Threshold FHE for Regulatory Compliance: Operator-Blind Computation with Forced Decryption

Zach Kelling

Lux Industries

2024

Abstract

We present a threshold Fully Homomorphic Encryption scheme for regulated financial infrastructure in which an operator evaluates arbitrary boolean circuits on encrypted data without observing plaintext, while a regulator holding t of n key shares retains the ability to force decryption for lawful inspection. The construction combines TFHE gate evaluation (AND, OR, NOT, NAND, MUX over LWE ciphertexts on the torus $\mathbb{T}$) with Shamir-based threshold decryption: the FHE secret key $\mathbf{s}$ is split into n shares such that any t shares reconstruct the partial decryption vector, while any $t-1$ shares are information-theoretically independent of $\mathbf{s}$. We prove three properties: (1) *operator blindness*—the compute node that evaluates homomorphic circuits learns nothing about the plaintext inputs or outputs; (2) *threshold decryptability*—any coalition of t share-holders can decrypt any ciphertext produced by the system; (3) *privacy below threshold*—any coalition of fewer than t share-holders gains no information beyond the public parameters. We instantiate the scheme on the Lux F-Chain using seven concrete applications (private balance, encrypted transfer, sealed-bid auction, private voting, encrypted ML inference, dark pool matching, compliance screening) and report benchmarks from the `luxfi/fhe` library: 12ms per bootstrapped gate, 47ms per encrypted order match, and 230ms per encrypted compliance check at NIST Level 1 (128-bit) security.

Contents

1 Introduction

Blockchain networks that process regulated financial instruments face a fundamental tension between confidentiality and auditability. Market participants demand that their positions, orders, and balances remain private from competitors and the public. Regulators demand the ability to inspect any record when exercising lawful authority. Classical encryption satisfies neither requirement simultaneously: encrypted data is opaque to both the operator and the regulator, while plaintext data is visible to both.

Fully Homomorphic Encryption resolves the computational half of this problem. An FHE scheme allows an untrusted operator to evaluate arbitrary functions on ciphertexts, producing encrypted outputs that decrypt to the correct function value. The operator processes data without ever observing it. However, standard FHE concentrates decryption power in a single key holder, creating a single point of trust that is unacceptable for both privacy (the key holder sees everything) and compliance (if the key holder is adversarial, the regulator cannot force decryption).

Threshold FHE distributes the decryption key via secret sharing. The key $\mathbf{s}$ is split into n shares $\mathbf{s}_1, \ldots, \mathbf{s}_n$ such that any subset of t shares can reconstruct a partial decryption, but fewer than t shares reveal nothing. By assigning shares to a mixed committee of regulated entities and the regulator itself, the scheme simultaneously achieves:

1. **Operator blindness.** The compute node has zero shares and cannot decrypt.
2. **Forced decryption.** The regulator, holding t shares (or the ability to compel t shareholders), can decrypt any ciphertext.
3. **Privacy.** No single entity (including the regulator, if it holds fewer than t shares alone) can unilaterally decrypt.

This paper formalizes the construction, proves its security properties, and demonstrates seven concrete applications deployed on the Lux F-Chain.

2 Preliminaries

2.1 TFHE: Torus Fully Homomorphic Encryption

Definition 2.1 (LWE over the Torus)**.** *Let n be the LWE dimension, $\mathbf{s} \in \{0,1\}^n$ the secret key, and $\sigma > 0$ the noise parameter. An LWE ciphertext encrypting $\mu \in \{0, 1/2\} \subset \mathbb{T}$ is a pair $(\mathbf{a}, b) \in \mathbb{T}^n \times \mathbb{T}$ where $\mathbf{a} \leftarrow \mathbb{T}^n$ uniformly and $b = \langle \mathbf{a}, \mathbf{s} \rangle + \mu + e$ with $e \leftarrow \mathcal{N}(0, \sigma^2)$.*

Definition 2.2 (Decryption)**.** *Given ciphertext $(\mathbf{a}, b)$ and secret key $\mathbf{s}$:*

$$\mu' = b - \langle \mathbf{a}, \mathbf{s} \rangle = \mu + e.$$

The plaintext bit is recovered by rounding: if μ' is closer to 0 than to $1/2$, output 0; otherwise output 1. Correctness holds when $|e| < 1/4$.

Definition 2.3 (Programmable Bootstrapping)**.** *Given a ciphertext c encrypting bit b and a lookup table T, programmable bootstrapping produces a fresh ciphertext encrypting $T[b]$ with noise independent of the input noise. This is the mechanism that enables unlimited gate depth: after each gate, the noise is reset.*

2.2 Shamir Secret Sharing

Definition 2.4 ((t, n)-Shamir Sharing)**.** *To share a secret $s \in \mathbb{Z}_q$, choose random $a_1, \ldots, a_{t-1} \leftarrow \mathbb{Z}_q$ and define the polynomial $f(x) = s + a_1 x + \cdots + a_{t-1} x^{t-1}$. Share i is $s_i = f(i)$ for $i \in \{1, \ldots, n\}$. Any t shares determine f (and thus s) via Lagrange interpolation. Any $t - 1$ shares are uniformly distributed, independent of s.*

2.3 Threshold Decryption

Definition 2.5 (Threshold LWE Decryption)**.** *The secret key $\mathbf{s} \in \{0,1\}^n$ is shared coordinate-wise: for each coordinate $j \in [n]$, the value s_j is shared via (t,n)-Shamir sharing. To decrypt ciphertext $(\mathbf{a}, b)$:*

1. *Each party i computes partial decryption $d_i = \langle \mathbf{a}, \mathbf{s}_i \rangle$ where $\mathbf{s}_i$ is their share vector.*
2. *The combiner collects t partial decryptions and reconstructs $\langle \mathbf{a}, \mathbf{s} \rangle$ via Lagrange interpolation on the partial decryptions.*
3. *The plaintext is round$(b - \langle \mathbf{a}, \mathbf{s} \rangle)$.*

3 Formal Construction

3.1 System Model

The system comprises four roles:

1. **Users.** Submit encrypted inputs. Each user encrypts their data under the system's public key before submission.
2. **Operator.** Evaluates homomorphic circuits on encrypted data. The operator holds no key shares and cannot decrypt.
3. **Decryption committee.** A set of n parties, each holding one share of the FHE secret key. Any t can cooperate to decrypt.
4. **Regulator.** Either a member of the committee with sufficient shares, or an entity that can compel t committee members to participate in decryption.

3.2 Key Generation

Algorithm 1 Threshold FHE Key Generation

Require: Threshold t, committee size n, LWE dimension N, RLWE degree $\hat{N}$
Ensure: Public key pk, bootstrapping key bsk, key shares $\{\mathbf{s}_i\}_{i=1}^n$

$\mathbf{s} \leftarrow \{0,1\}^N$ ▷ Sample LWE secret key
for $j = 1, \ldots, N$ **do**
  Share s_j via (t,n)-Shamir: party i receives $s_{j,i}$
end for
$\mathbf{s}_i \leftarrow (s_{1,i}, \ldots, s_{N,i})$ for each i
$\mathsf{pk} \leftarrow \mathsf{LWE.PublicKey}(\mathbf{s})$
$\mathsf{bsk} \leftarrow \mathsf{BootstrapKey}(\mathbf{s})$ ▷ RGSW encryptions of key bits
Erase $\mathbf{s}$ ▷ Secret key exists only as shares
return $(\mathsf{pk}, \mathsf{bsk}, \{\mathbf{s}_i\}_{i=1}^n)$

After key generation, the plaintext secret key $\mathbf{s}$ is erased. Only the shares $\mathbf{s}_1, \ldots, \mathbf{s}_n$ and the public material $(\mathsf{pk}, \mathsf{bsk})$ persist. The bootstrapping key bsk is public—it consists of RGSW encryptions of the key bits and is required for gate evaluation but does not reveal $\mathbf{s}$.

3.3 Homomorphic Evaluation

The operator receives encrypted inputs and evaluates boolean circuits using the TFHE gate-by-gate bootstrapping pipeline:

1. **Gate evaluation.** For each gate $g \in \{\text{AND}, \text{OR}, \text{NOT}, \text{NAND}, \text{NOR}, \text{XOR}, \text{XNOR}, \text{MUX}, \text{MAJORITY}\}$, the operator computes $c_{\text{out}} = \mathsf{Gate}(c_{\text{in}_1}, c_{\text{in}_2}, \mathsf{bsk})$.
2. **Bootstrapping.** Each gate application performs a programmable bootstrap that resets the noise, enabling unlimited circuit depth.

3. **No decryption.** The operator uses only pk and bsk. Without any share of $\mathbf{s}$, the operator cannot decrypt any intermediate or final ciphertext.

3.4 Threshold Decryption Protocol

When decryption is authorized (either by the committee or compelled by the regulator):

Algorithm 2 Threshold Decryption

Require: Ciphertext $c = (\mathbf{a}, b)$, threshold t, committee subset $S \subseteq [n]$ with $|S| \geq t$
Ensure: Plaintext bit μ

for $i \in S$ **do**
  Party i computes $d_i = \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i$ ▷ Add smudging noise e_i
  Party i publishes d_i
end for
Combiner computes Lagrange coefficients $\lambda_i = \prod_{j \in S \setminus \{i\}} \frac{j}{j-i}$
$D = \sum_{i \in S} \lambda_i \cdot d_i$ ▷ Reconstructs $\langle \mathbf{a}, \mathbf{s} \rangle + e'$
$\mu = \mathrm{round}(b - D)$
return μ

The smudging noise e_i added by each party ensures that the partial decryption d_i does not leak information about $\mathbf{s}_i$ beyond what is implied by the final plaintext. The smudging noise is chosen large enough to statistically mask the LWE error but small enough that the total accumulated noise remains below the decryption threshold $1/4$.

4 Security Proofs

4.1 Operator Blindness

Theorem 4.1 (Operator Blindness)**.** *Let Π be the threshold TFHE scheme described in Section 3. An operator that holds only the public key pk and bootstrapping key bsk (but no shares of $\mathbf{s}$) cannot distinguish encryptions of 0 from encryptions of 1 with advantage better than $\mathsf{negl}(\lambda)$, assuming the $\mathrm{LWE}_{n,q,\sigma}$ problem is hard.*

Proof sketch. The bootstrapping key bsk consists of RGSW encryptions of the secret key bits under the RLWE key. Under the RLWE assumption (which reduces to LWE), these encryptions are computationally indistinguishable from encryptions of zero. Therefore the operator's view—consisting of pk, bsk, and a collection of LWE ciphertexts—is computationally indistinguishable between the case where all ciphertexts encrypt 0 and the case where they encrypt 1. This is a standard IND-CPA argument for LWE-based encryption, extended to the bootstrapping key via the RLWE hardness assumption. □

4.2 Threshold Decryptability

Theorem 4.2 (Threshold Decryptability)**.** *For any ciphertext c produced by the system (either a fresh encryption or the output of homomorphic evaluation), any subset S of at least t shareholders can correctly decrypt c with probability $1 - \mathsf{negl}(\lambda)$.*

Proof sketch. By the correctness of Shamir's secret sharing (Definition 2.4), any t shares uniquely determine the polynomial f and thus the secret s_j for each coordinate j. Lagrange interpolation on the partial decryptions $\{d_i\}_{i \in S}$ reconstructs $\langle \mathbf{a}, \mathbf{s} \rangle$ plus accumulated smudging noise. The total noise is bounded by $|e| + \sum_{i \in S} |\lambda_i| \cdot |e_i|$. By choosing the smudging parameter $\sigma_{\mathrm{smudge}} = 2^{-\lambda} \cdot \sigma$ and observing that the Lagrange coefficients satisfy $\sum |\lambda_i| \leq \binom{n}{t} \cdot t^t$ (which is polynomial in n),

the total noise remains below $1/4$ with overwhelming probability for standard parameter choices ($N = 1024$, $q \approx 2^{27}$). Rounding then recovers the correct plaintext bit. □

4.3 Privacy Below Threshold

Theorem 4.3 (Privacy Below Threshold)**.** *Any coalition of fewer than t share-holders, even in cooperation with the operator, learns no information about any plaintext encrypted under the system beyond what is implied by the public parameters.*

Proof sketch. This follows from two independent arguments:

1. **Information-theoretic secret sharing.** By the privacy property of Shamir's scheme, any $t-1$ shares are uniformly distributed over $\mathbb{Z}_q^{t-1}$, independent of the secret. Therefore $t-1$ share-holders learn nothing about $\mathbf{s}$.
2. **Computational LWE hardness.** Even with $t-1$ shares, reconstructing the remaining share would require solving an LWE instance. The operator contributes only public material (pk, bsk, ciphertexts), which is computationally indistinguishable from random under LWE.

The composition of information-theoretic and computational security holds because the information-theoretic argument applies to the secret sharing layer independently of the computational argument for the encryption layer. □

5 Compliance Architecture

5.1 Regulatory Share Distribution

The threshold t and share distribution are configured to enforce regulatory access policy:

Configuration	t	n	**Share Distribution**
Regulator can decrypt alone	1	5	Regulator holds 1 share
Regulator + 1 custodian	2	5	Regulator: 1, custodians: 4
Regulator + quorum	3	7	Regulator: 1, custodians: 6
Multi-regulator	3	7	SEC: 1, FINRA: 1, custodians: 5

Table 1: Example share distributions for different compliance regimes.

The key design principle is that the regulator always has the *ability* to obtain t shares (either by holding them directly or by compelling custodians via legal process), but the operator *never* has any shares.

5.2 Forced Decryption Protocol

When a regulator issues a lawful decryption order:

1. The regulator identifies the ciphertext(s) to be decrypted and presents a signed decryption request to the committee.
2. Each compelled committee member computes their partial decryption d_i and submits it to the regulator.
3. The regulator reconstructs the plaintext using Algorithm 2.
4. An on-chain audit log records that decryption was performed (without revealing the plaintext), providing transparency.

This protocol ensures that decryption is auditable: the on-chain record proves that a lawful decryption occurred, and the threshold structure ensures that no unauthorized decryption is possible.

5.3 Operator Cannot Collude

Even if the operator is subpoenaed or compromised, it cannot assist in decryption because it holds zero shares. The operator's contribution to the system is purely computational: it evaluates homomorphic circuits using only public material. This architectural separation means that compromising the operator reveals no plaintext data.

6 Concrete Instantiations

We instantiate the threshold FHE scheme in seven applications deployed on the Lux F-Chain, each demonstrating a different aspect of the operator-blind computation model.

6.1 Private Balance

Users encrypt their token balances under the system public key. The chain state stores only ciphertexts. Balance queries return encrypted values that only the user (who holds the encryption randomness) or the decryption committee can read. The operator (validator nodes) processes transfers homomorphically without observing balances.

6.2 Encrypted Transfer

A transfer of amount a from user A to user B is computed as: $\mathsf{bal}'_A = \mathsf{Sub}(\mathsf{bal}_A, \mathsf{Enc}(a))$, $\mathsf{bal}'_B = \mathsf{Add}(\mathsf{bal}_B, \mathsf{Enc}(a))$. The operator verifies $\mathsf{bal}'_A \geq 0$ via an encrypted comparison (the CMPCOMBINE optimization from the TFHE library) without learning a, bal_A, or bal_B.

6.3 Sealed-Bid Auction

Bidders submit encrypted bids. The operator evaluates a homomorphic maximum circuit to determine the winner. Only the winning bid is decrypted (by the committee), and losing bids remain encrypted. The regulator can audit all bids post-hoc if required.

6.4 Private Voting

Shareholders submit encrypted votes. The operator tallies votes homomorphically. Only the final tally is decrypted. Individual votes remain private unless the regulator exercises forced decryption for fraud investigation.

6.5 Encrypted ML Inference

A compliance model (e.g., AML scoring) runs on encrypted transaction data. The model parameters are public, but the input data and output scores are encrypted. The operator evaluates the model as a boolean circuit via TFHE gates. Only the compliance officer (a committee member) decrypts the scores.

6.6 Dark Pool Matching

Institutional orders are submitted encrypted. The matching engine evaluates order crossing homomorphically. Matched trades are decrypted by the committee for settlement. Unmatched orders remain encrypted, preventing information leakage about institutional order flow.

6.7 Compliance Screening

Transaction metadata is encrypted and screened against a sanctions list represented as a homomorphic lookup table. The screening result (pass/fail) is encrypted. The compliance officer decrypts results; the operator learns nothing about either the transaction or the sanctions list entries.

7 Performance

All benchmarks are from the `luxfi/fhe` library running on a single core (Apple M4 Max, 4.05 GHz).

Operation	Latency	Parameters
Single bootstrapped gate	12 ms	N=1024, $q \approx 2^{27}$
8-bit encrypted addition	96 ms	8 gates deep
8-bit encrypted comparison	108 ms	CMPCOMBINE
Encrypted order match	47 ms	Price + quantity compare
Compliance check	230 ms	AML score circuit
Threshold partial decrypt	0.3 ms	Per-party, n=7
Lagrange reconstruction	0.1 ms	t=3 shares

Table 2: Benchmark results for threshold FHE operations.

Threshold decryption adds negligible overhead compared to the homomorphic computation: partial decryption is a single inner product ($O(N)$ multiplications), and Lagrange reconstruction is $O(t^2)$ field operations.

8 Formal Verification

The core properties are mechanized in Lean 4 in the `lux/proofs` repository:

- `Crypto.FHE.TFHE`: correctness of encryption/decryption, homomorphic gate correctness, noise growth bounds, bootstrapping preservation.
- `Crypto.Threshold.LSS`: Shamir sharing correctness, privacy, linearity, composability.
- `Crypto.Threshold.Composition`: uniform access structure across TFHE threshold decryption, FROST, and CGGMP21.
- `Crypto.ThresholdDecrypt`: combined proof that t shares reconstruct correctly and $t-1$ shares reveal nothing.

9 Related Work

Threshold FHE was introduced by Asharov et al. [3] in the BGV setting. Boneh et al. [4] extended threshold decryption to the CKKS scheme for approximate arithmetic. Our construction is the first to combine TFHE (boolean circuits) with threshold decryption for regulatory compliance in a blockchain setting.

The FHE-MPC hybrid model is explored in Mouchet et al. [5] (Lattigo library), who demonstrate multiparty CKKS computation. We differ in two respects: we use TFHE (exact boolean circuits, not approximate arithmetic) and we explicitly model the regulatory forced-decryption requirement.

Fhenix [6] and Zama [7] implement FHE on EVM chains but use a single decryption key held by a trusted party or a TEE. Our threshold approach eliminates this single point of trust.

10 Conclusion

Threshold FHE resolves the fundamental tension between privacy and regulatory access in financial infrastructure. The operator processes encrypted data without observing it. The regulator can compel decryption when legally authorized. No single party—not the operator, not any individual committee member, not even the regulator acting alone (unless configured with t shares)—can unilaterally access plaintext. The construction is deployed on the Lux F-Chain with seven concrete applications and mechanized proofs in Lean 4.

References

[1] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. *TFHE: Fast Fully Homomorphic Encryption over the Torus*. J. Cryptology, 33(1):34–91, 2020.

[2] A. Shamir. *How to Share a Secret*. Communications of the ACM, 22(11):612–613, 1979.

[3] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, D. Wichs. *Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE*. EUROCRYPT 2012.

[4] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. Rasmussen, A. Sahai. *Threshold Cryptosystems From Threshold Fully Homomorphic Encryption*. CRYPTO 2018.

[5] C. Mouchet, J. Bossuat, J.-P. Hubaux, C. Troncoso. *Multiparty Homomorphic Encryption from Ring-Learning-with-Errors*. PETS 2021.

[6] Fhenix. *Fhenix: Confidential Smart Contracts for Ethereum*. Whitepaper, 2024.

[7] Zama. *fhEVM: Confidential Smart Contracts on the EVM using Fully Homomorphic Encryption*. 2024.

[8] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. *Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks*. IACR ePrint 2021/091.

[9] T. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. CRYPTO 1991.

[10] I. Damgård, V. Pastro, N. Smart, S. Zakarias. *Multiparty Computation from Somewhat Homomorphic Encryption*. CRYPTO 2012.