Torus Threshold Fully Homomorphic Encryption: Boolean Circuits on Encrypted Data with ~~Distributed Decryption~~

Zach Kelling

Lux Industries

2025

Abstract

We present a threshold variant of Torus Fully Homomorphic Encryption (TFHE) that enables evaluation of arbitrary boolean circuits on encrypted bits with distributed decryption across t-of-n parties. The scheme encrypts individual bits as Learning With Errors (LWE) ciphertexts over the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, performs gate evaluation through programmable bootstrapping (blind rotation), and requires no interaction between parties during computation—only the decryption phase requires threshold cooperation. We describe the core TFHE gate evaluation pipeline: (1) LWE encryption of a single bit with noise parameter σ, (2) blind rotation using RGSW ciphertexts of the secret key bits to evaluate a lookup table encoding the gate function, (3) sample extraction and modulus switching to produce a fresh LWE ciphertext of the gate output, and (4) the CMPCOMBINE optimization that fuses comparison and bootstrap operations to evaluate multi-bit comparisons in a single bootstrapping pass. We formalize the threshold decryption protocol: each party holds a share of the LWE secret key $\mathbf{s}$, computes a partial decryption $d_i = \langle \mathbf{c}, \mathbf{s}_i \rangle$, and the combiner reconstructs the plaintext from t partial decryptions. We prove that threshold TFHE achieves IND-CPA security under the LWE assumption with the same parameters as standard TFHE (N=1024, $q \approx 2^{27}$, providing 128-bit security). The implementation builds on the `luxfi/lattice` library for ring operations and supports all 9 boolean gates (AND, OR, XOR, NOT, NAND, NOR, XNOR, MUX, MAJORITY). Benchmarks show 12ms per bootstrapped gate on a single core, enabling EVM-level boolean circuit evaluation on encrypted data with practical performance.

Contents

1 Introduction

Fully Homomorphic Encryption (FHE) enables computation on encrypted data without decrypting it, a capability with profound implications for blockchain privacy. Smart contracts operating on encrypted inputs can implement confidential token transfers, private auctions, sealed-bid voting, and encrypted machine learning inference—all without revealing the underlying data to validators or observers.

TFHE (Torus FHE) [1] is the leading boolean-circuit FHE scheme, encrypting individual bits and evaluating gates through a bootstrapping procedure that refreshes the noise after each gate. Unlike CKKS (which operates on approximate arithmetic) or BGV/BFV (which operate on integer arithmetic), TFHE's bit-level granularity is ideally suited for arbitrary computation: any function expressible as a boolean circuit can be evaluated gate-by-gate.

This paper extends TFHE with threshold decryption for blockchain applications: the secret key is distributed among n validators via Shamir sharing, and any t validators can collaboratively decrypt the result without reconstructing the full key. This is critical for decentralized FHE, where no single party should possess the decryption capability.

Contributions.

1. A complete description of the TFHE boolean gate evaluation pipeline, including the blind rotation and CMPCOMBINE optimization (§3).
2. Threshold decryption protocol with security proof under LWE (§4).
3. Concrete parameter selection and security analysis (§5).
4. Integration with the Lux FHE VM for on-chain encrypted computation (§6).
5. Benchmarks for all boolean gates and multi-bit operations (§7).

2 TFHE Preliminaries

2.1 Torus Arithmetic

Definition 2.1 (Torus)**.** *The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ is the group of real numbers modulo 1. In practice, we discretize to $\mathbb{T}_q = \mathbb{Z}_q/q$ by mapping $x \in \mathbb{T}$ to $\lfloor x \cdot q \rceil \in \mathbb{Z}_q$.*

Definition 2.2 (LWE Encryption)**.** *An LWE ciphertext encrypting bit $m \in \{0,1\}$ under secret key $\mathbf{s} \in \mathbb{Z}_q^n$ is:*

$$ct = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

where $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $b = \langle \mathbf{a}, \mathbf{s} \rangle + m \cdot \lfloor q/2 \rfloor + e$ with noise $e \xleftarrow{\$} D_\sigma$.

Decryption computes:

$$\hat{m} = \left\lfloor \frac{b - \langle \mathbf{a}, \mathbf{s} \rangle}{q/2} \right\rceil \mod 2 \tag{1}$$

This succeeds when $|e| < q/4$.

2.2 RGSW Encryption

Definition 2.3 (RGSW Ciphertext)**.** *An RGSW (Ring-GSW) ciphertext of $m \in \{0,1\}$ under RLWE secret $s(X) \in R_q$ is a matrix $\mathbf{C} \in R_q^{2\ell \times 2}$ such that for any RLWE ciphertext $(\mathbf{a}', b')$, the external product $\mathbf{C} \boxdot (\mathbf{a}', b')$ yields an RLWE encryption of $m \cdot Dec(\mathbf{a}', b')$ with controlled noise growth.*

RGSW ciphertexts of the secret key bits form the *bootstrap key*: $\text{BSK} = \{\text{RGSW}(s_i)\}_{i=1}^n$ where s_i are the LWE secret key coefficients.

3 Boolean Gate Evaluation

3.1 Bootstrapping via Blind Rotation

The core TFHE operation is *programmable bootstrapping*: given an LWE ciphertext $\mathrm{ct} = (\mathbf{a}, b)$ encrypting m, produce a fresh LWE ciphertext encrypting $f(m)$ for an arbitrary function $f : \{0,1\} \to \{0,1\}$.

Algorithm 1 TFHE Blind Rotation (Bootstrapping)

Require: LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$
Require: Bootstrap key $\mathrm{BSK} = \{\mathrm{RGSW}(s_i)\}_{i=1}^n$
Require: Lookup table (test vector) $\mathbf{v} \in R_Q$ encoding gate function
▷ Step 1: Initialize accumulator with rotated test vector
$\mathrm{ACC} \leftarrow X^{-\tilde{b}} \cdot \mathbf{v}$ ▷ RLWE encryption of rotated LUT
▷ where $\tilde{b} = \lfloor b \cdot 2N/q \rceil$ is the scaled phase
▷ Step 2: Blind rotation — multiply by X^{a_i} conditioned on s_i
for $i = 1$ to n **do**
  $\tilde{a}_i \leftarrow \lfloor a_i \cdot 2N/q \rceil$ ▷ scaled coefficient
  $\mathrm{ACC} \leftarrow \mathrm{CMUX}(\mathrm{BSK}[i], X^{\tilde{a}_i} \cdot \mathrm{ACC}, \mathrm{ACC})$
  ▷ CMUX: if $s_i = 1$, rotate by $X^{\tilde{a}_i}$; else keep ACC
end for
▷ Step 3: Sample extraction — extract coefficient 0 as LWE ciphertext
$\mathrm{ct}_{\mathrm{out}} \leftarrow \mathrm{SampleExtract}(\mathrm{ACC}, 0)$
return $\mathrm{ct}_{\mathrm{out}}$ ▷ fresh LWE encryption of $f(m)$

The CMUX gate is implemented as an RGSW external product:

$$\mathrm{CMUX}(\mathrm{RGSW}(s_i), \mathrm{ct}_1, \mathrm{ct}_0) = \mathrm{RGSW}(s_i) \boxdot (\mathrm{ct}_1 - \mathrm{ct}_0) + \mathrm{ct}_0$$

This selects ct_1 if $s_i = 1$ and ct_0 if $s_i = 0$, all homomorphically.

3.2 Gate Evaluation via Test Vectors

Each boolean gate is implemented by choosing the appropriate test vector $\mathbf{v}$:

Gate	Input	Test Vector Encoding
AND	$\mathrm{ct}_1, \mathrm{ct}_2$	$\mathrm{Bootstrap}(\mathrm{ct}_1 + \mathrm{ct}_2 - q/4, \mathbf{v}_{\mathrm{AND}})$
OR	$\mathrm{ct}_1, \mathrm{ct}_2$	$\mathrm{Bootstrap}(\mathrm{ct}_1 + \mathrm{ct}_2 + q/4, \mathbf{v}_{\mathrm{OR}})$
XOR	$\mathrm{ct}_1, \mathrm{ct}_2$	$\mathrm{Bootstrap}(2(\mathrm{ct}_1 + \mathrm{ct}_2), \mathbf{v}_{\mathrm{XOR}})$
NAND	$\mathrm{ct}_1, \mathrm{ct}_2$	$\mathrm{Bootstrap}(-\mathrm{ct}_1 - \mathrm{ct}_2 + q/4, \mathbf{v}_{\mathrm{NAND}})$
NOT	ct_1	$(-\mathbf{a}, q/2 - b)$ (no bootstrap needed)
MUX	$\mathrm{ct}_s, \mathrm{ct}_1, \mathrm{ct}_0$	Two bootstraps + addition
MAJORITY	$\mathrm{ct}_1, \mathrm{ct}_2, \mathrm{ct}_3$	$\mathrm{Bootstrap}(\mathrm{ct}_1 + \mathrm{ct}_2 + \mathrm{ct}_3, \mathbf{v}_{\mathrm{MAJ}})$

Table 1: Boolean gate implementations via TFHE bootstrapping

3.3 CMPCOMBINE Optimization

Definition 3.1 (CMPCOMBINE)**.** *The CMPCOMBINE gate fuses comparison and combination operations for multi-bit integers. For encrypted bits $a_0, \ldots, a_{k-1}$ and $b_0, \ldots, b_{k-1}$ repre-*

senting k-bit integers A and B:

$$\mathit{CMPCOMBINE}(A, B) = \begin{cases} 1 & \text{if } A > B \\ 0 & \text{if } A \leq B \end{cases}$$

evaluated in a single bootstrapping chain rather than k independent comparisons.

Algorithm 2 CMPCOMBINE — Multi-Bit Comparison in One Pass

Require: Encrypted bits $\mathrm{ct}_{a_0}, \ldots, \mathrm{ct}_{a_{k-1}}, \mathrm{ct}_{b_0}, \ldots, \mathrm{ct}_{b_{k-1}}$

```
carry ← Encrypt(0)                                   ▷ initial carry = 0
for i = 0 to k − 1 do                                ▷ LSB to MSB
    diff_i ← ct_{a_i} − ct_{b_i}                     ▷ homomorphic subtraction
    sum_i ← diff_i + carry
                                                     ▷ Bootstrap with carry-propagation test vector
    carry ← Bootstrap(sum_i, v_carry)
end for
return carry                                         ▷ encrypted comparison result
```

Theorem 3.1 (CMPCOMBINE Correctness)**.** *The CMPCOMBINE gate correctly computes $A > B$ using k bootstrappings (one per bit), compared to the naive approach requiring $2k - 1$ bootstrappings (one XOR + one AND per bit plus carry propagation).*

Proof. The carry c_i after processing bit i satisfies:

$$c_i = \begin{cases} 1 & \text{if } A[0:i] > B[0:i] \\ 0 & \text{if } A[0:i] \leq B[0:i] \end{cases}$$

where $A[0:i]$ denotes the $(i+1)$-bit prefix. The test vector $\mathbf{v}_{\text{carry}}$ encodes the function $f(x) = 1$ iff $x > 0$ (where x is the phase of sum_i).

Since $\mathrm{sum}_i = (a_i - b_i) + c_{i-1}$, the phase encodes:

- $a_i > b_i$: phase > 0 regardless of $c_{i-1} \Rightarrow c_i = 1$.
- $a_i = b_i$: phase $= c_{i-1} \Rightarrow c_i = c_{i-1}$ (carry propagated).
- $a_i < b_i$: phase < 0 regardless of $c_{i-1} \Rightarrow c_i = 0$.

Each bit requires one homomorphic subtraction ($O(1)$, no bootstrap) and one bootstrap for the carry, totaling k bootstraps. The naive approach requires separate XOR, AND, and carry gates per bit, totaling $\geq 2k - 1$ bootstraps. □

4 Threshold Decryption

4.1 Key Distribution

The LWE secret key $\mathbf{s} \in \mathbb{Z}_q^n$ is shared via Shamir secret sharing: party i holds $\mathbf{s}_i$ such that $\mathbf{s} = \sum_{i \in S} \lambda_i \mathbf{s}_i$ for any t-subset S.

4.2 Partial Decryption Protocol

Algorithm 3 Threshold TFHE Decryption

Require: Ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$
Require: Party i's share $\mathbf{s}_i$, signers S with $|S| \geq t$

▷ Phase 1: Each party computes partial decryption
$d_i \leftarrow \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i$ ▷ e_i: smudging noise for privacy
send d_i to combiner
▷ Phase 2: Combiner reconstructs plaintext
$d \leftarrow \sum_{i \in S} \lambda_i \cdot d_i$ ▷ $\approx \langle \mathbf{a}, \mathbf{s} \rangle + \text{smudge}$
$\hat{m} \leftarrow \lfloor (b - d)/(q/2) \rceil \mod 2$
return $\hat{m}$

4.3 Security of Threshold Decryption

Theorem 4.1 (Threshold IND-CPA Security)**.** *The threshold TFHE scheme achieves IND-CPA security under the LWE assumption: an adversary controlling $t-1$ parties learns nothing about the encrypted plaintext beyond what is revealed by the decryption output.*

Proof. The adversary observes $t-1$ partial decryptions $\{d_i\}_{i \in C}$ where C is the corrupted set with $|C| = t-1$. Each $d_i = \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i$.

By the Shamir sharing property, the $t-1$ shares $\{\mathbf{s}_i\}_{i \in C}$ reveal no information about $\mathbf{s}$ (the secret requires t shares for reconstruction). The smudging noise e_i with $\sigma_{\text{smudge}} \gg \sigma$ ensures that the partial decryptions are statistically independent of the missing share.

Formally, for any two messages m_0, m_1 and the corresponding ciphertexts:

$$\{\text{ct}, \{d_i\}_{i \in C}\} \overset{s}{\approx} \{\text{ct}, \{u_i\}_{i \in C}\}$$

where $u_i \overset{\$}{\leftarrow} \mathbb{Z}_q$ are uniform, with statistical distance $\leq n \cdot t \cdot \sigma / \sigma_{\text{smudge}} < 2^{-128}$ for appropriate $\sigma_{\text{smudge}} = 2^{40} \cdot \sigma$. □

Lemma 4.2 (Decryption Correctness Under Threshold)**.** *Threshold decryption succeeds (correct plaintext recovery) when the total noise (original LWE noise + smudging noise from all parties) satisfies:*

$$|e| + t \cdot |\sigma_{smudge}| < q/4$$

For parameters $\sigma = 3.2$, $\sigma_{smudge} = 2^{10}$, $t = 3$, and $q = 2^{27}$: the noise budget is $3 \times 2^{10} \approx 2^{12} \ll 2^{25} = q/4$.

5 Parameter Selection

Proposition 5.1 (Security Level)**.** *The PN10QP27 parameter set provides ≥ 128-bit classical security and ≥ 100-bit quantum security, matching the security of the standard TFHE parameter set. The LWE instance has dimension $n = 1024$ and noise ratio $\alpha = \sigma\sqrt{2\pi}/q \approx 2^{-23.4}$, for which the best known attacks (primal and dual lattice attacks) require $\geq 2^{128}$ classical operations.*

6 FHE VM Integration

The threshold TFHE scheme integrates with the Lux FHE VM as follows:

1. **Encryption**: Users encrypt inputs using the network's public key.
2. **Evaluation**: EVM precompiles execute FHE gates on encrypted data without decryption. Each Solidity opcode maps to a sequence of TFHE gates.

Parameter	Symbol	Value
LWE dimension	n	1024 (PN10QP27)
LWE modulus	q	$2^{27} \approx 134\text{M}$
Blind rotation dimension	N	1024
Blind rotation modulus	Q	2^{27} (same ring)
Key decomposition base	B_g	$2^7 = 128$
Noise std dev	σ	3.2
Smudging noise	σ_{smudge}	2^{10}

Table 2: TFHE parameter set (PN10QP27)

3. **Decryption**: When a smart contract requests decryption, the threshold decryption protocol distributes the task among t-of-n validators.

The `FHE.sol` library exposes FHE operations as Solidity functions: `FHE.add`, `FHE.sub`, `FHE.mul` for encrypted unsigned integers, and `FHE.lt`, `FHE.eq` for encrypted comparisons.

7 Empirical Evaluation

7.1 Gate Performance

Gate	Time (ms)	Ciphertext Size (bytes)
NOT	0.001	8,208
AND	12.3	8,208
OR	12.1	8,208
XOR	12.4	8,208
NAND	12.2	8,208
NOR	12.3	8,208
XNOR	12.5	8,208
MUX	24.6	8,208
MAJORITY	12.4	8,208

Table 3: TFHE gate evaluation performance (single core)

7.2 Multi-Bit Operations

Operation	Bits	Time (ms)
Addition (uint32)	32	780
Comparison (uint32)	32	395 (CMPCOMBINE)
Multiplication (uint8)	8	1,200
Equality (uint32)	32	420

Table 4: Multi-bit operation performance

Threshold	Decryption Time	Communication
2-of-3	5ms	24 KB
3-of-5	12ms	40 KB
5-of-10	28ms	80 KB

Table 5: Threshold decryption performance

7.3 Threshold Decryption

7.4 F-Chain NTT primitive on Apple Metal

The F-Chain TFHE pipeline reduces every gate evaluation to NTT-domain multiplications. The Metal NTT kernel (`luxcpp/fhe`) reaches ∼23.63× CPU at N=4,096, B=128 on Apple M1 Max (LP-137 Phase-2 roll-up, 2026-04-26 reproduction; Phase-3 unchanged). Smaller batches degrade gracefully: 9.0× at N=4,096, B=32, and 6.2× at N=8,192, B=128. This is one of three production workloads on the LP-137 substrate that beat CPU end-to-end (alongside C-Chain and B-Chain BLS aggregate batched verify) and is plausibly state-of-the-art on Apple Silicon for the TFHE NTT shape. The full empirical surface (CPU, Metal, MLX, CUDA, WGSL) is exercised by 2 174 gtest cases across the four backends that compile on each test host; CPU is the byte-equivalence ground truth that every backend is asserted against.

8 Conclusion

We have presented a threshold variant of TFHE that enables boolean circuit evaluation on encrypted data with distributed decryption. The CMPCOMBINE optimization reduces multi-bit comparison from $2k-1$ to k bootstrappings, making encrypted integer operations practical. The threshold decryption protocol preserves IND-CPA security under the LWE assumption while requiring only one round of communication. Integration with the Lux FHE VM enables on-chain confidential computation with the full expressiveness of boolean circuits.

References

[1] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptology*, 2020.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009.

[3] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *JACM*, 2009.

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM TOCT*, 2014.

[5] D. Boneh et al., "Threshold cryptosystems from threshold fully homomorphic encryption," in *CRYPTO*, 2018.

[6] G. Asharov et al., "Multiparty computation with low communication, computation and interaction via threshold FHE," in *EUROCRYPT*, 2012.