inference lets a (suitably structured) model classify or score encrypted inputs and return an encrypted result. The agent is then *operator-blind*: it does useful work on data it cannot read, on infrastructure that cannot read it either.

This composes with Hanzo's broader agentic runtime (the agentic-AI and edge papers): the same orchestration of specialized agents and tool calls runs, but the data plane underneath it is ciphertext. Autonomy and confidentiality stop being a trade-off.

5 Programmable Confidentiality: Rules and Policies on Ciphertext

The unifying idea is that a *rule* is just a function, and any function can in principle be evaluated under FHE. So the policies that govern a system—not only its data—can run on ciphertext. This is the capability the rest of the paper builds on, because most high-consequence decisions are policy decisions over sensitive inputs:

- **Releasability** — "may this datum cross to that domain / that external partner?" decided on the encrypted datum (a cross-domain or coalition guard is the defense instance; see cross-domain paper).
- **Eligibility / authorization** — "does this party satisfy the conditions?" decided without revealing the party's attributes.
- **Thresholds and limits** — "is the aggregate over/under the line?" decided without revealing the individual contributions.
- **Matching** — "do these two sealed values correspond?" (sealed-bid auctions, private set intersection, deconfliction) decided without opening either side.

In each case the *decision* is computed and enforced while the *inputs* remain encrypted, and only the (small, controlled) decision output is ever a candidate for threshold decryption. Confidentiality becomes programmable: you write the rule once and it runs blind.

6 High-Stakes Applications

The combination of an encrypted ledger and operator-blind agents answers a set of problems—across regulated industry, multi-party commerce, and government (defense included)—that have no clean classical solution:

- **Private healthcare and population analytics.** Hospitals and researchers compute statistics, risk scores, or model inferences over patient records while the records stay encrypted, so collaboration on protected health information never requires exposing it.
- **Multi-party computation without disclosure.** Independent parties compute a joint result—netting and exposure between financial counterparties, overlapping indicators between organizations, combined logistics between partners—over their combined data while each party's raw data stays encrypted. Nobody surrenders sources or proprietary data to get the joint answer, and the chain records that the computation occurred. (In a defense setting this is coalition analytics among allies.)
- **Releasability on ciphertext.** A guard evaluates a release policy against encrypted content and decides whether it may move between domains without ever seeing the content, removing the guard itself as a data tap—the pattern behind both commercial data-sharing controls and a defense cross-domain guard.
- **Private data fusion.** Multi-source correlation runs while the sources stay encrypted, so a breach of the fusion system does not expose the underlying inputs—whether those are commercial datasets or intelligence collection.

- **Encrypted ML inference.** A model classifies 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.
- **Encrypted operational state.** Records, work lists, and movement tables—an order book, a supply chain, or, in a defense setting, command, targeting, and logistics state—live as encrypted on-chain state with an immutable, post-quantum audit trail; agents act on them under encryption; reveal is quorum-gated and logged.
- **Computation on untrusted infrastructure.** Sensitive workloads run on third-party, shared, or commercial-cloud hardware without exposing the data to the host—widening where a regulated workload can legally compute.

7 Why the Speed Objection No Longer Holds

FHE's reputation for impracticality is a decade out of date, and three things close the gap in this stack:

1. **GPU acceleration.** The dominant cost in FHE is the number-theoretic transform (NTT) at the heart of polynomial multiplication and bootstrapping. These are massively parallel and map naturally onto GPUs; the same native-GPU foundation that accelerates the chain's cryptography accelerates the FHE layer, turning bootstrapping from seconds toward the practical range.
2. **Batching.** CKKS and BGV pack thousands of plaintext slots into one ciphertext, so a single homomorphic operation does thousands of useful operations at once—ideal for the vector and tensor workloads in analytics and inference.
3. **Selective FHE.** Most computations have a small sensitive core inside a large non-sensitive shell. You do not run the whole program under FHE—only the predicate or the field that must stay private. The expensive primitive is spent precisely where it buys confidentiality and nowhere else.

The honest framing is not "FHE is now as fast as plaintext"—it is not—but "the workloads that matter for confidential rules, matching, gating, and bounded inference are now practical, and GPU acceleration keeps widening the set."

8 Key Custody Without a Single Point of Trust

A confidentiality system is only as strong as its key handling. The design holds the decryption key nowhere: threshold decryption splits it t-of-n across independent parties (validators, regulators, partner organizations, or coalition members) so that decrypting any value requires an explicit quorum and no minority—and no single compromised host, insider, or seized server—can reveal anything. This is the structural answer to "who can see the data?": by construction, *nobody*, until enough authorized parties agree (validators, agencies, partner organizations, or coalition members), and that agreement is itself recorded on the chain.

9 Composition With the Rest of the Stack

FHE here is not a bolt-on library; it is one primitive in a stack Hanzo owns end-to-end, which is what makes the "native" claim real:

- **Native-GPU chain** provides the encrypted, immutable, post-quantum ledger and the GPU substrate the FHE layer rides on.
- **Post-quantum transport and identity** (ZAP, ML-KEM/ML-DSA) protect the ciphertext in motion and bind it to verifiable identities.
- **Agentic runtime** supplies the operator-blind automation that acts on the encrypted state.
- **Owned models** (the Zen family) supply the inference that, under CKKS, can run on encrypted inputs.

Because all four are owned and open, the confidentiality guarantee is auditable rather than asserted: any high-assurance customer—a regulated enterprise, a government agency, or a defense program—can inspect the whole path the data takes and confirm there is no point where it is silently in the clear.

10 Honest Maturity Assessment

This is a capability paper, and the credible version of it is explicit about what runs today versus what is an active frontier. No overstatement:

- **Production-grade today:** the GPU-accelerated FHE primitives (BFV/BGV/CKKS/TFHE), threshold decryption, the native-GPU post-quantum chain, and the agentic runtime each exist and run. Bounded-depth confidential workloads —rule and policy predicates, comparisons, matching/deconfliction, aggregate thresholds, and inference on suitably structured models—are practical now.
- **Active frontier:** fully general, arbitrary-depth FHE-native programmability at plaintext-class throughput is not solved—by anyone—and we do not claim it. The performance envelope is widening (faster bootstrapping, better GPU kernels, hybrid MPC/FHE designs), and the architecture is built so that workloads move from "frontier" to "practical" as that envelope grows, without redesign.

The strategic point stands regardless of where the line currently sits: Hanzo owns every layer the line runs through, so as FHE performance improves the gains land in a stack a customer already controls—rather than waiting on a closed vendor to expose the capability.

11 Conclusion

Confidentiality of data *in use* is the unsolved third of the problem, and it is the one that matters most for multi-party computation, private analytics, cross-organization data sharing, and computing on infrastructure you do not own—and, in government and defense, for coalition operations, cross-domain movement, and intelligence fusion. FHE closes the plaintext gap by computing on ciphertext; threshold decryption removes the single point of trust; GPU acceleration and selective application make the important workloads practical; and post-quantum anchoring protects the encrypted record for the long term. Hanzo's contribution is to make this the default across a chain and an agent runtime it owns end-to-end, so that "private ciphertext for every operation" is an architectural property of the stack rather than a feature of one application. Compute on data you are not allowed to see—and let the rules, the ledger, and the agents all run blind.