Muon: The Web3 Validation Layer

Reza Bakhshandeh

v1.0 2025

Contents

1 Introduction

Muon is a decentralized, general-purpose validation layer designed to overcome blockchain isolation and enable a new generation of modular dApps. It allows dApps to run essential components as request-based micro-validators on a secure, decentralized network, secured by a modular security stack, including EigenLayer.

2 Web3 Development Stack Problems

In Web2, projects typically consist of three components:

- **Frontend**: The user interface.
- **Backend**: Processes frontend requests and handles application logic.
- **Database**: Stores data and implements part of the logic through stored procedures.

In the Web3 development stack, smart contracts function as a combination of the database and stored procedures. However, the backend layer that processes and validates data before pushing it to the blockchain is missing. This leads to several challenges, such as blockchain isolation, inflexibility, and the inability to adapt to advancements in software development and emerging technologies.

Oracles, the primary solution for addressing blockchain isolation, are inflexible, costly, limited, and not truly trustless. Additionally, they function more as services that deliver data to smart contracts on a few select chains rather than providing a general, decentralized, and scalable solution to the problem.

3 Muon's Solution

Muon is a request-based, general-purpose validation layer that serves as a backend for Web3 apps, allowing them to pre-process and validate any data before pushing it on-chain.

DApps can deploy their trustless components as micro-validators (MuonApps) on the validation layer. Users or dApp clients send requests to these micro-validators that validate and preprocess data, generating proof that can be verified on any chain.

Developing MuonApps can be done using high-level programming languages like JavaScript. Unlike smart contracts, micro-validators are not isolated—they have full web access, allowing them to fetch and validate data from any source.

Muon protocol employs a multi-layer security stack, where each request to micro-validators is processed by multiple validators, ensuring high security.

4 How Muon Protocol Works

Developing and running MuonApps is both simple and straightforward. The process is as follows:

4.1 MuonApp Development

Developers can use high-level programming languages to develop MuonApps. For example, here is a sample MuonApp for a simple price feed.
`https://github.com/muon-protocol/muon-apps/blob/master/general/simple_oracle.js`

4.2 Deployment and Operation

MuonApps are deployed on the Muon Network and operate as micro-validators. Each app has its own address and methods, similar to smart contracts. Users and dApp clients can send requests to these micro-validators and receive cryptographic signatures in response.

4.3 Verifying Signatures on Smart Contracts or Off-Chain

The dApp client, user, or an executor pushes the signed data to a smart contract, which then verifies the signatures to validate the data. Additionally, Muon signatures can also be verified using off-chain tools, enabling the development of trustless apps that do not require smart contracts.

5 Modular and Multi-Layer Security Stack

Muon employs a modular security stack to ensure robust and decentralized protection. Multiple parties can operate Muon Apps and participate in transaction signing, providing comprehensive security. The current security layers include:

5.1 Muon TSS Network

Utilizes a Threshold Signature Scheme (TSS) with Multi-Party Computation (MPC) and a proof-of-randomness algorithm to run Muon Apps on a large network of nodes, ensuring distributed and tamper-proof execution.

5.2 Shield Nodes

Projects can deploy their own dedicated nodes to run Muon Apps, adding an extra layer of security tailored to their specific needs.

5.3 EigenLayer

Through a collaboration with EigenLayer, Muon Apps can operate on an AVS, further enhancing security and decentralization.

6 Structure and Components

6.1 MuonApps

MuonApps are applications that projects can develop and deploy on the network to run their micro-validators. Unlike smart contracts, which operate on the blockchain and are isolated from real-world data, MuonApps have access to real-world data and function similarly to services running on a computer or cloud platform.

Example of a MuonApp for a simple price feed:
`https://github.com/muon-protocol/muon-apps/blob/master/general/simple_oracle.js`

6.2 Muon TSS Network

The TSS Muon network is a decentralized network of nodes capable of deploying and running MuonApps. Each app is deployed on a dynamic subnet, where subnet nodes collaborate to operate MuonApps and provide a TSS signature as proof. This proof can be verified on both blockchain and off-chain components.

6.3 MuonAVS on EigenLayer

The Muon TSS network is decentralized and secure; it eliminates any single point of failure through threshold signatures. However, there still might be a risk of collusion among subnet nodes. That's why we have implemented a modular security model by combining proofs from different validators. MuonAVS on EigenLayer can run the Muon app engine, execute MuonApps, handle requests, and provide their own proofs as a new security layer.

6.4 Shield Server and Other Nodes

The security modules and validators are not limited to the Muon TSS network and EigenLayer. Projects can also run their own shield nodes or use nodes operated by other trusted parties.

6.5 Client SDKs

This SDK presents a set of smart contract libraries that allow dApps to verify proofs on-chain. This verification process is chain-independent and can be performed on any blockchain.

7 Use Cases

7.1 General-Purpose Oracle

A wide range of Oracle use cases, including cross-chain tools, bridges, and data feeds, can be handled using MuonApps.

7.2 Verifying Blockchain Data

Verifying blockchain data, such as storage variables, transactions, and blocks, is challenging for micro clients that need to verify chain data. These tools can delegate data verification to micro-validators.

7.3 Backend for dApps

The Web2 development stack has three components: frontend, backend, and database. The backend processes data and saves it in the database. Some databases support stored procedures, allowing developers to implement part of the logic within the database. In Web3, blockchains and smart contracts function like the database for dApps. A validation layer can act as the backend for dApps, enabling them to process data before pushing it to the blockchain. This approach makes dApp development more flexible and enhances their capabilities.

8 Muon vs Oracles

Oracles provide off-chain data to smart contracts but are often limited, inflexible, expensive, and not fully trustless. Muon is a flexible, cost-effective, chain-agnostic protocol that lets projects run custom solutions while owning their security.

9 Muon and EigenLayer

Muon and EigenLayer share the same vision: providing a modular stack for web3 development. Blockchains were originally designed as all-in-one platforms for running dApps, but their limitations stem from this monolithic approach.

By introducing a modular architecture where blockchains are just one component, Muon and EigenLayer enable a more flexible and scalable web3 development stack, positioning themselves as key players in its evolution.

The shift toward modular web3 architecture is already underway. Data availability layers like EigenDA are a part of this transformation, and Muon aims to serve as the validation layer—another essential component that web3 needs.

10 Benefits of Running a Validation Layer on EigenLayer

10.1 Easier EigenLayer Integration

Running an AVS for small to medium-scale projects is currently challenging. By separating the execution and validation layers, projects can integrate EigenLayer with just a few lines of code—making it as easy as deploying a smart contract.

10.2 A New Approach to App Development

EigenLayer aims to expand dApp development to a broader range of use cases. A general validation layer enables a more flexible and modular approach, aligning with EigenLayer's strategic vision.

10.3 Enabling Trustless Web2 development

Currently, trustless apps can only be built using smart contracts and blockchains, which are limited, inflexible, and not scalable for use cases like Web2 games.

A validation layer allows Web2 apps to be trustless by running trustless components as microvalidators. This lets them validate responses within their apps without relying on smart contracts.

10.4 Vampire Attack on Market-ready Use Cases

Integration with EigenLayer through the validation layer is simple for projects. Existing solutions like bridges and multichain tokens can be integrated to enhance security and reputation.

A LayerZero DVN has already been implemented on Muon, allowing any project using LayerZero technology to leverage it.

10.5 Supporting Validator-as-a-Service (VaaS) Tools

Currently, EigenLayer is the trustless version of AWS, Google Cloud, and similar services. The validation layer runs something like a microservices platform on EigenLayer, making it easier for small projects to leverage it.

10.6 Making EigenLayer Chain-Agnostic

Verifying Muon signatures and Muon AVS transactions can be done on any chain.

11 Applications and Markets

A trustless validation layer has the potential to serve a wide range of services. A simple micro-validator that loads and verifies blockchain data could power the next generation of cross-chain tools, entering markets currently dominated by major players like Cosmos and LayerZero. Additionally, it paves the way for developing trustless services that utilize blockchains primarily as storage and trustless databases.

Another significant market is expanding oracle solutions, providing a broader range of trustless data for dApps, including price feeds, real-world data, and more.

Moreover, in the era of AI, trustless data and action services for AI agents will become increasingly valuable. A trustless infrastructure tailored for AI agents represents a future market with immense potential.

A validation layer on EigenLayer can effectively target all of these markets, offering a secure and decentralized alternative for various applications.

12 MuonApp Examples

12.1 EVM Data Verifier

A MuonApp that loads and verifies data from any EVM-based chain:
`https://github.com/muon-protocol/muon-apps/blob/master/general/evm_data_verifier.js`

12.2 LayerZero DVN

MuonApp as the verifier for a LayerZero DVN:
`https://github.com/muon-protocol/muon-apps/blob/master/general/evm_data_verifier.js`

12.3 deRand: A VRF Protocol

A VRF protocol as a micro validator. It can be used as a programmable on-chain VRF, similar to oracles, or in off-chain trustless components that require randomness, including games.

https://github.com/muon-protocol/muon-apps/blob/master/general/derand_offchain_vrf.js

More details:
https://medium.com/muon/breaking-rng-barriers-off-chain-verifiable-randomness-856a435f3f5d

13 Muon Team

The Muon team is comprised of a collective of blockchain engineers, AI specialists, and product designers dedicated to building decentralized solutions for the economy of tomorrow.

Lead developer, Reza Bakhshandeh, is a seasoned expert with over 21 years of experience in blockchain and AI development.

14 Partners and Integrations

- Avalanche: https://www.avax.network/
- Symmio: https://www.symm.io/
- LayerZero: https://layerzero.network/
- Waterfall: https://waterfall.network/
- Arthera: https://www.arthera.net/
- XDC: https://xdc.org/
- Scroll Network: https://open.scroll.io/
- Linea: https://linea.build/
- Zellular: https://www.zellular.xyz/
- Fear: https://www.fear.io/
- Thena: https://thena.fi/
- Ramses: https://www.ramses.exchange/

15 Links and Resources

Explorer: https://explorer.muon.net
Website: https://www.muon.net/
Twitter: https://x.com/muon_net
Docs: https://docs.muon.net/muon-protocol
Muon Apps: https://github.com/muon-protocol/muon-apps/tree/master/general
Github: https://github.com/muon-protocol

MuonAVS on EigenLayer:
https://github.com/muon-protocol/muon-avs-registration
https://github.com/muon-protocol/muon-avs-js

16 Conclusion

We propose a solution for building a validation layer through the collaboration of EigenLayer and Muon as a micro-validator platform. This validation layer could become a key component of the Web3 development stack. It has the potential to address a wide range of existing markets while also creating new opportunities for developing dApps that can support a broader range of use cases.