Aligned: proof verification and aggregation layers for Ethereum

Federico Carrone, Roberto J. Catalán, Mauro Toscano, and Diego Kingston

November 2024

Abstract

We live in an era where we cannot trust what we read, see, or hear. This phenomenon has been unfolding worldwide, especially on the Internet. Advances in AI and a conflicted world create friction in communicating and exchanging value. We aim to build a future where humans can trust each other and coordinate on the Internet. The only way to achieve this future is through decentralization and verifiable computation. Blockchains are verifiable computers, attaining this property through economic incentives, consensus, and re-execution. The introduction of zero-knowledge proofs provided an alternative by verifying cryptographic proofs, which can be checked much faster, leading to the birth of zk-rollups. The solution has met some challenges related to fragmented liquidity and users, high verification costs due to the Ethereum Virtual Machine's (EVM) constraints, and difficulties in keeping up with the innovation in proving systems. In other words, the infrastructure at our disposal was not designed to be the universal verifier we need. EigenLayer removes the constraints of the EVM by allowing developers to create new protocols using Ethereum's trust layer. This allows for open innovation in Ethereum. We can introduce a new infrastructure to accelerate Ethereum's roadmap without modifying the underlying base protocol. We propose Aligned as a zk-verification layer, providing affordable verification with high throughput and low latency. This will be made possible by creating a layer specifically designed with zk-proofs in mind, granting developers access to a decentralized verification network that is fast, affordable, scalable, and secured by Ethereum through restaking via EigenLayer. It offers a more cost-effective solution, reducing the dependency on fluctuating gas prices and improving the bridging and overall user experience. Moreover, we are helping to foster open innovation in Ethereum through verifiable computation, leading to the creation and mass adoption of new trustless applications by quickly incorporating new customized proof systems, lowering verification costs, and making them more developer-friendly. This will help Ethereum update and adopt new proof systems and technologies without changing it.

1 Introduction

The financial system depends on intermediaries: an army of auditors, regulators, and accountants. The correct functioning of the financial machine is based on the integrity of its financial institutions. Integrity is maintained with positive economic incentives, jail time, fines, and costly lawsuits if the intermediaries do not behave as the state and society expect from them. Bitcoin [1], due to the 2008 crisis, created a permissionless financial system where its users can send and receive digital money without intermediaries and where nobody can block transactions. In countries where stagnation and inflation erode the trust of citizens in the financial system and the state, Bitcoin [1] and stablecoins on top of Ethereum [2] are used daily by the young population to save and avoid capital controls. In developed countries, its usage is not as massive, as most citizens trust the traditional financial

system and the state. However, the world is becoming more complex. Banks are failing in the US and Europe, a new war is taking place in Europe, debt levels are not sustainable in many countries, and technological change keeps accelerating. The progress in artificial intelligence (AI) and a conflicted world create friction in how we communicate and exchange value. We aim to build a future where humans can trust each other and coordinate on the Internet. The only way to achieve this future is through decentralization and verifiable computation. Blockchains are decentralized, verifiable computers that achieve verifiability through reexecution and consensus. The problem with re-execution is that all the nodes must perform the same work, and the weakest ones act as bottlenecks, limiting their throughput. Adding more hardware makes the blockchain more robust but does not make it faster, contrary to what happens in Web2. The use of succinct cryptographic proofs of integrity allows us to remove these problems by having one party generate a proof of the computation, and the rest can verify this proof much faster than re-executing it. This allows blockchains to scale by proving computation off-chain and verifying it on-chain while keeping the same guarantees as if the nodes had re-executed the computation.

The advancement of proof systems over the last decade has allowed us to prove more complex computations, enabling new applications. The development of general-purpose virtual machines for proving programs written in languages such as Rust makes writing verifiable apps simpler, reducing costs and go-to-market time. However, high verification costs and small throughput remain a challenge that hinders further progress: Only projects with enough capital or scale can afford the fees. In Ethereum, verification competes for blockspace with other applications; network congestion can result in higher costs. Most proof systems cost over 250,000 gas in Ethereum, and with the current gas cap set at 30,000,000, the EVM cannot process more than 120 proofs per block. Therefore, removing these barriers is crucial for the adoption of ZK.

With Bitcoin, Ethereum, ZK, and Layer 2s, EigenLayer is one of the most significant innovations in the space. Until now, blockchains have sold trust through block space. EigenLayer enables the sale and purchase of trust in other forms. This solves the problem developers face when building their infrastructure: providing their economic security [3]. Proof of stake (PoS) has emerged as a prevailing consensus mechanism, as it is not energy-intensive as proof of work (PoW) nor as centralized as proof of authority. However, bootstrapping a PoS network meets several challenges, such as [3]:

- It is difficult to know where the stakers are located.
- Stakers must invest large amounts of money in obtaining a stake in the network, usually in the form of a token, which is a volatile asset.
- Stakers must forgo other reward opportunities.
- The security model is inadequate since the cost to corrupt the dApp is the cost needed to compromise its weakest infrastructural dependency.

Ethereum stakers incur fixed costs to run the network and receive rewards from the network to compensate for those costs and inflation. EigenLayer allows them, using the same stake (that is, restaking), to participate in other applications (called actively validated services, AVS) and obtain additional rewards. The marginal cost of participating is small. Restaking pools security, instead of fragmenting it. It also creates a marketplace where validators can choose whether to opt in or out of each application built on EigenLayer [3]. Among the possible applications that can be built on top of EigenLayer, we have [3]:

- Hyperscale data availability layers.

- Decentralized sequencers.
- Oracles.
- Opt-In MEV management.
- Fast-Mode bridges for rollups.

Most new blockchain developments occur around or end up on Ethereum, which boasts the largest community of researchers and engineers and is a significant source of liquidity for decentralized applications. However, Ethereum was not originally designed with zero-knowledge or validity proofs in mind. Some of these design considerations introduce an additional burden to proving systems, and it is difficult to follow the fast pace and changes in the ZK industry. How can we deal with these complications without sacrificing Ethereum's security? Can we further extend the capabilities of Ethereum to take advantage of all the power of proving systems? The answer lies in creating a verification layer that can tap into Ethereum's trust via EigenLayer. In this way, we can benefit from Ethereum's decentralized infrastructure without dealing with the constraints of the EVM.

Aligned creates a new layer that rents the security offered by Ethereum to verify zero-knowledge proofs. The boundaries of the EVM no longer constrain this layer, and several verifiers can be deployed to verify any proof quickly. Why do we need such a layer?

1. Ethereum is expensive in terms of storage, execution, and consensus.
2. It is not a fair market for proof systems (precompiles subsidizing KZG commitments)
3. Updating and including new proof systems or techniques is hard.
4. Several proof systems are costly to run on the EVM, even though they are efficient for many applications.
5. The gas fee can change due to something related to storage, which affects proof verification. Such a critical piece of infrastructure should not compete for block space with other applications. In the long term, the demand for Ethereum will always increase and, therefore, the cost of verification will also increase.
6. Reduced latency.
7. As ZK becomes more accessible, the demand for ZK verification will increase. We have seen developments towards zkvms for general-purpose languages such as Rust, which abstract developers from the gory details of ZK and write their code as they have always done.
8. It simplifies bridging between different chains, partially solving the fragmentation of liquidity and state. By being able to verify any proof, we avoid having to write a specific wrapper to make the proofs EVM-friendly.

Aligned provides an alternative to reduce costs and significantly increase throughput. This is achieved by two different products: the proof verification layer and the proof aggregation layer. The proof verification layer works with a subset of Ethereum's validators via restaking. Validators (Operators) receive proofs, verify them using the verification code written in Rust or another higher-level language, and sign messages with BLS signatures. If a two-thirds majority agrees, the results are posted as verified in Ethereum. Since Aligned's operators only need to run the verification code on bare metal, we have several advantages compared to running it on top of the EVM:

- We can optimize the code for speed, not gas consumption.
- We can leverage parallelization to increase throughput.
- Since the gas limit does not constrain us, we can verify proof systems that are too expensive for Ethereum, such as Kimchi or Binius [4].
- Adding new proof systems is straightforward.

Preliminary numbers show that Aligned can verify more than 1000 proofs per second, over two orders of magnitude more than the EVM at nominal capacity. We can split the task creation and verification cost between thousands of proofs using effective batching techniques.

The proof aggregation layer performs recursive proof verification to batch several proofs into one that will be verified on-chain. The validity of the final proof implies that the original proofs are also valid. The steps for this compression are the following:

1. Conversion from the original proof system to a recursion-friendly proof system.
2. Performing a reduction in the number of proofs using an n-ary tree (for simplicity, a binary tree).
3. Convert the last proof into a low-cost proof system.

Rollups and aggregation layers usually follow this strategy; the main differences are in the choice of proof systems and proof conversion. For example, recursion-friendly proof systems are STARKs or Plonky, and low-cost proof systems include Groth16 [5] or Plonk [6] with KZG commitments.

We think a combination of efficient proving markets and verification networks is fundamental to providing cost-effective solutions to make these tools more massive and set the foundations for the future of decentralized applications.

2 Use cases

Among the possible use cases of Aligned, we have:

- Soft finality for Rollups and Appchains: Aligned provides fast verification of ZK proofs, which can be used to provide soft finality to rollups or other applications.
- Fast bridging: building ZK bridges requires checking a succinct proof to show that the current state of a chain is correct, and then users need to show that their account state is correct. Many ZK protocols use hash functions such as Poseidon or Rescue Prime, which do not have precompiles on Ethereum, making verifying the chain's state and account expensive. With Aligned, you can show your account state using another ZK proof, and all proofs can be verified cheaply and with low latency in Aligned.
- New settlement layers (using Aligned + EigenDA) for Rollups and Intent-based systems.
- P2P protocols based on SNARKs such as payment systems and social networks.
- Alternative L1s interoperable with Ethereum.
- Verifiable Machine Learning: with general-purpose zkvms, we can prove code written in Rust, solving part of the problem of using ML. However, most zkVMs use STARK-based proof systems, leading to high on-chain costs or expensive wrapping. With Aligned, you can directly verify your proof from the zkVM for much less than Ethereum.

- Cheap verification and interoperability for Identity Protocols.
- ZK Oracles: With ZK oracles, we can show that we have a piece of information off-chain and produce a ZK proof by doing some computation with those data. Aligned reduces the cost of using those oracles.
- New credential protocols such as zkTLS-based systems: you can create proofs of data shown on your web browser and verify the result in Ethereum.
- ZK coprocessor: ZK allows complex computations to be delegated from the blockchain to a coprocessor. This can retrieve information from the blockchain and perform the computations securely more efficiently.
- Encrypted Mempools using SNARKs to show the correctness of the encryption.
- Protocols against misinformation and fake news: you can generate proof that an image or audio comes from a given device and show that a published image results from certain transformations performed on the original image.
- On-chain gaming: building zk-based games and verifiable on-chain leaderboards.

3 Aligned's architecture

Figure 1 shows the core components of Aligned and how they work together.

3.1 Components

Aligned has the following components:

- User CLI: used to interact with the verifier task batcher. Sends proof and public input data and receives the verification data in Aligned.
- Verifier Task Batcher: Receives tasks from users, creates batches of tasks, publishes the proof and public data in the Data service, and sends the batches' data to Ethereum. This service is permissionless, so they can run their batcher if someone wants.
- Service Manager (Ethereum Contract): Receives the batches' data and signatures from the BLS signature aggregator. Provides information to validators and light clients on the batches/tasks.
- Data service: temporarily stores the data for proof and public input.
- Operators: In charge of verifying the proofs in each batch and signing messages with the results.
- BLS signature aggregator: receives the signatures from the operators, checks that there is a quorum, and performs the aggregation of the signatures.
- Light Client: samples random tasks from the Service Manager, checks the proofs, and compares with the results posted in Ethereum by Aligned. If there are differences, it can trigger via the proof service an L1 verification and, in case of malicious behavior by Aligned's operators, leads to slashing.

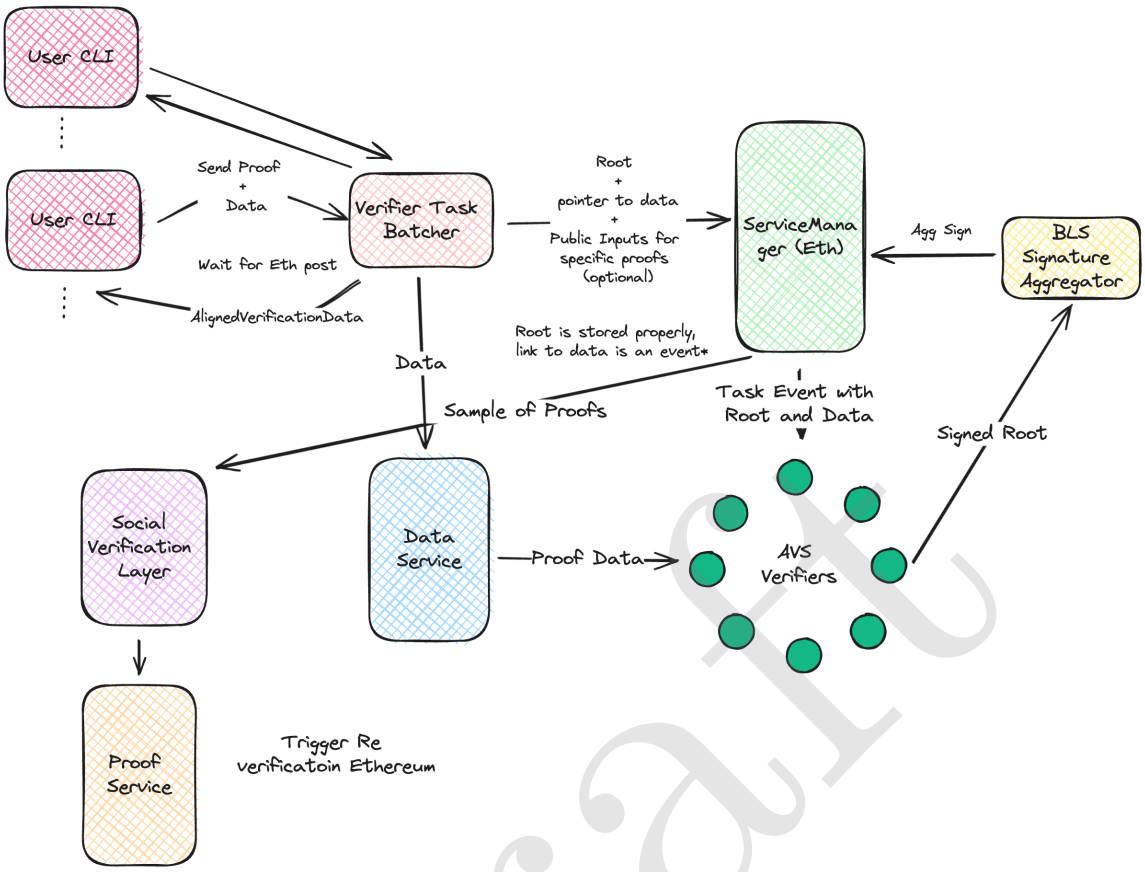

Figure 1: Aligned's core components

- Proof service: receives results from the light clients; if there are differences with the results posted by Aligned, it triggers a re-verification in Ethereum. Notice that any user can also trigger this re-verification.
- Proof aggregator: Once Aligned's operators have verified tasks, it performs recursive proof verification to create one proof that will attest to the validity of all proofs contained in the batch. This proof will be verified on-chain.

3.2 Proof verification layer

3.2.1 Flow for sending a proof and publishing the result in Ethereum (Proof verification layer)

The flow for sending a proof and having the results in Ethereum is as follows:

1. The user uses a provided CLI or SDK to send one proof or many to the batcher and waits.
2. The batcher answers with ProofVerificationData for each proof.

3. The user invokes an IsVerified function with these data in Solidity to check that the proof is valid.

4. (Optional) The user checks that the commitment to the proven program matches what it expects.

3.2.2 Full flow with the internals of the proof (Proof verification layer)

1. The user uses a provided CLI or SDK to send one proof or many to the batcher and waits (Alternatively, the user can run a batcher or interact directly with Ethereum).

2. The batcher accumulates proofs of many users for a few blocks (typically 1-3).

3. The batcher creates a Merkle tree with commitments to all the data submitted by users, uploads the proofs to the Data Service, and creates the verification task in the ServiceManager.

4. The operators, using the data in Ethereum, download the proofs from the DataService. They then verify that the Merkle root is equal to the one in Ethereum and verify all the proofs.

5. If the proofs are valid, they sign the root and send this to the BLS signature aggregator.

6. The signature aggregator accumulates the signed responses until reaching the quorum, then sends the aggregated signature to Ethereum.

7. Ethereum verifies the aggregated signatures and changes the state of the batch to "verified."

3.2.3 Batch structure

The task batch consists of a Merkle tree containing relevant information to verify the proof in the lower-level leaves. The root of the Merkle tree is posted to Ethereum with a pointer to where the data is stored. Each leaf contains the following information:

1. A commitment to the public input of the proof.

2. A commitment to the proof and the information of the proof system.

3. A commitment to the program or the verification key (depending on the proof system used).

4. The address of the proof generator/submitter (optional).

Figure 2 shows the data structure for the batch of proofs.

3.2.4 Reading the results from Ethereum

Once the batch results have been checked on Ethereum, the Aligned contract is updated with them. The consumer's contract can query the Aligned contract to check whether the proof has been included in a successful batch.

Additionally, the contract should be set to receive only proofs of specific programs. For example, in an L2, this may be a particular program that represents the blockchain's state transition.

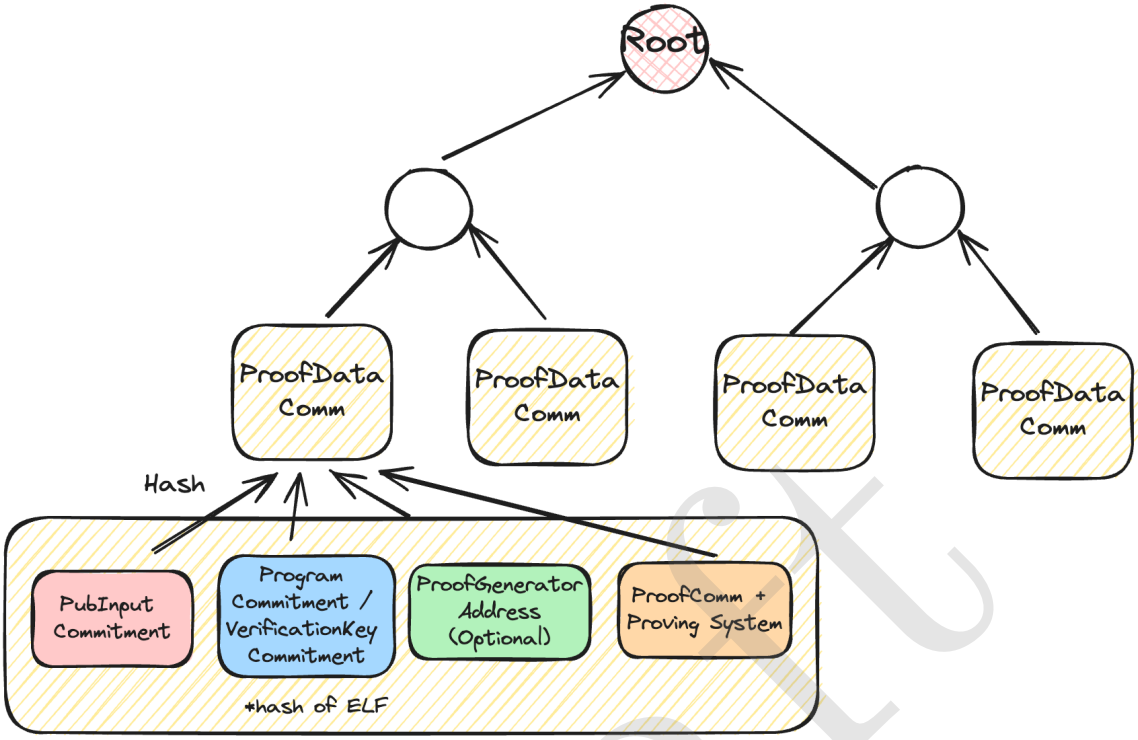

Figure 2: Batch data structure

3.3 Proof aggregation layer

A general Rust prover generates the verification proof for all proofs included in a given period (Figure 3). Even if this proof can be quite ample, there is only one proof that attests to the validity of all proofs. No matter the verifier's complexity, it can be written in Rust and proven by a zero-knowledge virtual machine. To aggregate all the proofs, in the first step, all the proofs are transformed into the execution proofs of the virtual machine, achieving the uniformity of the proof (Figure 3). We can then shrink the proof size by recursively proving the verification of proofs, as shown in the tree diagram, for example, in figure 4.

3.4 Slashing

We need to introduce slashing to have a proper incentive mechanism for a decentralized, permissionless network. This will allow the protocol to penalize participants in case of malicious activity. This area is still in development in most AVS from EigenLayer, so currently, we are exploring two options: one for the short term and another for the future.

The first option consists of subjective slashing: the network needs the consensus of two-thirds of the total operators to finalize and post the result in Ethereum. The operators not in the majority of the quorum will be penalized for being against the result reached by most of the network. This is not a perfect mechanism, but considering that the client software for Aligned Layer will be lightweight, the minimum hardware requirements will be low. Therefore, the network can be easily decentralized

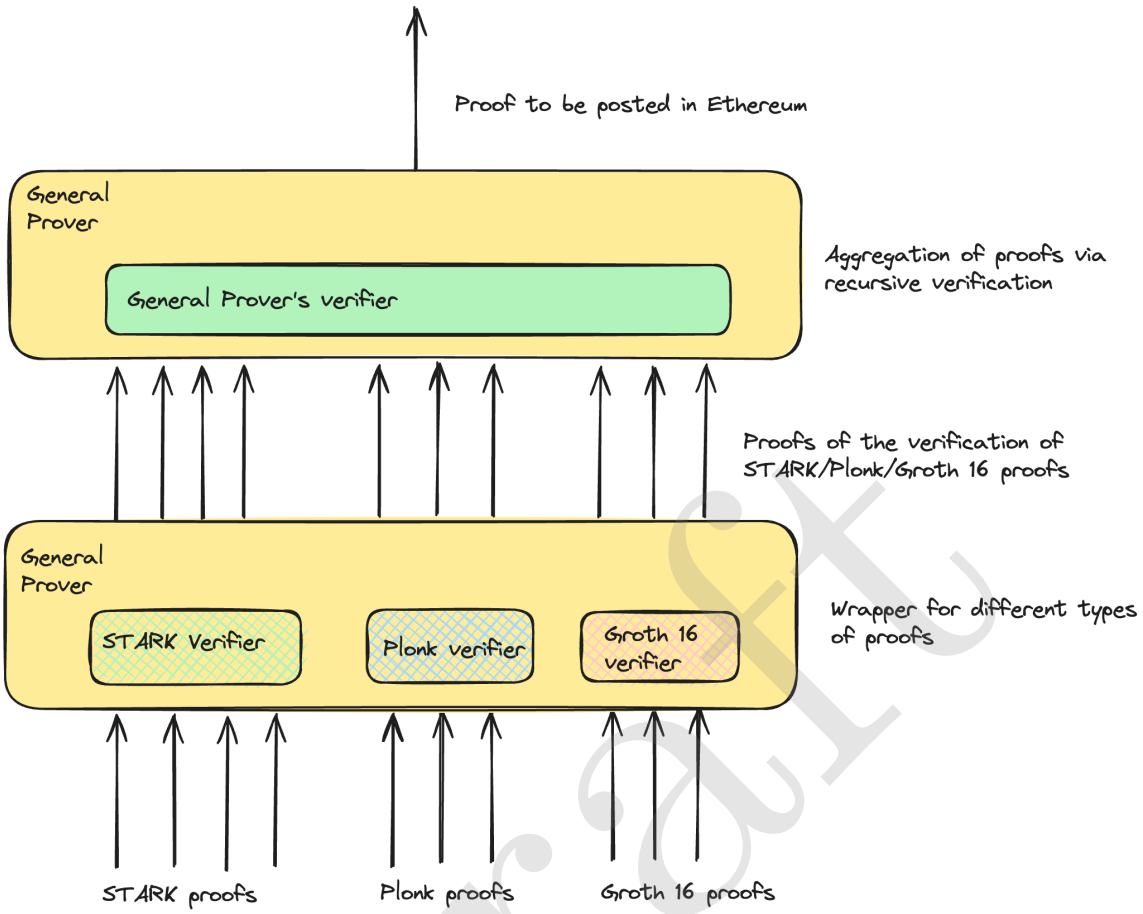

Figure 3: General prover's flow

by allowing many participants to join as operators. The more decentralized the network, the more likely the majority is honest.

In the long term, we are exploring a slashing design that will be categorized for different proving systems. Not all proving systems integrated by Aligned can be verified on-chain in Ethereum, so objective slashing is not doable for all the proof systems accepted. Even though this is the case, the protocol could still provide slashing for different proof systems on-chain, allowing these proof systems to have stronger security guarantees when used in Aligned. For example, Aligned offers verification for Cairo proofs that can be verified in Ethereum if necessary. If most of the network acts maliciously, an honest operator or third party could trigger a slash event by asking the protocol to run the computation fully on-chain. The major drawback is that this would fragment the security guarantees for each type of proof, but it will improve them for many of the proof systems used today in Ethereum.

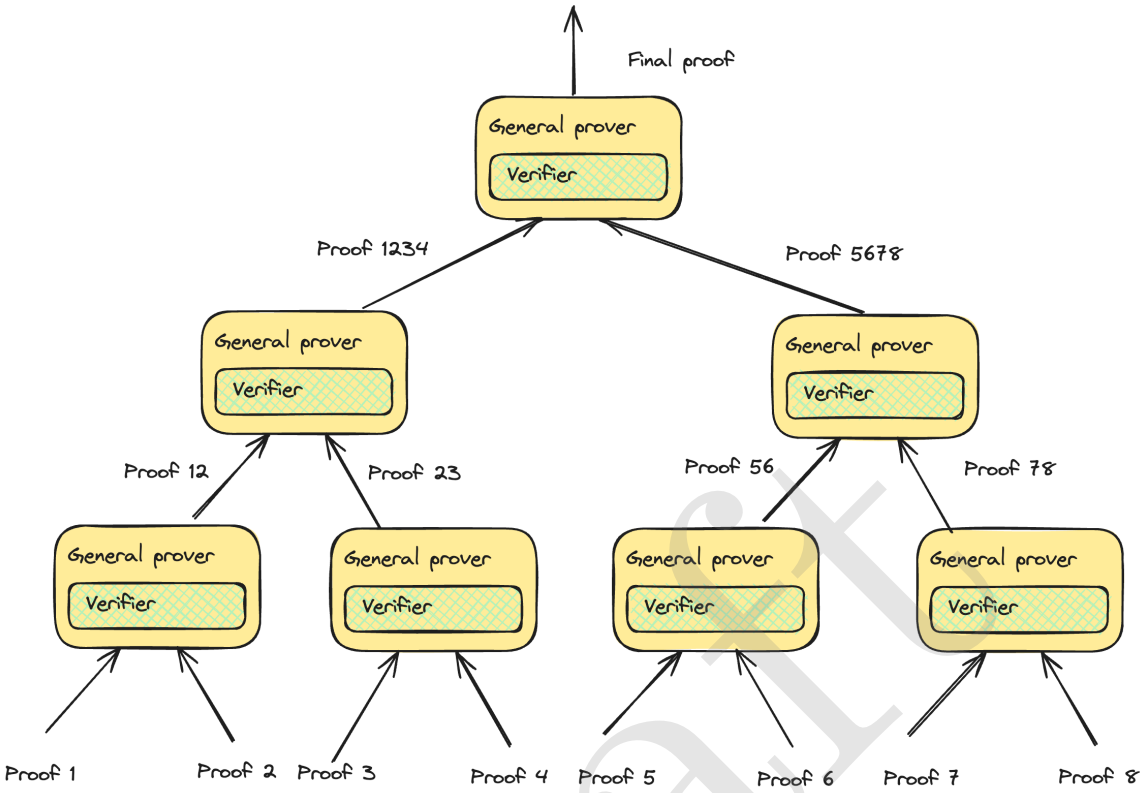

Figure 4: Recursion tree

4 Payment system

The payment system contract handles users' payments to fund their verification tasks. To use the batcher, the user must have funds in his balance account. The batcher has an associated batcher payments smart contract, which receives users' payments and ensures that it can only use these funds to send proofs to Aligned.

Users deposit funds into this smart contract using standard transfers to this address, and the smart contract updates the users' balances. When users send proofs to the batcher, the latter checks whether they have enough funds. Once the batch is complete, the batcher calls its smart contract with the users' data and deducts funds from the senders' balances, creating a new batch in Aligned's service manager. This includes tokens to pay the aggregator.

Users can withdraw or leave the remaining funds to pay for future tasks. To avoid a denial of service on the batcher, the users must first call the contract's unlock function and then call the withdraw function at least 100 blocks later to finish the withdrawal.

The flow for the payment system is shown in figure 5. The steps are:

1. When the batcher calls the smart contract to create a new batch, it gets the batch of Merkle tree leaves, with each leaf signed by the user.

2. The contract then rebuilds the Merkle tree to check the batch Merkle root and verifies the

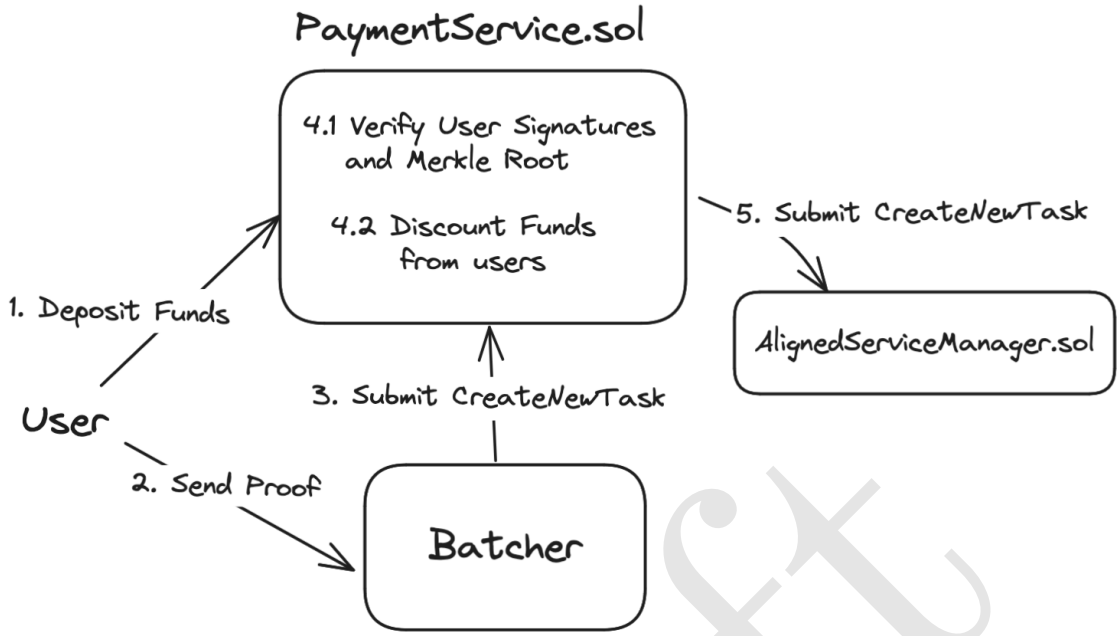

Figure 5: Payment system's flow diagram

user signatures. Each signature also contains a nonce that can only be used once per user to avoid the batcher being able to reuse the same signature.

3. The contract will discount the corresponding funds from the user's balance and create a new batch in the Aligned Service Manager if and only if the Merkle root and the signatures are valid.

5 Costs

The costs for proof verification consist of the following:

- Proof and public input data storage.
- Running the proof verification algorithm.
- Reading and using the result from the verification.

The cost on Ethereum depends on the amount of data/computing required, the network congestion, and the valuation of ETH. The computational effort is measured in gas. For reference, the cost of a transaction in Ethereum is 21,000 gas.

To transform to USD, we need to multiply the gas cost by the gas cost expressed in gwei (10^{-9} ETH) and the conversion rate between ETH and USD:

$$C_{USD} = C_{gas} V_{gas} V_{ETH}$$

For example, if the gas cost is eight gwei/gas and the ETH is worth 3000 USD/ETH, a transaction costs:

$$C_{USD} = 21,000 \times 8 \times 10^{-9} \times 3000 = \$0.504$$

Aligned reduces cost by splitting the cost of task creation and verification among several proofs. The gas cost per proof for a batch containing N proofs is:

$$C_{gas} = \frac{C_{task} + C_{verification}}{N} + C_{read}$$

C_{read} in this case is the cost the user has to pay to use the proof in a contract.

6 Dual Staking

We are proposing a dual-staking model. First, we need to bootstrap the proof of stake network using ETH with restaking from EigenLayer. In the second stage, as with any piece of critical infrastructure, we need a native token for governance and sovereignty. We want the costs of compromising both liveness and security to be high.

In particular, governance will be necessary to decentralize the decision-making of important architectural changes, such as which proof systems to include in a new version of the protocol.

7 Comparing verification in Aligned and Ethereum

Table 1 compares verification using Aligned's proof verification layer, the proof aggregation layer and Ethereum. Aligned offers a more cost-effective alternative with practically the same security guarantees as Ethereum. Soft finality reduces latency, improving the user experience and bridging different chains simpler and faster.

Variable	Proof verification layer	Proof aggregation layer	Ethereum alone
Speed	Faster	Slower	Base
Cost	Cheaper	Cheaper	Base
Security	Proportional	Base	Base

Table 1: Comparison in terms of speed, cost, and security. Ethereum is the base case

8 Conclusion

Blockchains provide integrity through re-execution and consensus. The introduction of zero-knowledge proofs helped solve scalability issues by providing a faster alternative to reexecution: verification of short cryptographic proofs. Rollups helped solve scalability issues using this technology while introducing fragmented liquidity and users, requiring bridging and making verification costs dependent on gas prices. EigenLayer allows developers to create protocols that tap into Ethereum's trust network. We propose Aligned, an infrastructure dedicated to proof verification and aggregation, tapping into Ethereum's security via EigenLayer. Aligned allows different proof systems to be introduced, making it easy for developers to use proving systems tailored to their needs. The system allows for fast, soft finality, which reduces latency and lowers bridging costs, making the user experience simpler and

better. It offers two different operation modes, a proof verification layer and a proof aggregation layer, tailored for various needs while ensuring low costs and high throughput.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Gavin Wood. Ethereum: a secure decentralized generalized transaction ledger, 2014.

[3] The EigenLayer Team. Eigenlayer: The restaking collective.

[4] Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. Cryptology ePrint Archive, Paper 2023/1784, 2023. https://eprint.iacr.org/2023/1784.

[5] Jens Groth. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Paper 2016/260, 2016. https://eprint.iacr.org/2016/260.

[6] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019. https://eprint.iacr.org/2019/953.