Obscura: A Protocol for Verifiable, Privacy-Preserving Reputation in Decentralized Finance

The Obscura Team

October 9, 2025

Abstract

The proliferation of decentralized finance (DeFi) has exposed a critical infrastructure gap: the absence of a credibly neutral, privacy-preserving, and verifiable reputation system. Existing solutions are fragmented, rely on centralized intermediaries, or lack the cryptographic guarantees necessary for high-value financial applications. This paper introduces Obscura, a novel protocol designed to establish the definitive Verifiable Reputation Layer for the decentralized economy. Obscura synthesizes three core technologies into a cohesive architecture: a Byzantine Fault Tolerant (BFT) Decentralized Oracle Network (DON) for tamper-resistant data ingestion, Confidential Computing via Trusted Execution Environments (TEEs) for secure management of sensitive credentials, and Zero-Knowledge (ZK) proofs for the private verification of trading performance. By integrating these primitives with a robust cryptoeconomic model, Obscura enables traders to build an immutable, on-chain track record from their activities across any centralized or decentralized exchange, without compromising privacy. All verification and settlement occur on a dedicated Horizen Layer 3, creating a new paradigm for trust and accountability in DeFi.

1 Introduction: The Reputation Deficit in DeFi

The foundational promise of decentralized finance is the creation of a financial system that is open, permissionless, and free from the opacity of traditional intermediaries. However, the absence of a reliable identity and reputation layer has led to significant capital inefficiencies and systemic risks. Without a mechanism to verify the track record and capabilities of participants, the ecosystem is forced to rely on over-collateralization as its primary risk management tool, limiting capital access and creating a high barrier to entry.

Initial attempts to solve this problem have proven inadequate, falling prey to one of two fundamental flaws:

- **The Centralized Oracle Problem:** Many systems rely on a single, trusted entity to ingest trading data from external sources.[1] While a Zero-Knowledge proof can verify the correctness of a computation on this data, it cannot attest to the integrity of the data itself. This creates a "garbage in, garbage out" scenario, where the cryptographic guarantees of the ZK layer are rendered moot by a centralized point of failure and trust.[2] Such a system is not truly verifiable; it is merely a verification of a central party's claims.

- **Lack of Cryptoeconomic Security:** Systems that rely solely on technical security measures without underlying economic incentives are inherently fragile. Without a mechanism to economically connect data providers to their claims - a concept often termed "skin in the game" there is no rational deterrent against malicious behavior, especially as the value secured by the protocol increases. [4]

Obscura is architected from first principles to solve these challenges. It replaces centralized trust with a distributed, cryptoeconomically-secured consensus and leverages cutting-edge cryptography to ensure that verifiable reputation can be built without sacrificing user privacy.

1.1 Architectural Tenets

The Obscura protocol is founded on four inviolable principles that guide the design of every component:

1. **Privacy by Design:** User privacy is not an afterthought but a core architectural requirement. Sensitive data, such as API credentials, is handled exclusively within hardware-isolated Trusted Execution Environments (TEEs).[6] Performance claims are verified on-chain using Zero-Knowledge proofs, allowing a trader to prove their track record without revealing the specific trades that constitute it.[8]

2. **Decentralized Trust:** The protocol's integrity is not derived from trust in the Obscura entity, but from a BFT consensus among a network of independent, economically-incentivized actors.[9] This principle is embodied by the Sentinel DON, which ensures that all data entering the system is the result of a distributed consensus, making it resilient to censorship and manipulation.[10]

3. **Confidential Execution:** All off-chain operations involving secrets are performed within a TEE. The integrity of this environment is guaranteed by a process of

cryptographic attestation, which programmatically proves that only the specific, audited version of our code can access sensitive user data, creating an unbreakable link between verified code and its execution.[7]

4. **User-Centric Abstraction:** The profound complexity of the underlying cryptographic and distributed systems is completely abstracted from the end-user. The protocol provides a seamless experience for both signal providers and followers, masking the sophisticated machinery that guarantees the system's security and integrity [1].

2 Protocol Architecture

The Obscura protocol is a hybrid system composed of interoperating off-chain services and on-chain smart contracts. The off-chain layer handles high-frequency data ingestion and confidential computation, while the on-chain layer on the Horizen L3 serves as the immutable settlement and truth layer.

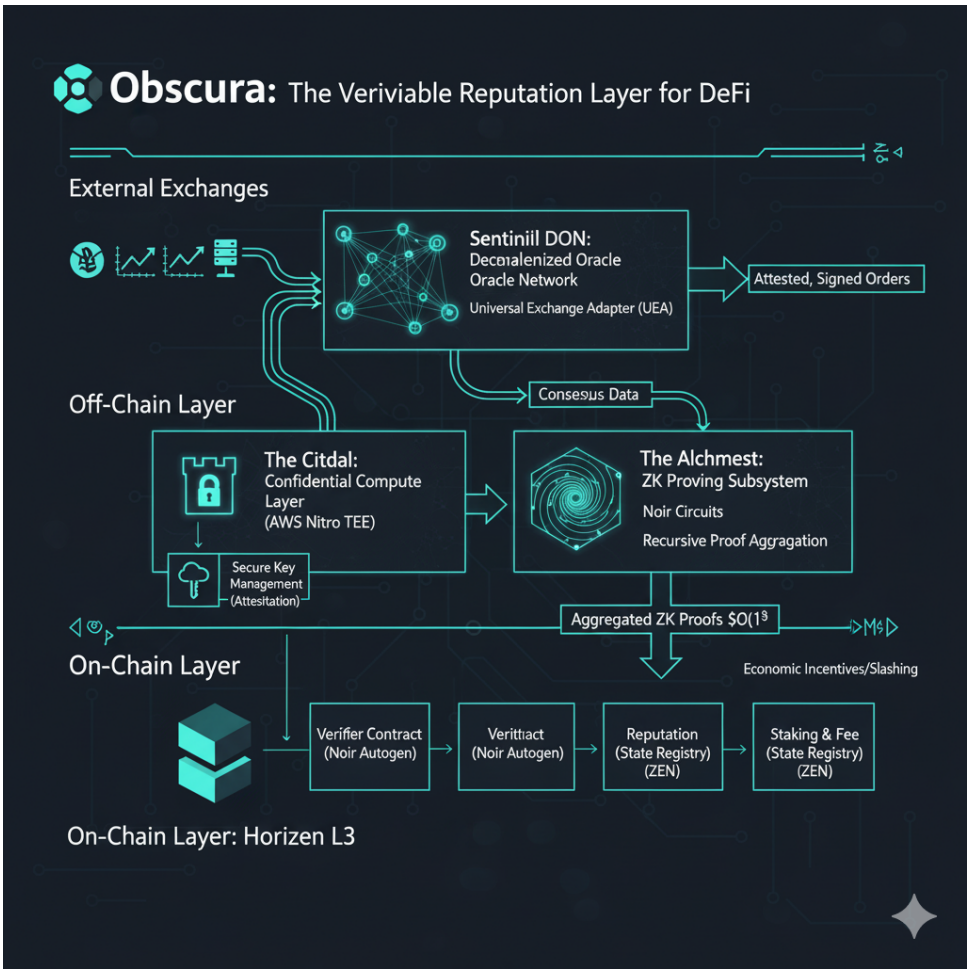

Figure 1: Obscura High-Level System Architecture.

2.1 The Sentinel DON: Decentralized Data Ingestion

The Sentinel DON is the gateway for all external data into the Obscura ecosystem, replacing the fragile, centralized oracle model with a robust, BFT network.

2.1.1 Consensus Mechanism

The network is composed of n independent node operators who collectively source, validate, and report trade data. The consensus protocol is modeled after Chainlink's Off-Chain Reporting (OCR) protocol, which is highly efficient and resilient.[10] The process for each reporting round is as follows:

1. **Leader Election:** A leader is pseudo-randomly elected for a given epoch.
2. **Observation Request:** The leader broadcasts a request for observations to all follower nodes.
3. **Distributed Observation:** Each node independently fetches trade data from the relevant exchanges via a Universal Exchange Adapter (UEA). The UEA normalizes data from disparate CEX and DEX sources into a canonical format.
4. **Signed Observations:** Each follower node cryptographically signs its observation and transmits it to the leader.
5. **Report Generation:** The leader waits to receive a quorum of at least $2f+1$ signed observations (where f is the number of faulty nodes the network can tolerate).[10] It then assembles these observations into a single report, which contains the consensus view of the trade data (e.g., a medianized value for price).
6. **Attestation:** The leader circulates the final report to the followers, who validate it and return their signatures. Once the leader has a quorum of signatures on the final report, it becomes an attested report.
7. **On-Chain Transmission:** A single node transmits this highly compressed, attested report to the Horizen L3, drastically reducing on-chain transaction costs compared to a naive model where each node submits its own data.[14]

2.1.2 Cryptoeconomic Security

The integrity of the Sentinel DON is secured by a robust cryptoeconomic model built on staking and slashing. To participate, node operators must lock a significant amount of Horizen's native ZEN token in the on-chain `StakingContract`. This stake acts as a financial bond guaranteeing the honesty of their data reporting.[15]

Malicious or negligent behavior is disincentivized through slashing. Let v_i be the value reported by node i and $\tilde{V}$ be the consensus median of all reported values in a round. A node's deviation score δ_i is calculated. Slashing is triggered if a node's cumulative deviation score over a time window T exceeds a governance-defined threshold Θ:

$$\sum_{t \in T} \delta_i(t) > \Theta$$

Upon a successful slashing event, a portion of the offending node's staked ZEN is seized, economically punishing the bad actor and securing the network.[16]

2.2 The Citadel: Confidential Compute Layer

The Citadel is the fortified core of the off-chain system, responsible for handling all sensitive user credentials. It is built upon AWS Nitro Enclaves, a TEE that provides hardware-level isolation for code and data.[6]

2.2.1 Hardware-Enforced Isolation

A Nitro Enclave is a partitioned virtual machine with no persistent storage, no external networking, and no interactive access.[7] The Nitro Hypervisor creates a hardware-enforced boundary that prevents any process on the parent EC2 instance—including root users or administrators—from accessing the enclave's memory or CPU state.[7] Communication occurs exclusively over a secure, in-memory `vsock` channel.[18]

2.2.2 Cryptographic Attestation and Key Management

The security of the Citadel hinges on cryptographic attestation. This process allows the enclave to prove its identity and integrity to an external service, in this case, AWS Key Management Service (KMS). The flow is as follows:

1. Upon boot, the enclave requests a signed attestation document from the Nitro Hypervisor. This document contains a set of cryptographic hashes of the enclave's software and configuration, known as Platform Configuration Registers (PCRs).[6]

2. The enclave sends this signed document to AWS KMS to request the decryption of a master Data Encryption Key (DEK).

3. AWS KMS verifies the signature on the document and, critically, compares the PCRs within the document to a pre-configured access policy. The policy is configured to only permit decryption if the PCRs exactly match the values of the audited and approved enclave image.[21]

4. If verification succeeds, KMS returns the plaintext DEK to the enclave over a secure channel.

This mechanism guarantees that only the specific, untampered version of our signing code can ever gain access to the keys required to manage user API credentials. Any modification to the code, however minor, will alter the PCR hash, causing attestation to fail and locking the enclave out of its secrets.[20]

2.3 The Alchemist: Zero-Knowledge Proving Subsystem

The Alchemist service transforms the raw, consensus-verified trade data from the Sentinel DON into private, verifiable proofs of performance.

2.3.1 Circuit Design and Optimization

We have selected the Noir programming language for its developer-friendly, Rust-like syntax, which abstracts away much of the low-level complexity of ZK circuit design.[8] The underlying proving system is UltraHonk, which avoids the need for a per-circuit trusted setup.[1] The trade performance circuit is meticulously optimized to minimize its

"gate count," which directly impacts proving time and cost. Key optimization strategies include:

- **Fixed-Point Arithmetic:** All financial calculations are performed using scaled integers to avoid the inefficiency of floating-point arithmetic in ZK circuits.[24]
- **Arithmetic Favoritism:** The circuit logic is designed to maximize the use of native field arithmetic operations (addition and multiplication), while minimizing more costly operations like comparisons and bitwise logic.[24]
- **Unconstrained Functions:** Helper functions that do not need to be proven within the circuit (e.g., data formatting) are executed as unconstrained code by the prover, performing complex pre-computation outside the expensive proving environment.[24]

2.3.2 Recursive Proof Aggregation

Submitting a ZK proof for every individual trade is computationally and economically prohibitive. The Alchemist service implements a proof aggregation strategy to overcome this limitation. This process uses recursion to combine a large number of individual proofs into a single, compact proof that can be verified on-chain with constant cost, $O(1)$, regardless of the number of underlying trades.[25]

Let Π_i be the base proof for an individual trade. A specialized aggregator circuit, A, takes two proofs as input and outputs a single new proof that attests to their validity. This is applied recursively in a binary tree structure:

$$\Pi_{agg} = A(A(\Pi_1, \Pi_2), A(\Pi_3, \Pi_4), \dots)$$

This recursive composition allows the system to compress thousands of individual trade verifications into a single, efficient on-chain transaction, making the protocol scalable to even the most active traders.[25]

2.4 On-Chain Infrastructure (Horizen L3)

The on-chain component of Obscura is a set of modular, interoperable smart contracts deployed on the Horizen L3. These contracts, written in Solidity and hardened with OpenZeppelin libraries, form the protocol's immutable truth layer.

- **Verifier Contract:** A highly optimized, minimal contract that is auto-generated by the Noir toolchain. Its sole function is to verify the aggregated ZK proofs submitted by the Alchemist service.
- **Reputation Contract:** The central on-chain registry of trader reputations. It maintains the state of each trader's verified performance metrics. The state can only be modified by a successful call to its `updateReputation` function, which requires a valid ZK proof that has been successfully checked by the `VerifierContract`. The reputation update function, $\mathcal{R}$, can be modeled as:

  $$R_{t+1} = \mathcal{R}(R_t, \mathcal{P}_{agg}, \mathcal{I}_{pub})$$

  where R_t is the reputation at time t, and ΔR is the state transition function that computes the reputation delta based on the aggregated proof $\mathcal{P}_{agg}$ and its associated public inputs $\mathcal{I}_{pub}$.

- **Staking & Fee Settlement Contract:** This contract manages all economic activity. It holds the ZEN tokens staked by Sentinel DON operators, enforces slashing penalties as dictated by governance, and manages the collection and distribution of performance fees between signal providers and their followers.

3 System Workflows

3.1 Nominal Operation: From Trade to Proof

1. **Execution & Indexing:** A trader executes a trade on an external exchange. The Sentinel DON nodes independently detect and reach a BFT consensus on the trade's details.

2. **Proof Generation & Aggregation:** The consensus-verified data is consumed by the Alchemist service, which generates a base ZK proof. This proof is then recursively combined with others into a single aggregated proof.

3. **On-Chain Verification:** The Alchemist submits the single aggregated proof to the `ReputationContract` on the Horizen L3. The contract verifies the proof via the `VerifierContract` and immutably updates the trader's on-chain reputation score.

3.2 Secure Copy Trade Execution

1. **Initiation:** A follower initiates a copy trade via the frontend. The request is routed to the Conductor service.

2. **Confidential Signing Request:** The Conductor fetches the follower's encrypted API key blob and the unsigned trade parameters. It sends this payload to the Citadel (TEE Service) over the secure `vsock`.

3. **Attested Signing:** Inside the Nitro Enclave, the TEE service decrypts the API keys into protected memory, signs the trade order, and immediately purges the plaintext keys. It returns only the final, signed order payload to the Conductor. The secret never leaves the hardware-enforced boundary of the enclave.[7]

4. **Broadcast:** The Conductor broadcasts the signed order to the target exchange for execution.

4 Security & Threat Model

The security of the Obscura protocol is analyzed using the STRIDE framework, addressing threats at every layer of the architecture.

5 Conclusion

Obscura presents a novel and comprehensive architecture for a verifiable, privacy-preserving reputation layer for decentralized finance. By rejecting the fragile security models of centralized oracles and instead building upon a foundation of decentralized trust, confidential

Table 1: STRIDE Threat Model for Obscura Protocol

Threat Category	Scenario	Mitigation Strategy
Tampering	Oracle Manipulation: A malicious actor attempts to feed false trade data into the system to illegitimately alter reputation scores.	Primary: BFT consensus via the Sentinel DON's OCR protocol. Secondary: Cryptoeconomic security via staking and slashing, making such attacks economically irrational.[26]
Information Disclosure	TEE Compromise: An attacker attempts to extract sensitive API keys from the Nitro Enclave via a software or side-channel attack.	Primary: Hardware-enforced isolation of Nitro Enclaves. Critical: The cryptographic attestation flow with AWS KMS ensures that only the verified, untampered code can access secrets.[12]
Tampering	Smart Contract Exploit: A vulnerability in the Solidity code (e.g., re-entrancy) is exploited to drain funds or corrupt reputation state.	Strict adherence to security best practices (Checks-Effects-Interactions), extensive use of audited OpenZeppelin libraries, and multiple independent professional security audits.
Elevation of Privilege	Protocol Key Compromise: An attacker gains control of administrative keys for the protocol's services or governance.	Protocol-level keys are managed using a Gnosis Safe multi-signature wallet, requiring a quorum of keyholders for any privileged action.
Denial of Service	API Gateway Overload: The user-facing API is flooded with requests, making the service unavailable.	Implementation of strict IP-based and user-based rate limiting at the network edge, coupled with a cloud provider's DDoS protection service.

computing, and zero-knowledge cryptography, the protocol provides a credibly neutral infrastructure for the future of on-chain identity. The integration of a robust cryptoeconomic model ensures that the network is not only technically secure but also economically rational for participants to maintain. Obscura is designed to be a foundational public good that will unlock new forms of capital efficiency and trust, paving the way for the next generation of sophisticated DeFi applications.