Local-First Regulated Software: Private by Default, Compliant by Design

Zach Kelling
Liquidity Securities LLC
z@lux.network

April 2026

Abstract

Kleppmann et al. (2019) demonstrated that local-first software—where data lives on the user's device and collaboration happens via CRDTs—can give users both real-time collaboration and true data ownership. Seven years later, we extend this thesis to *regulated* industries: financial services, healthcare, and legal, where privacy and compliance are not optional features but legal requirements.

We show that it is possible to build software where: (1) operators cannot see user data, (2) computations run on encrypted data via fully homomorphic encryption (FHE), (3) regulators can force-decrypt only through a threshold approval process (MPC), and (4) an immutable audit trail records every access. We demonstrate this architecture in production at Liquidity.io, a regulated securities exchange (ATS/BD/TA) where the blockchain is the source of truth, SQLite is the local cache, and every byte of user data is encrypted with keys the operator never holds.

1 Introduction

The original local-first paper identified seven ideals: fast, multi-device, offline, collaborative, long-lived, private, and user-controlled. These ideals assumed a consumer context—notes apps, drawing tools, Kanban boards. In regulated industries, two additional requirements dominate:

1. **Compliance**: regulators must be able to audit and, when legally compelled, access specific data (SEC Rule 17a-4, FINRA books and records, GDPR right of access).

2. **Non-custodial privacy**: the platform operator must *not* be able to access user data during normal operations, even though they host the infrastructure.

These requirements appear contradictory: how can an operator host data they cannot read, while simultaneously guaranteeing that a regulator *can* read it when legally required? We resolve this contradiction with a three-layer architecture combining local-first principles, fully homomorphic encryption, and threshold decryption via multi-party computation.

2 Architecture

2.1 Three Layers

1. **Data Plane** (local-first): Each user's data is stored in an encrypted SQLite database on their device (or in an encrypted per-org SQLite on the server). The encryption key is derived from the user's passkey via HKDF. The operator stores only ciphertext.

2. **Compute Plane** (FHE): Operations that need to touch multiple users' data—compliance checks, portfolio analytics, order matching—run on encrypted data using TFHE. The server performs useful computation without ever decrypting.

3. **Trust Plane** (MPC + chain): The FHE secret key is Shamir-split across N MPC nodes. Decryption requires t-of-N key holders to approve. Approvals, denials, and decrypted results are recorded on-chain for immutable audit.

2.2 Normal Operation: Nobody Can Decrypt

During normal operation:

- Users encrypt data client-side with the FHE public key.
- The server stores FHE ciphertext in SQLite.
- Queries return encrypted results (metadata is visible, values are not).
- Homomorphic operations (comparison, addition, compliance checks) run on ciphertext without decryption.
- No single party—not the operator, not any individual MPC node, not even a rogue employee—can access plaintext.

2.3 Regulatory Access: Threshold Decrypt

When a regulator issues a lawful subpoena:

1. The regulator submits a decryption request to the platform, specifying the legal basis and target records.
2. The request is published to all N MPC key holders (which may include the operator, independent trustees, the regulator's own node, and a hardware security module).
3. Each key holder independently reviews the request and, if satisfied, submits their Shamir share.
4. When t-of-N shares are collected, the FHE secret key is reconstructed *in memory only*, the target records are decrypted, and the key is immediately zeroed.
5. The decrypted data is returned to the requester.
6. The entire process—request, approvals, decryption, and result hash—is recorded on-chain in the WORM audit trail.

This ensures:

- No single party can unilaterally access data.
- Every access is auditable and attributable.
- The operator can prove to regulators that they *cannot* access data without the threshold process (a strong defense against "insider threat" liability).

3 Implementation

We implement this architecture at Liquidity.io using:

- **Hanzo Base**: Go framework with embedded SQLite, admin UI, auth, and realtime. Each org gets an encrypted SQLite file.
- **Lux FHE**: Pure Go TFHE library supporting bit-level encryption, boolean gates, integer arithmetic, and threshold key sharing (LSSS).
- **Lux MPC**: Go daemon implementing CGGMP21 (ECDSA) and FROST (EdDSA) threshold signing, with HSM co-signing attestation.
- **Liquid EVM**: Sovereign L1 blockchain with 19 precompiles including FHE verify, ML-KEM, ML-DSA, FROST, and CGGMP21.
- **WORM Audit Trail**: Hash-chained, append-only records archived to GCS Coldline ($0.004/GB/month) for SEC 17a-4 compliance.

3.1 Regulated Vault Demo

We provide a complete demo application (`regulated-vault`) that implements this architecture as a single Go binary:

```
cd ~/work/lux/fhe/cmd/demos/regulated-vault
go run . serve --http=0.0.0.0:8090
```

The demo exposes seven endpoints:

- `POST /v1/vault/store` — encrypt and store (operator sees only ciphertext)
- `GET /v1/vault/list` — list records (encrypted, no plaintext)
- `POST /v1/vault/compute` — homomorphic computation
- `POST /v1/vault/request` — request decryption (starts approval process)
- `POST /v1/vault/approve/{id}` — MPC key holder approves
- `GET /v1/vault/reveal/{id}` — get plaintext (only after t-of-N approvals)
- `GET /v1/vault/audit` — immutable audit trail

4 Comparison with Kleppmann (2019)

5 Security Properties

1. **Operator-blind**: The operator stores only FHE ciphertext. Even a compromised operator cannot access plaintext without t-of-N MPC shares.
2. **Post-quantum**: ML-KEM (FIPS 203) for key encapsulation, ML-DSA (FIPS 204) for signatures, FROST for threshold EdDSA. All activated at genesis on the Liquid EVM.
3. **Non-custodial**: Users hold assets in MPC wallets (2-of-3 threshold). The platform cannot unilaterally move assets.
4. **Tamper-evident**: Hash-chained audit trail. Any modification breaks the chain and is detectable via `VerifyChain()`.
5. **Archival**: Records exported to GCS Coldline as compressed NDJSON, organized by `org/service/year/month/day`.

Ideal	Kleppmann (2019)	This Work
Fast (local-first)	SQLite + CRDTs	SQLite + Base
Multi-device	CRDT sync	Per-org encrypted SQLite
Offline	Full	Full
Collaboration	CRDTs	CRDTs + FHE
Longevity	File formats	SQLite + chain
Privacy	E2E encryption	FHE (operator-blind)
User control	Full	Full
Compliance	Not addressed	MPC threshold decrypt
Audit trail	Not addressed	WORM hash-chain
Regulatory access	Not addressed	t-of-N forced decrypt

Table 1: Extending the seven ideals of local-first software to regulated industries.

6 Use Cases

We demonstrate eight concrete applications of this architecture:

1. **Encrypted Dark Pool**: Orders submitted as FHE ciphertext. Matching engine runs homomorphically. Only matched orders are threshold-decrypted.
2. **Private Compliance**: SEC diversification rules checked on encrypted portfolios. Boolean result (compliant/non-compliant) revealed without exposing holdings.
3. **Sealed-Bid Auction**: Bids encrypted. Winner determined via FHE comparison. Only winning bid decrypted.
4. **Shareholder Voting**: Encrypted votes tallied homomorphically. Final count revealed without individual vote exposure.
5. **Confidential NAV**: Fund holdings encrypted. NAV computed homomorphically. Individual positions never exposed.
6. **Private Market Making**: Market makers submit encrypted quotes. Best bid/ask selected homomorphically.
7. **Compliance Proofs**: Prove regulatory compliance without revealing the underlying data.
8. **Regulated Vault**: General-purpose encrypted storage with MPC threshold decrypt for lawful access.

Each use case is implemented as a single Go binary using Hanzo Base, with embedded admin UI, encrypted SQLite, and FHE computation.

7 Conclusion

Kleppmann showed that local-first software can give users both collaboration and ownership. We show that the same principles extend to regulated industries—where privacy is a legal requirement, not just a feature—by adding FHE for operator-blind computation and MPC for threshold-controlled regulatory access.

The key insight is that *trust is not binary*. Between "the operator can see everything" and "nobody can see anything" lies a spectrum of threshold policies that match real-world governance:

boards require quorum, courts require warrants, auditors require evidence. Our architecture encodes these social processes in cryptographic protocols.

The software exists, compiles, and runs. The regulated vault demo is a single `go run . serve` away from production. We invite regulators, compliance officers, and privacy advocates to examine the code, run the demos, and verify the claims.

Trust is not a promise. Trust is a threshold.