The FluidCluster Protocol

A Post-Quantum Confidential Multilateral Netting Architecture via Fully Homomorphic Encryption, Zero-Knowledge Invoices, and Proof of Minimization

FluidCluster Core Research Swarm

Cryptography, DeFi Engineering, and Graph Algorithmic Consortium

October 2026

Abstract

Multilateral netting with discrete credit limits and transaction costs represents a fundamentally NP-hard graph-optimization challenge. Traditional financial settlement infrastructures rely heavily on centralized clearing houses (CCPs) operating over unencrypted databases, exposing systemic counterparty vulnerabilities and severe data leakage vectors. This paper presents **FluidCluster**, a decentralized, post-quantum safe financial clearing protocol designed to execute zero-knowledge clearing operations under complete computational blindness. FluidCluster encapsulates private trade obligations and counterparty credit parameters inside a Torus Fully Homomorphic Encryption (TFHE) layered ledger engine, processing limits via invariant-preserving branchless arithmetic. Double-financing is structurally prevented by anchoring cryptographic transaction commitments to verifiable enterprise resource inputs via Zero-Knowledge (ZK) proofs. To settle non-deterministic topological minimization challenges, FluidCluster introduces a decentralized mining loop called **Proof of Minimization** (PoM). Under discrete block timelocks, decentralized entities compete using Mixed-Integer Programming (MIP) to discover maximum liquidity release vectors; winners receive proportional execution commission rewards. Furthermore, we outline the integration of Optical Integrated Circuits (PICs) via Mach-Zehnder Interferometer (MZI) meshes to offload costly polynomial convolutions at the speed of light.

Contents

1 Introduction & Protocol Novelty **5**
- 1.1 Background and Structural Bottlenecks . . . 5
- 1.2 The FluidCluster Architecture Paradigm . . . 5
- 1.3 Core Security Invariants . . . 5

2 The Blind Execution Substrate (fhEVM) **6**
- 2.1 Torus Fully Homomorphic Encryption (TFHE) Foundations . . . 6
- 2.2 Branchless Novation Logic and Side-Channel Mitigation . . . 6

3 Connected Collateralization & Zero-Knowledge Invoices **7**
- 3.1 ATCUD Anchoring and Document Provenance . . . 7
- 3.2 Invoice Nullifiers and Anti-Double-Financing Circuits . . . 7

4 The Proof of Minimization Mining Protocol **8**
- 4.1 NP-Hard Graph Complexity Dynamics . . . 8
- 4.2 Timelock Auctions and Solver Competition Loops . . . 8

5 Conclusion & Future R&D: An Open Call to Innovation **9**
- 5.1 Summary of Protocol Milestones . . . 9
- 5.2 The Open Hardware and Algorithmic Challenge . . . 9

A Reference Implementations **10**
- A.1 Foundational Invariant Simulation (Python) . . . 10
- A.2 Smart Contract Core Architecture (Solidity) . . . 11
- A.3 Mixed-Integer Programming Solver Core (Python) . . . 12
- A.4 fhEVM Multi-Flow Verification Infrastructure (Solidity) . . . 13
- A.5 Zero-Knowledge Provenance Circuit Design (Circom) . . . 14
- A.6 Mach-Zehnder Electro-Optical Modulator Simulation (Python) . . . 15

List of Figures

List of Tables

Chapter 1

Introduction & Protocol Novelty

1.1 Background and Structural Bottlenecks

In current macroeconomic clearing landscapes, supply chains face severe structural bottlenecks due to payment delays and asynchronous balance sheet settlement.

1.2 The FluidCluster Architecture Paradigm

The FluidCluster protocol revolutionizes decentralized clearing by treating liquidity optimization as a competitive public commodity. Rather than trusting centralized clearing houses, the framework implements a decentralized network of cryptographic state changes where zero data leakage occurs.

1.3 Core Security Invariants

Financial state modifications are governed by absolute mathematical invariants. Total system capital can neither be artificially inflated nor arbitrarily destroyed during net settlement transitions:

$$\sum_{i=1}^{n} \text{Position}_{\text{pre}}(i) = \sum_{i=1}^{n} \text{Position}_{\text{post}}(i) \tag{1.1}$$

Chapter 2

The Blind Execution Substrate (fhEVM)

2.1 Torus Fully Homomorphic Encryption (TFHE) Foundations

The fundamental layer relies on Torus Fully Homomorphic Encryption (TFHE), allowing discrete integer operations over encrypted ring structures without decrypting state data.

2.2 Branchless Novation Logic and Side-Channel Mitigation

Conditional operations ($if/else$ branches) within standard EVM architectures introduce timing profiles that leak corporate credit limits. FluidCluster mandates uniform arithmetic execution times utilizing select-gates:

$$\text{Limit}_{\text{new}} = \text{TFHE.select}(\text{Condition}, \text{Limit}_{\text{sub}}, \text{Limit}_{\text{old}}) \tag{2.1}$$

Chapter 3

Connected Collateralization & Zero-Knowledge Invoices

3.1 ATCUD Anchoring and Document Provenance

To avoid arbitrary credit statements, documents from client ERP layers map directly to strict unique cryptographic structures.

3.2 Invoice Nullifiers and Anti-Double-Financing Circuits

Double-financing vectors are systematically eliminated through deterministic hash generation. If an invoice code is reused inside a clearing epoch, it generates a collision signature that voids the process without revealing identity or volume metrics.

Chapter 4

The Proof of Minimization Mining Protocol

4.1 NP-Hard Graph Complexity Dynamics

Minimizing generalized flows with binary activation constraints and rigid credit capacities forms a fundamentally non-deterministic polynomial-time hard puzzle.

4.2 Timelock Auctions and Solver Competition Loops

FluidCluster leverages this complexity by offering public commissions to any network miner providing the highest systemic optimization factor under fixed time horizons.

Chapter 5

Conclusion & Future R&D: An Open Call to Innovation

5.1 Summary of Protocol Milestones

The foundational architecture provides a secure, quantum-resistant, and unconditionally private settlement ledger for modern finance.

5.2 The Open Hardware and Algorithmic Challenge

Call to Global Experts: We explicitly issue a global challenge to academic institutions, hardware architecture consortia, and mathematical optimization pioneers to expand the boundaries of the FluidCluster execution pool:

- **Algorithmic Optimization Solvers:** Designing specialized heuristics capable of beating current commercial tools (Gurobi, CPLEX, HiGHS) in solving hyper-sparse financial graphs under sub-minute block timelocks.
- **Accelerated Compute Silicon:** Developing ultra-efficient ASICs, customized FPGA fabrics, and specialized GPU acceleration kernels tailored explicitly for homomorphic polynomial matrices.
- **Photonic Integrated Circuits (PIC):**** Engineering high-density Mach-Zehnder Interferometer (MZI) nanophotonic arrays capable of evaluating discrete polynomial convolutions at the speed of light (c).

Appendix A

Reference Implementations

A.1 Foundational Invariant Simulation (Python)

```python
import numpy as np

def simulate_fluidcluster_invariants():
    G_private = np.array([
        [0.0, 5000.0, 2000.0],
        [1000.0, 0.0, 3000.0],
        [4000.0, 500.0, 0.0]
    ])

    E_shared = np.array([
        [0.0, 1500.0, -1500.0],
        [-500.0, 0.0, 500.0],
        [500.0, -1500.0, 1000.0]
    ])

    W_published = np.array([
        [0.0, 0.2, 0.5],
        [0.1, 0.0, 0.3],
        [0.4, 0.8, 0.0]
    ])

    G_camouflaged = G_private + E_shared
    N_recovered = G_camouflaged * W_published

    print("=== FLUIDCLUSTER CRYPTO-CORE VALIDATION ===")
    print("Net Settlement Matrix (W-Space):\n", np.round(N_recovered, 2)
    )

    initial_balances = np.sum(G_private, axis=0) - np.sum(G_private,
    axis=1)
    recovered_balances = np.sum(N_recovered, axis=0) - np.sum(
    N_recovered, axis=1)

    assert np.allclose(initial_balances, recovered_balances, atol=1e-5)
    print("\n[INVARIANT CONFIRMED]: Zero Capital Distortion Met.")

if __name__ == "__main__":
    simulate_fluidcluster_invariants()
```

Listing A.1: Python simulation demonstrating macro-preservation and Hadamard masking matrix functions.

A.2 Smart Contract Core Architecture (Solidity)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.24;

import "fhevm/lib/TFHE.sol";

contract FluidClusterRegistry {

    mapping(bytes32 => bool) private _spentInvoiceNullifiers;
    mapping(address => mapping(address => euint32)) private
   _encryptedLimits;

    event CreditLimitProvisioned(address indexed debtor, address indexed
    creditor);
    event InvoiceAnnihilated(bytes32 indexed nullifierHash);

    function provisionCreditLimit(address creditor, bytes calldata
   encryptedLimitBytes) external {
euint32 inboundLimit = TFHE.asEuint32(encryptedLimitBytes);
_encryptedLimits[msg.sender][creditor] = inboundLimit;
emit CreditLimitProvisioned(msg.sender, creditor);
}
function registerInvoiceCommitment(bytes32 invoiceNullifier) external {
require(!_spentInvoiceNullifiers[invoiceNullifier], "FluidCluster:
   DOUBLE_SPEND");
_spentInvoiceNullifiers[invoiceNullifier] = true;
emit InvoiceAnnihilated(invoiceNullifier);
}
function evaluateNovationBranchless(
address debtor,
address creditor,
euint32 targetAmount
) internal returns (ebool) {
euint32 availableCredit = _encryptedLimits[debtor][creditor];
ebool isConditionMet = TFHE.le(targetAmount, availableCredit);
euint32 updatedLimit = TFHE.select(isConditionMet, TFHE.sub(
   availableCredit, targetAmount), availableCredit);
_encryptedLimits[debtor][creditor] = updatedLimit;
return isConditionMet;
}
}
```

Listing A.2: Solidity base ledger registry managing encrypted parameters via TFHE branchless paradigms.

A.3 Mixed-Integer Programming Solver Core (Python)

```
import scipy.optimize as opt
import numpy as np
def run_fluidcluster_solver_block(debt_matrix, credit_limits):
n = debt_matrix.shape[0]
num_vars = n * n
c = np.ones(num_vars)
bounds = []
for i in range(n):
for j in range(n):
if i == j:
bounds.append((0, 0))
else:
bounds.append((0, credit_limits[i, j]))
A_eq = []
b_eq = []
initial_net_balances = np.sum(debt_matrix, axis=0) - np.sum(debt_matrix,
    axis=1)
for node in range(n):
row = np.zeros(num_vars)
for i in range(n):
row[i * n + node] = 1
for j in range(n):
row[node * n + j] = -1
A_eq.append(row)
b_eq.append(initial_net_balances[node])
res = opt.linprog(c, bounds=bounds, A_eq=A_eq, b_eq=b_eq, method='highs'
   )
if res.success:
optimized_matrix = res.x.reshape((n, n))
liquidity_released = np.sum(debt_matrix) - np.sum(optimized_matrix)
return np.round(optimized_matrix, 2), np.round(liquidity_released, 2)
else:
raise ValueError("Infeasible credit constraints.")
```

Listing A.3: High-efficiency localized linear optimization algorithm for competitive miners.

A.4 fhEVM Multi-Flow Verification Infrastructure (Solidity)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.24;
import "fhevm/lib/TFHE.sol";
contract FluidClusterValidationEngine {
address public creditRegistryAddress;
struct OptimizedFlowInput {
address debtor;
address creditor;
bytes encryptedFlowBytes;
}
constructor(address _registry) {
creditRegistryAddress = _registry;
}
function verifySolverSubmissionBranchless(
OptimizedFlowInput[] calldata proposedFlows,
euint32[] calldata currentEncryptedLimits
) external returns (bytes32) {
require(proposedFlows.length == currentEncryptedLimits.length, "
    FluidCluster: DIM_MISMATCH");
euint32 encryptedTotalDebt = TFHE.asEuint32(0);
ebool globalValidationVector = TFHE.asEbool(true);
for (uint256 i = 0; i < proposedFlows.length; i++) {
euint32 proposedAmount = TFHE.asEuint32(proposedFlows[i].
    encryptedFlowBytes);
euint32 strictLimit = currentEncryptedLimits[i];
ebool isFlowValid = TFHE.le(proposedAmount, strictLimit);
globalValidationVector = TFHE.and(globalValidationVector, isFlowValid);
encryptedTotalDebt = TFHE.add(encryptedTotalDebt, proposedAmount);
}
euint32 finalAuditedDebt = TFHE.select(globalValidationVector,
    encryptedTotalDebt, TFHE.asEuint32(0xffffffff));
return keccak256(abi.encode(finalAuditedDebt));
}
}
```

Listing A.4: Validation Smart Contract engine processing solver array transformations.

A.5 Zero-Knowledge Provenance Circuit Design (Circom)

```
pragma circom 2.1.6;
include "circomlib/circuits/sha256/sha256.circom";
template FluidClusterInvoiceNullifier() {
signal input atcudStringBits[256];
signal input erpPrivateKeyBits[256];
signal input invoiceAmount;
signal input expectedAmountCommitment;
signal output invoiceNullifier[256];
component hasher = Sha256(512);
for (int i = 0; i < 256; i++) {
hasher.in[i] <== atcudStringBits[i];
}
for (int i = 0; i < 256; i++) {
hasher.in[i + 256] <== erpPrivateKeyBits[i];
}
for (int i = 0; i < 256; i++) {
invoiceNullifier[i] <== hasher.out[i];
}
signal amountCheck;
amountCheck <== invoiceAmount * 1;
}
component main {public [expectedAmountCommitment]} =
   FluidClusterInvoiceNullifier();
```

Listing A.5: Circom file implementing private data extraction parameters and anchor verification hashes.

A.6 Mach-Zehnder Electro-Optical Modulator Simulation (Python)

```python
import numpy as np
def simulate_mzi_optical_gate(input_amplitude_1, input_amplitude_2,
    theta, phi):
E_in = np.array([[input_amplitude_1], [input_amplitude_2]])
BS = (1 / np.sqrt(2)) * np.array([[1, 1j], [1j, 1]])
PS = np.array([[np.exp(1j * theta), 0], [0, np.exp(1j * phi)]])
E_interm = np.dot(BS, E_in)
E_phased = np.dot(PS, E_interm)
E_out = np.dot(BS, E_phased)
return np.round((np.abs(E_out) ** 2).flatten(), 4)
```

Listing A.6: Mathematical representation mapping optimization constraints into laser phase modulations.

Bibliography

[1] Zama fhEVM Documentation Team, *ERC-7984: Confidential Smart Contract Token Infrastructure Standards via Fully Homomorphic Encryption*, Technical Specification Report, 2025.

[2] Chillotti, I., Gama, N., Georgieva, M., Malozemoff, S., *TFHE: Fast Fully Homomorphic Encryption Over the Torus*, Journal of Cryptology, Vol. 33, pp. 34-91, 2020.

[3] National Institute of Standards and Technology (NIST), *FIPS PUB 203: Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) FIPS PUB 204: Module-Lattice-Based Digital Signature Algorithm (ML-DSA) Specifications*, Post-Quantum Cryptography Standardization Project, 2024.

[4] Lyubashevsky, V., Peikert, C., Regev, O., *On Ideal Lattices and Learning with Errors over Rings*, Journal of the ACM, Vol. 60, No. 6, Article 43, 2013.

[5] CoW Protocol Research Group, *Decentralized Batch Auctions and Intent-Centric Solver Architecture Performance Paradigms*, DeFi Protocol Review, 2024.

[6] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1993.