FHE + MPC Hybrid Systems for Trustless Private Computation

Zach Kelling

Lux Industries

2024

Abstract

We present a hybrid cryptographic architecture combining Fully Homomorphic Encryption (FHE) and Multi-Party Computation (MPC) to achieve trustless private computation. FHE alone computes on encrypted data but concentrates trust in whoever holds the decryption key. MPC alone distributes trust but requires data sharing among participants. The hybrid approach eliminates both weaknesses: FHE handles encrypted computation while MPC distributes the decryption key via threshold secret sharing, ensuring no single party can decrypt intermediate results. We formalize the construction using TFHE for boolean circuit evaluation and FROST-based threshold decryption for key distribution. We demonstrate the architecture on a private orderbook matching engine where N market participants submit encrypted orders, the matching engine evaluates the order book homomorphically, and a t-of-n MPC committee decrypts only the matched trades. No party—including the matcher—ever observes the full order book in cleartext. The implementation runs on LiquidFHE (Lux F-Chain) with M-Chain MPC providing threshold decryption, achieving 47ms per encrypted order match at NIST Level 1 security.

1 Introduction

Private computation on blockchain networks faces a fundamental dilemma. Two mature cryptographic techniques exist—Fully Homomorphic Encryption and Multi-Party Computation—but each has a critical limitation when deployed in isolation.

FHE alone. TFHE [1] enables arbitrary boolean circuit evaluation on encrypted data. The computational model is clean: encrypt inputs, evaluate the circuit homomorphically, decrypt the output. The problem is the decryption key. Whoever holds it can decrypt *any* intermediate ciphertext, not just the final output. In a blockchain context, this means a single operator (or a compromised key) can observe all encrypted state.

MPC alone. Protocols such as SPDZ [2] and GMW distribute computation across N parties such that no coalition smaller than a threshold t can learn any party's private input. The problem is that inputs must be secret-shared among the participants. For N parties each holding private data, this requires $O(N^2)$ communication per operation, and every party must remain online for the entire computation.

The hybrid. FHE handles computation. MPC handles decryption. The FHE secret key is split via Shamir secret sharing across n MPC nodes. Computation proceeds homomorphically without any interaction. Only when a result must be revealed do t-of-n MPC nodes participate in threshold decryption.

Property	**FHE only**	**MPC only**	**FHE+MPC**
No single point of trust	×	✓	✓
No data sharing required	✓	×	✓
Non-interactive compute	✓	×	✓
Threshold decryption	×	N/A	✓
Offline tolerance	✓	×	✓*

*Parties must be online only during threshold decryption, not during computation.

2 Threat Model

We consider an adversary who may corrupt up to $t-1$ of n MPC decryption nodes and has full read access to the FHE computation environment (e.g., the blockchain state). The adversary's goal is to learn private inputs.

Definition 2.1 (Security Goal)**.** *No coalition of fewer than t decryption nodes, even with full access to all FHE ciphertexts and the homomorphic computation transcript, can distinguish any ciphertext from random.*

This holds under the LWE assumption (for TFHE) and the discrete logarithm assumption (for FROST key shares), independently.

3 FHE Background: TFHE

TFHE operates on the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. A single bit $m \in \{0, 1\}$ is encrypted as an LWE sample:

$$\text{ct} = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \quad \text{where} \quad b = \langle \mathbf{a}, \mathbf{s} \rangle + m \cdot \lfloor q/2 \rfloor + e$$

The secret key $\mathbf{s} \in \{0, 1\}^n$ and noise e drawn from a discrete Gaussian with standard deviation σ.

3.1 Gate Evaluation

Boolean gates are evaluated via programmable bootstrapping:

1. **Homomorphic gate**: Compute $\text{ct}_{\text{out}} = f(\text{ct}_1, \text{ct}_2)$ for gate f (AND, OR, XOR, etc.) via lookup table encoded in the bootstrapping key.
2. **Blind rotation**: Rotate a test polynomial by the phase of the input ciphertext using RGSW ciphertexts of the key bits.
3. **Sample extraction**: Extract a fresh LWE ciphertext from the rotated polynomial.

Each bootstrapped gate costs approximately 12ms on a single core (benchmarked on the `luxfi/lattice` library with $n = 1024$, $q \approx 2^{27}$).

3.2 The Key Trust Problem

Standard TFHE decryption recovers the plaintext as:

$$m = \text{round}\left(\frac{b - \langle \mathbf{a}, \mathbf{s} \rangle}{q/2}\right)$$

Whoever holds $\mathbf{s}$ can decrypt any ciphertext—including intermediate computation results, cancelled orders, private balances, and rejected transactions. In a blockchain deployment, this is unacceptable.

4 Threshold Decryption via MPC

4.1 Key Generation

The FHE secret key $\mathbf{s} \in \{0, 1\}^n$ is generated via distributed key generation (DKG). No single party ever holds the full key.

1. Each of n MPC nodes generates a random key share $\mathbf{s}_i \in \{0, 1\}^n$.
2. Shares are combined using Shamir secret sharing over $\mathbb{Z}_q$ such that $\mathbf{s} = \sum_{i=1}^{n} \lambda_i \cdot \mathbf{s}_i$ for Lagrange coefficients λ_i.
3. The public encryption parameters (bootstrapping key, key-switching key) are derived from commitments to $\mathbf{s}$ without revealing it.

4.2 Partial Decryption

To decrypt a ciphertext $(\mathbf{a}, b)$, each of t participating nodes computes a partial decryption:

$$d_i = \langle \mathbf{a}, \mathbf{s}_i \rangle \bmod q$$

The combiner reconstructs:

$$\langle \mathbf{a}, \mathbf{s} \rangle = \sum_{i \in S} \lambda_i \cdot d_i \bmod q$$

and recovers $m = \text{round}\left(\frac{b - \langle \mathbf{a}, \mathbf{s} \rangle}{q/2}\right)$.

Theorem 4.1 (Threshold Security)**.** *If the LWE problem with parameters (n, q, σ) is hard, then no coalition of fewer than t parties can distinguish a TFHE ciphertext from random, even given access to the partial decryption oracle for other ciphertexts.*

Proof sketch. Reduce to standard LWE. Given $t - 1$ key shares, the adversary's view of the remaining share is uniformly random over $\mathbb{Z}_q^n$ (information-theoretic security of Shamir sharing). The ciphertext is therefore an LWE sample with unknown key, which is pseudorandom under LWE. □

5 FROST + FHE Key Shares

We use FROST (Flexible Round-Optimized Schnorr Threshold signatures) [3] for MPC coordination, adapted to manage FHE key shares rather than signing keys.

5.1 Why FROST

FROST provides:

- Two-round threshold signing (commitment + response).
- Robustness: if a participant fails, the protocol completes without them (as long as $\geq t$ participate).
- No trusted dealer: DKG produces shares without any party seeing the full secret.

In our construction, the FROST DKG protocol generates both:

1. The FHE secret key shares $\{\mathbf{s}_i\}_{i=1}^n$ for threshold decryption.
2. A Schnorr signing key for threshold attestation (proving that a legitimate decryption occurred).

5.2 Decryption Protocol

Algorithm 1 Threshold FHE Decryption with FROST Attestation

Require: Ciphertext $(\mathbf{a}, b)$, threshold t, participants $S \subseteq [n]$ with $|S| \geq t$
Ensure: Plaintext m and threshold attestation σ

Round 1 (Commitment):
for all $i \in S$ **do**
  Compute partial decryption $d_i = \langle \mathbf{a}, \mathbf{s}_i \rangle \bmod q$
  Compute commitment $C_i = \text{Hash}(d_i \| \text{nonce}_i)$
  Broadcast C_i
end for
Round 2 (Reveal + Combine):
for all $i \in S$ **do**
  Reveal d_i and nonce_i
  Verify $C_i = \text{Hash}(d_i \| \text{nonce}_i)$
end for
Compute $\langle \mathbf{a}, \mathbf{s} \rangle = \sum_{i \in S} \lambda_i \cdot d_i \bmod q$
Recover $m = \text{round}\left(\frac{b - \langle \mathbf{a}, \mathbf{s} \rangle}{q/2}\right)$
Produce FROST threshold signature σ over $(m, \text{ct\_hash})$
return (m, σ)

The FROST attestation σ proves to any verifier that a legitimate t-of-n decryption occurred, without revealing which t nodes participated.

6 Application: Private Orderbook Matching

We instantiate the FHE+MPC hybrid for a private orderbook matching engine deployed on the Lux network.

6.1 Problem Statement

N market participants submit limit orders (buy/sell, price, quantity, asset pair). The matching engine must:

1. Match overlapping buy/sell orders at the intersection price.
2. Reveal only matched trades to the relevant counterparties.
3. Never reveal the full order book to any single party—including the matching engine itself.

6.2 Protocol

1. **Order submission.** Each participant encrypts their order $(side, price, quantity)$ under the FHE public key and submits the ciphertext on-chain.
2. **Homomorphic matching.** The matching engine (a smart contract or off-chain coprocessor) evaluates the matching logic on encrypted data:
   - Compare encrypted prices: $\text{FHE.Compare}(\text{bid}_i, \text{ask}_j) \to \text{ct}_{match}$
   - Compute matched quantity: $\text{FHE.Min}(\text{qty}_i, \text{qty}_j) \to \text{ct}_{fill}$
   - Generate encrypted trade receipts for matched pairs.

3. **Selective decryption.** Only matched trade receipts are submitted to the MPC committee for threshold decryption. Unmatched orders remain encrypted and are never decrypted.

4. **Settlement.** Decrypted trade receipts are executed on-chain. The MPC attestation proves correct decryption.

6.3 Security Properties

Property	Guarantee
Order privacy	Unmatched orders are never decrypted by anyone
Matching integrity	FHE evaluation is deterministic and verifiable
No front-running	Matching engine cannot read encrypted prices
Selective disclosure	Only counterparties learn trade details
Threshold trust	t-of-n MPC nodes must collude to break privacy

6.4 Throughput: The Bootstrap Is the Bottleneck

Privacy is not free, and the price is throughput. Every comparison and minimum in the homomorphic matching step is a sequence of boolean gates, and each gate requires a programmable bootstrap (PBS) to refresh ciphertext noise. The PBS rate is therefore the hard ceiling on the private path. At this paper's parameters—∼12ms per bootstrapped gate on a single core—a GPU bootstrap kernel sustains only **single- to double-digit full PBS operations per second per GPU**. This is six to seven orders of magnitude below a cleartext CLOB matcher, which fills at ∼100–250 million orders per second per GPU on the public D-Chain path (Lux Lightspeed DEX). The privacy multiple is the gap between those two rates.

Two engineering consequences follow:

- **Batched per market.** Bootstraps are amortized by batching all encrypted orders for a single market into one homomorphic matching pass, rather than evaluating order-by-order. The PBS kernel processes the batch; the per-order cost falls toward the per-gate floor as the batch fills the GPU.
- **P3Q-attested receipts.** Each matched trade receipt that leaves the FHE+MPC pipeline is attested with a P3Q ML-DSA-65 threshold signature before settlement, so the settlement chain verifies correctness without re-running the homomorphic matcher. The ML-DSA-65 verify path is measured at ∼4,714 verifies per second; verification is not the bottleneck—the bootstrap is.

The operative conclusion for system design: the FHE+MPC private path is *not* a drop-in replacement for the public CLOB. It is the correct substrate for low-frequency, value-bearing private markets (dark-pool crossing, treasury rebalancing, confidential securities) where the seven-order-of-magnitude throughput cost buys order-book secrecy that no cleartext matcher can offer. High-frequency markets stay on the public cleartext D-Chain path.

7 Implementation on Lux

7.1 F-Chain: LiquidFHE Runtime

The Lux F-Chain provides the FHE computation environment:

- TFHE boolean circuits via the `luxfi/lattice` library.
- Parameters: $n = 1024$, $q \approx 2^{27}$, NIST Level 1 security (128-bit).
- Bootstrapping key size: 24 MB per user key.
- Gate evaluation: 12ms per bootstrapped gate (single core), 3.2ms with 4-way parallelism.

7.2 M-Chain: MPC Threshold Network

The Lux M-Chain provides the threshold decryption infrastructure:

- FROST-based DKG for FHE key generation.
- t-of-n threshold decryption with configurable (t, n).
- Default configuration: 3-of-5 for application chains, 5-of-9 for the primary network.
- Decryption latency: 35ms for 3-of-5, 82ms for 5-of-9 (including network round trips).

7.3 End-to-End Benchmarks

Operation	**Time**	**Size**
Order encryption (3 fields)	0.4ms	12 KB
Price comparison (16-bit)	38ms	—
Quantity minimum (16-bit)	41ms	—
Full match (compare + fill)	47ms	—
Threshold decryption (3-of-5)	35ms	1.2 KB
Threshold decryption (5-of-9)	82ms	2.1 KB
FROST attestation	12ms	64 B
End-to-end match + decrypt	**94ms**	—

8 Multi-Party Encrypted Workflows

The FHE+MPC hybrid generalizes beyond orderbook matching to any multi-party encrypted workflow.

8.1 Pattern

1. N parties encrypt their private data under a shared FHE public key.
2. A function $f(x_1, \ldots, x_N)$ is evaluated homomorphically on the encrypted inputs.
3. The result $\text{FHE.Eval}(f, \text{ct}_1, \ldots, \text{ct}_N)$ is submitted to the MPC committee.
4. The MPC committee performs threshold decryption and returns the plaintext result with an attestation.

No party ever sees another party's private input. The MPC committee never sees any input—only the encrypted result of the computation.

8.2 Applications

Application	**Private Inputs**	**Decrypted Output**
Private orderbook	Orders (side, price, qty)	Matched trades only
Sealed-bid auction	Bids	Winning bid + winner
Private voting	Votes	Tally only
Credit scoring	Financial data	Score (not data)
Supply chain pricing	Unit costs	Total cost (not components)

9 Comparison with Alternative Approaches

Approach	Trust	Privacy	Latency	Scalability
TEE (SGX/SEV)	Hardware vendor	Side-channel risk	Low	High
Pure MPC (SPDZ)	t-of-n	Full	High ($O(N^2)$)	Low
Pure FHE	Key holder	Compute only	Medium	Medium
ZK proofs	Prover	Prover only	Variable	Medium
FHE+MPC	t-of-n	**Full**	**Medium**	**Medium**

The hybrid achieves the trust distribution of MPC with the non-interactive computation of FHE. The main cost is FHE's bootstrapping overhead (1000–10000× cleartext), which we address through parallelism and application-specific circuit optimization.

10 Conclusion

FHE and MPC solve complementary halves of the private computation problem. FHE eliminates the need to share data; MPC eliminates the single point of trust for the decryption key. The hybrid construction—FHE computation with FROST-based threshold decryption—provides a practical framework for multi-party private computation on blockchain networks.

The implementation on the Lux network (F-Chain for FHE, M-Chain for MPC) achieves 94ms end-to-end latency for private orderbook matching, with configurable trust thresholds and no party ever observing the full order book in cleartext.

References

[1] Chillotti, Gama, Georgieva, Izabachène, "TFHE: Fast Fully Homomorphic Encryption over the Torus," Journal of Cryptology, 2020.

[2] Damgård, Pastro, Smart, Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," CRYPTO 2012.

[3] Komlo, Goldberg, "FROST: Flexible Round-Optimized Schnorr Threshold Signatures," SAC 2020.

[4] Lux Industries, Inc., "Torus Threshold Fully Homomorphic Encryption: Boolean Circuits on Encrypted Data with Distributed Decryption," 2025.

[5] Lux Industries, Inc., "M-Chain: Multi-Party Computation on the Lux Network," 2025.

[6] Shamir, "How to Share a Secret," Communications of the ACM, 1979.