Encrypted Trade Battle Arena

> Privacy-Preserving DeFi Strategy Marketplace powered by EigenLayer AVS & Fhenix CoFHE

CipherTradeArena is a decentralized strategy competition platform where traders submit **fully encrypted DeFi strategies** that compete for capital allocation based on simulated performance. Built on EigenLayer's AVS framework with Fhenix's CoFHE (Confidential Fully Homomorphic Encryption), it ensures complete strategy privacy until execution.

📋 Table of Contents

🎯 Overview

The Problem

Traditional algo trading competitions reveal strategies before execution, leading to:

- **Front-running** of winning strategies
- **Strategy theft** and copycatting
- **MEV extraction** by observers
- **Limited alpha** for participants

Our Solution

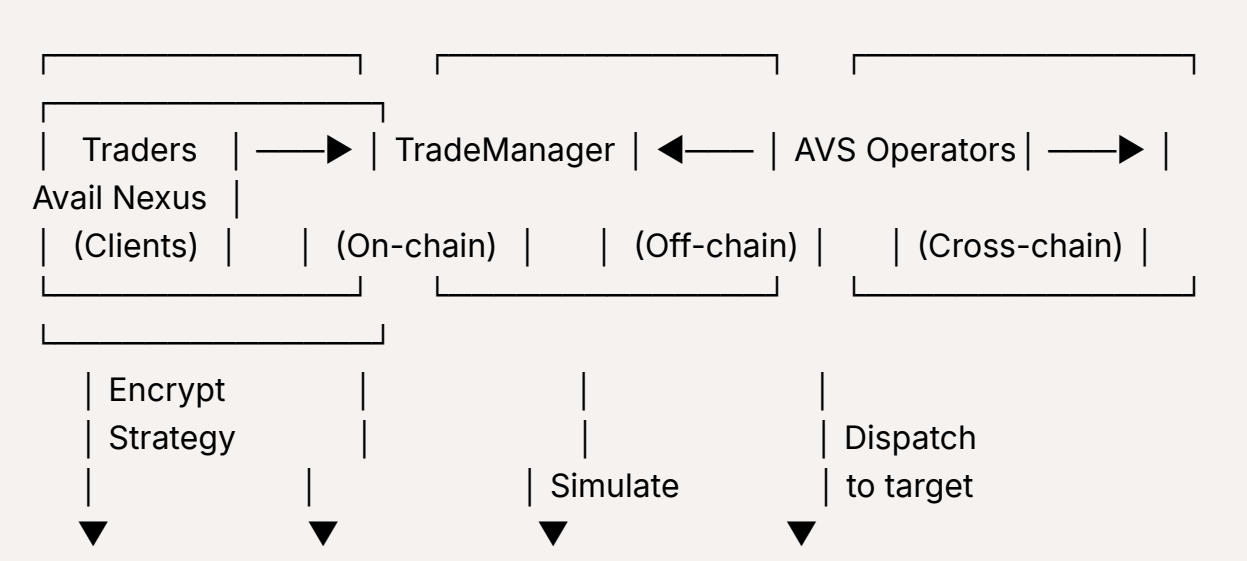
CipherTradeArena uses **Fully Homomorphic Encryption** (FHE) via Fhenix CoFHE coprocessors to enable:

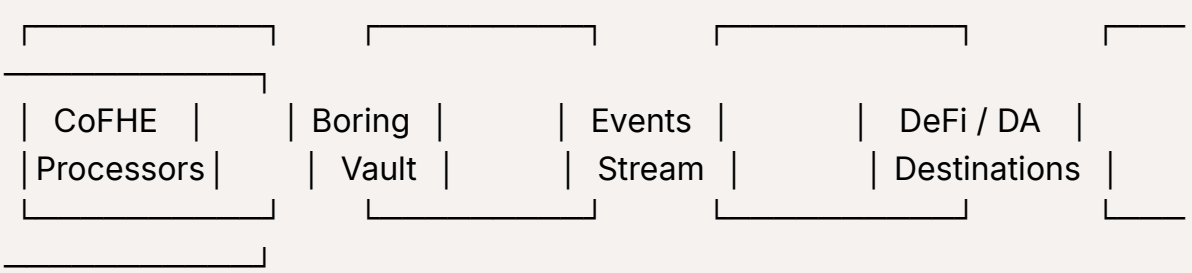
- ✅ **Blind Strategy Submission** - Traders encrypt strategies client-side using FHE coprocessors
- ✅ **Private Simulation** - AVS operators decrypt and simulate off-chain
- ✅ **Encrypted APY Reporting** - Performance metrics stay confidential
- ✅ **Privacy-Preserving Execution** - Aggregated deployment hides individual strategies

---

🏗️ Architecture

System Components





Core Contracts

Contract	Description	Base Sepolia Address
TradeManager	Main strategy competition contract	0xea93e65fefaF6EB63f32BF2Da6739d2Fb2373EE4
SimpleBoringVault	Strategy execution vault	0x67B492e98AF5bb869dB56d1eAe32D410a7dA0324
ECDSAStakeRegistry	Operator stake management	0x8fC8CFB7f7362E44E472c690A6e025B80E406458

🔄 How It Works

Phase 1: Epoch Initialization

```
%%{init: {'theme':'dark', 'themeVariables': { 'primaryColor':'#03dac6', 'primaryTextColor':'#fff', 'primaryBorderColor':'#00a896', 'lineColor':'#03dac6', 'secondaryColor':'#bb86fc', 'tertiaryColor':'#3700b3', 'background':'#121212', 'mainBkg':'#1f1f1f', 'secondBkg':'#2d2d2d', 'labelBackground':'#2d2d2d', 'actorBkg':'#424242', 'actorBorder':'#03dac6', 'actorTextColor':'#fff', 'signalColor':'#03dac6', 'signalTextColor':'#fff'}}}%%
sequenceDiagram
    participant Admin
    participant TradeManager
    participant CoFHE

    rect rgb(30, 60, 80)
        Note over Admin,TradeManager: PHASE 1: Epoch Initialization
```

```
    Admin->>CoFHE: Encrypt simulation window
    Note over Admin,CoFHE: Start: 7 days ago<br/>End: 1 day ago
    Admin->>TradeManager: startEpoch(encSimStart, encSimEnd, weights)
    TradeManager-->>Admin: Epoch started
  end
```

Phase 2: Strategy Submission

```
%%{init: {'theme':'dark', 'themeVariables': { 'primaryColor':'#4ecdc4', 'primaryTextColor':'#fff', 'primaryBorderColor':'#45b7d1', 'lineColor':'#4ecdc4', 'secondaryColor':'#ff6b6b', 'tertiaryColor':'#3700b3', 'background':'#121212', 'mainBkg':'#1f1f1f', 'secondBkg':'#2d2d2d', 'labelBackground':'#2d2d2d', 'actorBkg':'#424242', 'actorBorder':'#4ecdc4', 'actorTextColor':'#fff', 'signalColor':'#4ecdc4', 'signalTextColor':'#fff'}}}%%
sequenceDiagram
  participant Trader
  participant CoFHE
  participant TradeManager
  participant AVS
  participant "Avail Nexus"

  rect rgb(60, 80, 40)
    Note over Trader,TradeManager: PHASE 2: Strategy Submission
    Trader->>CoFHE: Encrypt strategy nodes
    Note over Trader,CoFHE: Encoder, Target,<br/>Selector, Args
    Trader->>TradeManager: submitEncryptedStrategy(targetChainId)
    TradeManager->>AVS: Grant decryption permission
    TradeManager-->>Trader: Strategy submitted
    TradeManager-->>"Avail Nexus": Emit StrategySubmitted metadata
    AVS->>AVS: Listen for StrategySubmitted event
    "Avail Nexus"-->>AVS: Surface target chain routing
  end
```

Phase 3: AVS Processing

```
%%{init: {'theme':'dark', 'themeVariables': { 'primaryColor':'#bb86fc', 'primaryTextColor':'#fff', 'primaryBorderColor':'#9f6bff', 'lineColor':'#bb86fc', 'secondaryColor':'#03dac6', 'tertiaryColor':'#3700b3', 'background':'#121212', 'mainBkg':'#1f1f1f', 'secondBkg':'#2d2d2d', 'labelBackground':'#2d2d2d', 'actorBkg':'#424242', 'actorBorder':'#bb86fc', 'actorTextColor':'#fff', 'signalColor':'#bb86fc', 'signalTextColor':'#fff'}}}%%
sequenceDiagram
    participant AVS
    participant CoFHE
    participant TradeManager
    participant "Avail Nexus"

    rect rgb(40, 80, 60)
        Note over AVS,TradeManager: PHASE 3: AVS Processing (Off-chain)
        AVS->>TradeManager: Fetch encrypted strategy
        AVS->>CoFHE: Batch decrypt nodes
        CoFHE-->>AVS: Decrypted values
        AVS->>"Avail Nexus": Fetch cross-chain routing context
        "Avail Nexus"-->>AVS: Unified balances & target chain info
        AVS->>AVS: Simulate on 100k notional
        AVS->>AVS: Calculate APY
        AVS->>CoFHE: Encrypt APY
        AVS->>TradeManager: reportEncryptedAPY()
    end
```

Phase 4: Epoch Closure & Finalization

```
%%{init: {'theme':'dark', 'themeVariables': { 'primaryColor':'#ff6b6b', 'primaryTextColor':'#fff', 'primaryBorderColor':'#ee5a6f', 'lineColor':'#ff6b6b', 'secondaryColor':'#4ecdc4', 'tertiaryColor':'#3700b3', 'background':'#121212', 'mainBkg':'#1f1f1f', 'secondBkg':'#2d2d2d', 'labelBackground':'#2d2d2d', 'actorBkg':'#424242', 'actorBorder':'#ff6b6b', 'actorTextColor':'#fff', 'signalColor':'#ff6b6b', 'signalTextColor':'#fff'}}}%%
sequenceDiagram
```

```
    participant Admin
    participant TradeManager
    participant AVS
    participant "Avail Nexus"

    rect rgb(60, 40, 80)
        Note over Admin,AVS: PHASE 4: Epoch Closure & Finalization
        Admin->>TradeManager: closeEpoch()
        TradeManager->>TradeManager: Request APY decryption
        AVS->>AVS: Listen for EpochClosed event
        AVS->>AVS: Fetch all APYs
        AVS->>AVS: Rank by APY
        AVS->>AVS: Select top winners
        AVS->>TradeManager: finalizeEpoch(winners, APYs)
        TradeManager-->>"Avail Nexus": Emit FinalizedEpoch metadata
        TradeManager-->>Admin: Winners selected
        "Avail Nexus"-->>AVS: Confirm cross-chain dispatch windows
    end
```

Phase 5: Strategy Execution

```
%%{init: {'theme':'dark', 'themeVariables': { 'primaryColor':'#ffd93d', 'primaryTextColor':'#000', 'primaryBorderColor':'#f4c430', 'lineColor':'#ffd93d', 'secondaryColor':'#6bcf7f', 'tertiaryColor':'#3700b3', 'background':'#121212', 'mainBkg':'#1f1f1f', 'secondBkg':'#2d2d2d', 'labelBackground':'#2d2d2d', 'actorBkg':'#424242', 'actorBorder':'#ffd93d', 'actorTextColor':'#fff', 'signalColor':'#ffd93d', 'signalTextColor':'#fff'}}}%%
sequenceDiagram
    participant AVS
    participant TradeManager
    participant "Avail Nexus"
    participant BoringVault
    participant "Base DeFi"
    participant "Target Chains Defi"
```

```
    rect rgb(30, 80, 50)
        Note over AVS: PHASE 5: Strategy Execution
        AVS->>AVS: Get winner strategies from events
        AVS->>AVS: Aggregate calls
        AVS->>AVS: Reconstruct calldata
        AVS->>AVS: Sign for consensus
        AVS->>TradeManager: executeEpochTopStrategiesAggregated() (base c
hain)
        TradeManager->>BoringVault: Execute aggregated strategy
        BoringVault->>"Base DeFi": Deploy capital
        AVS->>"Avail Nexus": Dispatch cross-chain bundles
        "Avail Nexus"->>"Target Chains Defi": Relay aggregated calldata
    end
```

🤖 Operator Setup

1. Register as Operator

```
# Start operator (auto-registers on first run)
ts-node operator/index.ts
```

What happens:

- Registers with EigenLayer DelegationManager
- Registers with ECDSAStakeRegistry
- Registers with TradeManager
- Starts monitoring for StrategySubmitted events

2. Operator Workflow

```
┌──────────────────────────────────────────────────────────────────┐
│  Operator Lifecycle                    │
├──────────────────────────────────────────────────────────────────┤
│                                  │
```

```
│  1. Start operator                          │
│     ts-node operator/index.ts               │
│                                             │
│  2. Process strategies (automatic)          │
│     - Decrypt encrypted strategies          │
│     - Simulate on historical data           │
│     - Calculate APY                         │
│     - Report encrypted APY to chain         │
│                                             │
│  3. Close & Finalize epoch (manual)         │
│     ts-node operator/closeAndFinalizeEpoch.ts 1 │
│     - Closes epoch after deadline           │
│     - Reads APYs from events                │
│     - Ranks strategies                      │
│     - Submits winners to chain              │
│                                             │
│  4. Execute strategies (manual)             │
│     ts-node operator/executeAggregatedStrategies.ts 1 │
│     - Aggregates winner strategies          │
│     - Reconstructs calldata                 │
│     - Executes via BoringVault              │
│                                             │
└─────────────────────────────────────────────────────────────┘
```

👤 Trader Guide

1. Start an Epoch (Admin Only)

```
ts-node operator/setupAndStartEpoch.ts
```

2. Submit Encrypted Strategy

```
ts-node operator/createEncryptedStrategyInputs.ts
```

Example Strategy (2 nodes):

```
// Node 1: Supply 1000 USDC to Aave
{
  target: AavePool,
  function: "supply",
  args: [USDC, 1000e6, trader, 0]
}

// Node 2: Borrow 500 USDT from Aave
{
  target: AavePool,
  function: "borrow",
  args: [USDT, 500e6, 2, 0, trader]
}
```

3. Wait for Results

After epoch ends:

- Operators simulate your strategy
- APY is calculated and reported (encrypted)
- You can view your own APY: `tradeManager.viewMyAPY(epochNumber)`

4. Winners Announced

Top strategies by APY receive capital allocation:

- **1st Place**: 50% of capital
- **2nd Place**: 30% of capital
- **3rd Place**: 20% of capital

Avail Nexus SDK Integration

The operator integrates the in-progress `@avail-project/nexus-core` SDK to fetch Nexus data directly from a Node.js runtime. Because the upstream SDK expects an

injected browser provider, we expose a minimal EIP-1193 wrapper around our `ethers.Wallet` before calling `initializeNexus`, keeping nonce management and signing fully under operator control.

- `operator/nexus/eip1193Provider.ts` defines `WalletEip1193Provider`, translating core `eth_*` RPC methods into wallet actions while forwarding everything else to the connected `JsonRpcProvider`.
- `operator/nexus/runtime.ts` provides a lightweight `window` shim (crypto, timers, optional WebSocket) so the SDK can bootstrap without a browser environment.
- `operator/nexus/sdk.ts` wires the shim and provider together, yielding a reusable `NexusSDK` instance that the off-chain workflow can use alongside EigenLayer calls.

This setup lets us bypass the browser-only requirement from the Avail docs and run the Nexus SDK inside the AVS operator process with full EIP-1193 compatibility.

Supported Protocols (Testnet)

- **Aave V3**: Lending & borrowing
- **Pendle**: Yield trading
- **Morpho**: Optimized lending
- **Mock Protocols**: For testing

Privacy Guarantees

1. **Strategy Privacy**: Encrypted until operator decryption grant
2. **APY Privacy**: Encrypted for individual traders, only finalized winners revealed
3. **Execution Privacy**: Aggregated calls hide individual contributions
4. **Simulation Window Privacy**: Encrypted to prevent overfitting

---