AVS for Prediction Markets - Devcon Hacker House 2024

Links:

Github

Introduction

As a part of the Infinite Hacker House at Devcon Bangkok our team of 4 Blockchain at Berkeley members hacked from Nov 11 - Nov 15. This research document details our overarching idea, our progress, and our experience working with Layer and Eigenlayer.

In this project, we implemented an AVS-powered decentralized oracle, incorporating restaking mechanisms provided by EigenLayer to help build trust and efficiency of prediction markets.

Why Prediction Markets on Eigenlayer

Eigenlayer enables “restaking” where users can leverage their staked ETH across multiple networks, thereby compounding security. This boosts the security of a prediction market by utilizing existing staked ETH to secure multiple applications. Prediction markets often involve real financial incentives, so the increased security from a widely staked asset like ETH can make the platform more resilient to attacks and manipulation.

Eigenlayer’s architecture supports interoperability between different modules, allowing a prediction market to easily incorporate services like identity verification, custom dispute resolution, or specialized data feeds. This modularity makes it easy to integrate various decentralized applications that can enhance a prediction market, like specialized oracle services or fraud detection algorithms.

A sample use case would be the example of sports betting. Our AVS oracle could receive data from verifiable APIs such as ESPN. This data would be verified by staked operators who validate the outcome of the event before finality on-chain. This sequence helps to ensure security and transparency for those who placed bets on the outcome of this event.

By reusing Ethereum’s existing staked assets for security, Eigenlayer minimizes the need for prediction market participants to pay additional security fees. This efficiency lowers transaction costs and makes participation more affordable for users, which is crucial in prediction markets where users expect low transaction fees.

Prediction markets often benefit from having a long-term, sustainable model where users have ongoing incentives to participate honestly. By building on Eigenlayer, which aligns incentives for Ethereum stakers to maintain security, prediction markets can tap into a consistent and active base of stakers incentivized to keep the system secure and operational over time.

Technical Design

On the frontend, users can bet on on-chain or real-world activities. This would operate similarly to other on-chain betting markets like Polymarket.

The frontend itself would be run on a centralized server but all logic would be run using off-chain contracts and logic.

Oracle

A major improvement for using Eigenlayer compared to other chains would be the use of a decentralized oracle.

Oracles on Eigenlayer can use the restaked ETH of stakers as collateral to ensure the integrity of data they provide. If a staker provides incorrect or manipulated data, their restaked collateral can be slashed. This creates strong incentives for honest participation and robustly secures the oracle.

Oracles inherit Ethereum-level security by being secured through staked ETH, avoiding reliance on potentially less secure validators or external collateral systems.

Oracles can rely on a network of restaked nodes that aggregate data from off-chain sources. Each node’s contribution to the oracle’s output could be weighted by the size of its stake. This ensures that those with higher financial exposure to slashing have a greater influence, increasing the reliability of the data.

As a preliminary design, the oracle we implemented is a simple Threshold Consensus: > ½ majority must agree on the submitted data for it to be finalized.

In applications such as prediction markets, where timely resolution is of utmost importance, efficient consensus was a significant factor in deciding what threshold consensus to use. Ultimately, we decided on using a threshold of ½ majority because it requires the least amount of time and computation, allowing for quick resolution of events.

Operators who submit incorrect or deliberately false data are penalized through slashing mechanisms. The threat of financial loss enforces honest reporting, aligning operators' incentives with the accuracy of the oracle.

Once the oracle reaches consensus, the data is delivered to the requesting smart contract on-chain via an Automated Oracle Contract. This ensures the data is tamper-proof and verifiable by all participants.

The final data is written to the Data Feed Contract and made available to the requesting application.

Emitting State

When the frontend submits an event, contracts take the event and emit it to operators, who have a time frame to vote on it. Theoretically, operators would only vote on it after the event has occurred/when there is an obvious right/wrong decision. Any operator that choses a decision that constitutes the minority decision gets slashed.

Task Queue

Tasks are queued to execute the on chain logic after state is emitted. This uses Layer's task queue.

Our Proposed Implementation

[

Screenshot 2024-11-21 at 2.35.01 AM

1050×712 104 KB

](https://canada1.discourse-cdn.com/flex028/uploads/eigenlayer/original/2X/b/b1d94438b97213c574153d870690db4dca0f7c3c.png)

[

1069×1600 124 KB

](https://canada1.discourse-cdn.com/flex028/uploads/eigenlayer/original/2X/9/978855eb947822eea229a59c360cf750a4a296cf.jpeg)

Figure 1: Sequence diagram illustrating the AVS workflow, from task creation to operator consensus and finalization.

Feedback

We had an amazing time building with EigenLayer and Layer, but had a few speed bumps along the way. While Layer documentation was sufficient, more detailed documentation regarding smart contract interaction and deployment would have helped us accelerate the troubleshooting process. Documentation regarding front-end interaction with Layer contracts point to inactive links and may need to be updated.

In the process of building out the backend, we faced compilation issues regarding the task queue, especially when trying to read tasks from the queue. Additional documentation with an example reading from a task queue would have been of great help here.

Since we were new to the AVS-related technologies like WASM and WASI, we spent extra time understanding these components. However, we see huge potential in Layer and believe that improved documentation, more examples, and better debugging tools may make it much more accessible for a wide range of developers.

Conclusion and Next Steps

Ultimately, we had an overwhelmingly positive experience building on EigenLayer and Layer. We believe that this project demonstrates the utility and potential of a decentralized oracle powered using AVS. With benefits to security, scalability, and cost, there are clear incentives to develop such an initiative.

Immediate next steps of this project involve expanding the potential range of events on which users can place bets on. Increasing the scope of supported events on our platform will help facilitate a larger, more diverse audience as well as increasing engagement with the decentralized ecosystem.

The entire team had an amazing experience interacting with both the EigenLayer and Layer teams. We look forward to exploring further innovations in decentralized infrastructure and enhancing the capabilities of AVS-powered systems.

Authors

Derrick Cui, Sara Magaziotis Ginori, Shrithan Sandadi, Ezra Tramble