

Dismantling the Circle Arc Trojan Horse

Part II: Cryptographic Foundations & E-Barter Architectures

Project Anchor — FluidCluster.pt

FluidCluster Research Nucleus

School of Vanguard Cryptography and Confidential Computing

September 2026

Mathematical Summation: Silicon vs. Lattice

Lattice Cryptography Foundations (LWE)

Encrypted Multilateral Netting & E-Barter

The Mathematical Shield: Learning With Errors (LWE)

Instead of trusting hardware enclaves (TEEs) prone to microarchitectural leakage, the photonic **fhEVM** relies on the hardness of Lattice-based problems.

LWE Encryption Setup

Given a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a public matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, a ciphertext sample $(\mathbf{a}, b)$ encrypting a plaintext bit $m \in \{0, 1\}$ is defined as:

$$\mathbf{a} \sim \mathbb{Z}_q^n, \quad b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \cdot \left\lfloor \frac{q}{2} \right\rfloor \pmod{q}$$

where $e \sim \chi$ is a small error/noise drawn from a discrete Gaussian distribution χ .

- **Semantic Security:** Discerning $\mathbf{s}$ from $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ is asymptotically equivalent to finding the shortest vector in a high-dimensional lattice (worse-case hardness).

The Noise Bottleneck & Photonic Bootstrapping

Homomorphic operations alter the noise parameter e , bounds of which must be managed to preserve decryption integrity.

Noise Accumulation

$$\text{Enc}(m_1) + \text{Enc}(m_2) \implies e_{\text{new}} = e_1 + e_2$$

$$\text{Enc}(m_1) \times \text{Enc}(m_2) \implies e_{\text{new}} \approx e_1 \cdot e_2$$

If $e_{\text{new}} \geq \frac{q}{4}$, the plaintext state undergoes irreversible entropy corruption.

The Photonic Resolution

- ▶ Traditional CPUs spend milliseconds running a **Bootstrapping** circuit (evaluating decryption homomorphically to refresh noise).
- ▶ **Photonic Processors** execute Torus-FHE (TFHE) blind rotations via optical phase modulation at light speed ($\sim$ nanoseconds).

The E-Barter Architecture: Open Account Netting

Eliminating debt-based fiat requires resolving gross obligations into net zero-sum transactions blindly without token intermediation.

Encrypted Matrix Formulation

Let $\mathbf{O}$ be an $N \times N$ encrypted matrix where $\mathbf{O}_{ij} = \text{Enc}(v_{ij})$ represents the un-cleared gross obligation of Participant i to Participant j .

1. **Homomorphic Row-Column Netting:** Calculate net exposure vector $\mathbf{E}$ blindly:

$$\mathbf{E}_i = \sum_{j=1}^N \mathbf{O}_{ji} - \sum_{j=1}^N \mathbf{O}_{ij} \pmod{q}$$

2. **Circuit Closure:** A Zero-Knowledge Proof (ZKP) verifies $\sum_{i=1}^N \mathbf{E}_i = 0$ while the balance states $\mathbf{E}_i$ remain fully hidden within the fhEVM.

Smart Contract Blueprint: Blind Netting Engine

Conceptual interface for the **fhEVM-based E-Barter Core** mapping out the secure state transitions natively over ciphertexts (euint64):

fhEVM Barter Contract Interface

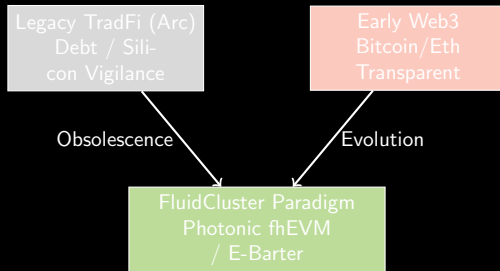
```
pragma solidity ^0.8.24;
import "fhevm/lib/TFHE.sol";

contract PhotonicNettingEngine {
    // Encrypted ledger matrix: Mapping(Participant => Mapping(Counterparty => EncryptedObligation))
    mapping(address => mapping(address => euint34)) private encryptedLedger;
    mapping(address => euint34) private netExposures;

    function recordObligation(address _to, bytes calldata _encryptedValue) public {
        euint34 amount = TFHE.asEuint34(_encryptedValue);
        encryptedLedger[msg.sender][_to] = TFHE.add(encryptedLedger[msg.sender][_to], amount);
    }

    function executeBlindNetting(address[] memory participants) public {
        for (uint i = 0; i < participants.length; i++) {
            euint34 totalIn = TFHE.asEuint34(0);
            euint34 totalOut = TFHE.asEuint34(0);
            // Multi-lateral loop executed inside the fhEVM executor blinds...
            netExposures[participants[i]] = TFHE.sub(totalIn, totalOut);
        }
    }
}
```

The Post-Bitcoin Synthesis



- ▶ **Mathematical Completeness:** Auditable settlement parameters without data disclosure.
- ▶ **Systemic Security:** Total elimination of MEV vectors because memory blocks remain fully blind up to execution settlement.