Hydra Proofs

A Minimal Instantiation of the Oblivious Compute Primitive

Abstract

Hydra Proofs are a minimal, working instantiation of the Oblivious Compute (OC) primitive. They demonstrate that distributed convergence can be achieved through deterministic selection and erasure, without consensus protocols, historical logs, or coordinated agreement. Hydra Proofs are exercised within the Hydra system, which exists to make the Oblivious Compute mechanism observable in practice. Neither Hydra nor Hydra Proofs are production systems.

1. Motivation

Oblivious Compute reframes distributed correctness as the result of competitive erasure rather than negotiated agreement. While the primitive is conceptually simple, its credibility depends on whether it can be instantiated without quietly reintroducing history, coordination, or implicit consensus mechanisms. Hydra Proofs exist to answer that question directly.

Hydra Proofs are not designed to optimize for performance, fairness, or fault tolerance. Their purpose is existence. They ask whether a distributed system can converge by exchanging complete candidate states and deterministically erasing losers, while remaining oblivious to ordering, provenance, and past interactions. If this can be shown in even the smallest non-trivial system, the primitive is viable.

2. System Overview

Hydra Proofs operate within Hydra, a minimal peer-based execution environment. Each Hydra node maintains a local candidate state and exchanges candidate states directly with others. Nodes do not track message order, retain logs, or participate in rounds of agreement. At any moment, multiple candidate states may exist across the network.

When candidate states interact, Hydra Proofs apply a deterministic selection rule. One state survives. All others are erased completely. Nodes overwrite their local state with the surviving state and continue. Discarded states leave no trace and exert no influence on future behavior.

Hydra does not attempt to reconcile divergence. Divergence is expected. Convergence occurs not by alignment, but by elimination.

3. The Three-Crown Gate

Hydra Proofs implement a fixed three-crown gate. This structure is chosen as the smallest non-degenerate configuration capable of inducing competitive collapse without escalation or deadlock. Pairwise interaction alone is insufficient to demonstrate oblivious convergence; triadic interaction is the minimal structure that allows selection pressure to resolve contention deterministically.

In this sense, the three-crown gate behaves less like a messaging topology and more like a minimal partitioning scheme. By forcing candidate states into a triadic comparison domain, the gate constrains competition locally and prevents ambiguity from propagating globally. Partitioning here is not persistent or structural, but momentary and functional, existing only long enough for erasure to resolve contention.

The interaction resembles a cyclic comparator rather than a sequence. Like Rock–Paper–Scissors, resolution depends only on immediate dominance relationships between presented states, not on ordering, accumulation, or history. No prior outcomes influence the result, and no state gains advantage by persistence alone.

The three-crown gate is not presented as optimal or universal. It is presented as sufficient. By fixing the gate structure, Hydra Proofs isolate the Oblivious Compute mechanism itself rather than exploring a broader design space.

4. Convergence Through Erasure

Hydra Proofs do not merge, average, or negotiate between competing states. When states collide, one survives and the others disappear. There is no memory of failed candidates, no rollback, and no appeal. Correctness is defined operationally as the state that remains.

Adversarial or malformed states are not filtered in advance. They are treated identically to well-formed states. If they fail to survive, they are erased. If they survive, they become the truth. Byzantine behavior cannot accumulate leverage because erased states leave no residue and cannot influence future selection.

In this sense, Hydra Proofs are indifferent to intent. The mechanism does not attempt to detect lies. It simply removes everything that does not persist.

5. Scope and Intent

Hydra Proofs are intentionally minimal. They do not provide durability guarantees, auditability, fairness, or production-grade fault tolerance. These omissions are not oversights but boundaries. Hydra Proofs exist to demonstrate that oblivious convergence is possible without carrying historical state forward.

As an instantiation, Hydra Proofs should be read alongside the Oblivious Compute primitive. They are not a replacement for consensus systems, nor an attempt to solve all distributed coordination problems. They demonstrate a different axis of computation, where correctness emerges through erasure rather than agreement.

6. Conclusion

Hydra Proofs show that Oblivious Compute can be instantiated in a concrete distributed system using only deterministic selection and erasure. By fixing the gate structure and minimizing system complexity, Hydra exposes the primitive directly, without orchestration layers or hidden coordination.

If Hydra Proofs fail, they fail cleanly. If they succeed, they demonstrate that correctness without memory is not only conceivable, but implementable.

A minimal illustrative code skeleton accompanying Hydra Proofs is provided in the repository for reference.