Aligned Layer: universal verification layer (DRAFT)

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

March 2024

Abstract

We live in an era where we cannot trust what we read, see, or even hear. This phenomenon has been unfolding worldwide, especially on the Internet. The advancements in AI and a conflicted world are creating frictions 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 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 with some challenges, related to fragmented liquidity and users, high verification costs due to EVM constraints and the difficulties in keeping 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 the trust layer from Ethereum. This allows for open innovation in Ethereum. We can introduce new infrastructure to accelerate the roadmap for Ethereum without modifying the underlying base protocol. We propose Aligned Layer to serve as the universal verification layer, aiming to be the main source of truth for the Internet. 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 making bridging and overall user experience better. Moreover, we are helping foster open innovation in Ethereum through verifiable computation, leading to the creation and mass adoption of new trustless applications, by easily incorporating new taylored proof systems, lowering verification costs, and making it more developer-friendly. This will help Ethereum update and adopt new proof systems and technologies without changing Ethereum itself.

1 Introduction

The financial system depends on the existence of intermediaries: an army of auditors, regulators, and accountants. The correct working of the financial machine relies 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], a result of the 2008 crisis, created a permissionless financial system where its users can send and receive digital money without intermediaries and without anybody being able to block transactions. In countries like Argentina, Nigeria, or Lebanon, where stagnation and inflation erode its citizens' trust in the financial system and the state, Bitcoin [1] and stablecoins on top of Ethereum [2] are used on a daily basis by the young population to save and avoid capital controls. In developed countries, its usage is not as massive since 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, the fight between the left and the right is retaking the main stage, the tension between the West and the East is increasing, and technological change keeps accelerating.

Bitcoin [1] and Ethereum [2] provide integrity through a protocol with its incentives and punishments. Integrity is provided by having each node re-execute each transaction and reach an agreement (consensus). Thus, re-execution and consensus are the basis of truth. Blockchains are verifiable computers. They invert the relationship between hardware and software. We can run code over a distributed piece of software, the blockchain, without needing to trust a physical computer it is running on.

Ethereum enables smart contract app developers to build applications that do not need to be trusted. The apps inherit the properties of the verifiable computer where they run. These applications compete for block space, which is the way blockchains provide trust. This resource is limited and puts constraints on the types of applications we can run and results in increased costs.

Alternative Layer 1s like Cosmos, Solana, Avalanche, and BNBCHAIN chose to change the parameters of the verifiable computer. They reduced the number of nodes, compromised on liveness guarantees, or increased hardware requirements to become faster and cheaper, but at the cost of reduced safety.

One of the downsides of blockchains is that adding new hardware to the system does not make them faster. This is because every node has to re-execute the computations, making less powerful devices the bottleneck of the system. Zero-knowledge proofs enabled scaling blockchains by adding more hardware and allowing complex computations to be checked much faster than naïve re-execution. The core idea in zero-knowledge proofs is that we can verify a short string (typically in the order of kB, generally much smaller than all the information needed to prove the statement) with a verification time that is at most logarithmic in the size of the computation, $\mathcal{O}(\log n)$, with n the number of steps of the computation. The consequences of such scaling were already explored in 1991 by Babai et al. [3], showing that it could be possible that a very weak but trustworthy computer checks and controls what a group of very powerful but trustless servers are doing. Even though this was clear in theory, it only became practical after 2014. Since then, we have seen a Cambrian explosion of proof systems, using different finite fields, elliptic curves, hash functions, and polynomial commitment schemes, leading to tradeoffs in proving and verification times, proof sizes, etc.

ZK Layer 2s (zk-rollups) like zkSync, Starknet, Polygon, Scroll, and Taiko have extended the capabilities of Ethereum to be faster and cheaper like alternative Layer 1s while maintaining the security guarantees of Ethereum. ZK makes more efficient use of block space, leading to reduced costs. Rollups outsource execution to a node or group of nodes, but absorb trust from Ethereum by proving computation to Ethereum via an EVM contract, using cryptoeconomic and cryptographic guarantees [4]. The dependency on escape hatches and multisigs still needs to be resolved, and it will once the technology matures. It also creates the problem of fragmented liquidity and users, resulting in the need for bridging, adding overhead, and making user experience more complicated.

Until now, you could only build apps that inherit the trust of the verifiable computer if you built the app on top of it. EigenLayer [4] enables the creation of apps that inherit the trust of Ethereum without the need to create the app on top of the blockchain itself. You can build new blockchains with different consensus mechanisms like alternative Layer 1s. EigenLayer brings innovation back to Ethereum. It also enables the building of distributed systems like bridges, Data Availability layers, MEV layers, or even ZK verification layers like Aligned Layer on top of it. It extends the capabilities of Ethereum in a different way than Layer 2s and ZK-enabled solutions.

With Bitcoin, Ethereum, ZK, and Layer 2s, EigenLayer is one of the most significant innovations in the space. Until now, blockchains have sold trust via block space. EigenLayer enables the selling and buying of trust in other forms. This solves the problem developers face when building their own infrastructure: providing their economic security [4]. 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 [4]:

- It is difficult to know where stakers are located.
- Stakers have to invest large amounts of money to obtain 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 in 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 in which validators can choose whether to opt in or out of each application built on EigenLayer [4]. Among the possible applications that can be built on top of EigenLayer we have [4]:

- 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 major source of liquidity for decentralized applications. However, Ethereum was not originally designed with zero-knowledge proofs or validity proofs in mind. Some of these design considerations introduce an additional burden to proving systems, and it is hard 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 leverage all the power of proving systems? The answer lies in creating a verification layer that can tap into Ethereum's trust via EigenLayer. This way, we can benefit from Ethereum's decentralized infrastructure without having to deal with the constraints of the EVM.

Aligned Layer 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 check any type of 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, and this has an impact on 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 increase as well.
6. Reduced latency.
7. As ZK becomes more accessible, the demand for ZK verification will soar. We have seen developments towards zkvms for general-purpose languages such as Rust, which abstract developers from the gory details of zk and just 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 type of proof, we avoid having to write a specific wrapper to make proofs EVM-friendly.

Among the possible usecases of Aligned Layer we have:

- Soft finality for Rollups and Appchains.
- Fast bridging.
- New settlement layers (use 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
- Cheap verification and interoperability for Identity Protocols
- ZK Oracles
- New credential protocols such as zkTLS based systems.
- ZK Coprocessor
- Encrypted Mempools using SNARKs to show the correctness of the encryption.
- Protocols against misinformation and fake news.
- On-chain gaming

2 Aligned architecture

2.1 Overview

As pointed out, verification is the main problem we must address. We want to be free from the constraints of the EVM to use any proof system and cryptographically secure elliptic curve, achieving a cost-efficient and scalable solution. Creating a decentralized verifier offers some challenges, and some careful consideration is needed to make it flexible enough and provide the chance for fast, soft finality while at the same time keeping the security provided by Ethereum. It should also let developers choose any proof system suited to their needs (in terms of speed, proof size, ease of development, and security considerations). We can leverage EigenLayer to tap into Ethereum's validator network to verify proofs and then use general-purpose proving systems to aggregate proofs and use Ethereum to check just one proof, avoiding competition for space and computational resources. Figure 1 shows the core elements:

- Aligned Layer: It receives proofs from different proof systems, verifies them, sends the final result to Ethereum, and posts the data into a DA layer.
- Data Availability Layer (DA): provides storage for the different proofs.
- General Prover/Verifier: Every several days, takes the proofs from the DA layer and generates a proof of the verification of all the proofs. The general prover can be based on the SP1, Risc0 or Nexus [5]'s virtual machine, which are virtual machines that can prove general Rust code. The proof of verification of the proofs is done using the corresponding verifier codes in Rust. The verification can be done using a tree structure.
- Ethereum: the source of trust and liquidity.

2.2 Interaction between the main components

Aligned Layer receives several proofs from different sources, which could be generated using different proof systems. These have varying proof sizes, verification times, and different verification logic. However, all proofs share a common characteristic: their verification is fast. Aligned Layer has dedicated verifiers, which can be used to check the validity of each of them and post the results to Ethereum. It offers two advantages over Ethereum: cheaper verification costs and the ability to easily update, including new verifiers to check proofs from newer proof systems. The users may take the verification result in Aligned to achieve a fast, soft finality with security guarantees similar to Ethereum's. They could also wait for the verification of the aggregated proof in Ethereum, achieving finality later but with all the guarantees of Ethereum.

The proofs are stored in a Data Availability Layer, which offers a cost-efficient storage strategy. The operators in Aligned fetch the proof's data from this layer.

Ethereum receives the verification results from Aligned Layer. However, Ethereum itself does not have access to verify the proofs since this would be too costly. Instead, a general Rust prover generates proof of verification for all the proofs included in a given period (Figure 2). Even if this proof can be quite ample, it is only one proof that attests to the validity of all proofs. No matter how complex the verifier is, it can be written in Rust and proven by a zero-knowledge virtual machine, such as SP1 or Risc0. To aggregate all the proofs, in the first step, all proofs are transformed into proofs of execution of the virtual machine, achieving proof uniformity (Figure 2). We can then shrink proof size by recursively proving the verification of proofs, as shown in the tree diagram, for example in figure 3.

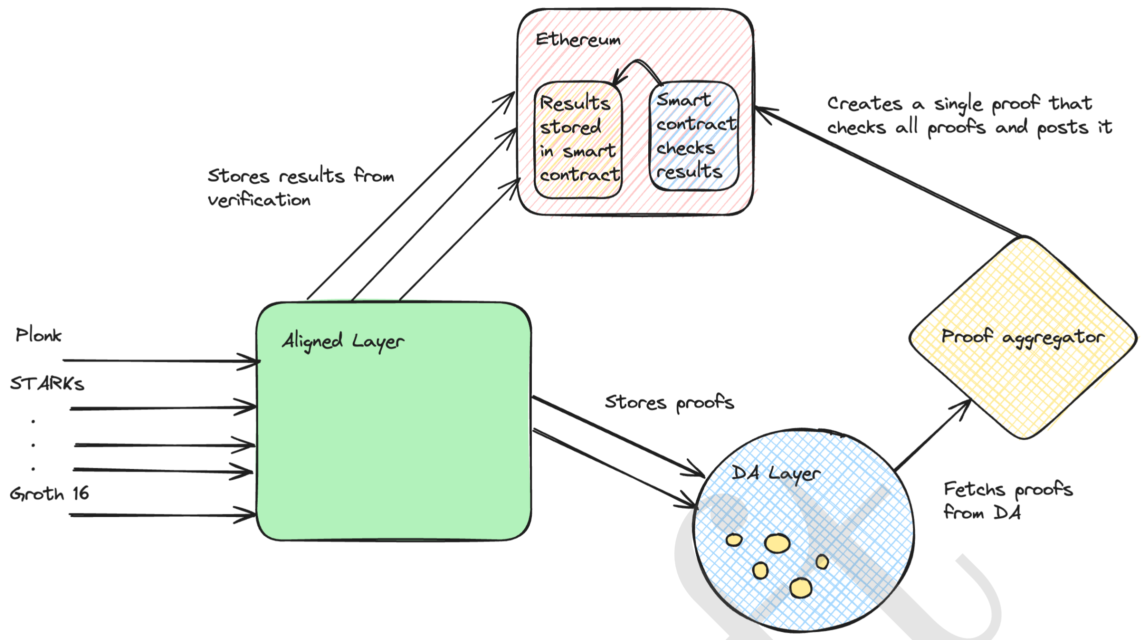
Ethereum
Results stored in smart contract
Smart contract checks results
Creates a single proof that checks all proofs and posts it
Stores results from verification
Plonk
STARKs
.
.
.
Groth 16
Aligned Layer
Stores proofs
Proof aggregator
DA Layer
Fetchs proofs from DA

Figure 1: Core components

2.3 Slashing

In order to have a proper incentive mechanism for a decentralized permissionless network we need to introduce slashing. This will allow the protocol to penalize the 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 who are not in the majority of quorum will get penalized for being against the result reached by most of the network. This is not a perfect mechanism by any means, but taking into account the client software for Aligned Layer will be lightweight, the minimum hardware requirements will be low. Therefore the network can be easily decentralized by allowing many participants to join as operators. The more decentralized the network, the more likely the majority is honest.

Long term we are exploring a slashing design that will get categorized for different proving systems. Not all proving systems integrated by Aligned will be able to get 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 system on-chain allowing this proof systems to have stronger security guarantees when used in Aligned. For example, Aligned Layer offers verification for Cairo proofs that can also be verified in Ethereum, if necessary. This means that, in case the majority 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 this would fragment the security guarantees for each type of proof but it will overall improve them for many of the used proof systems today in Ethereum.

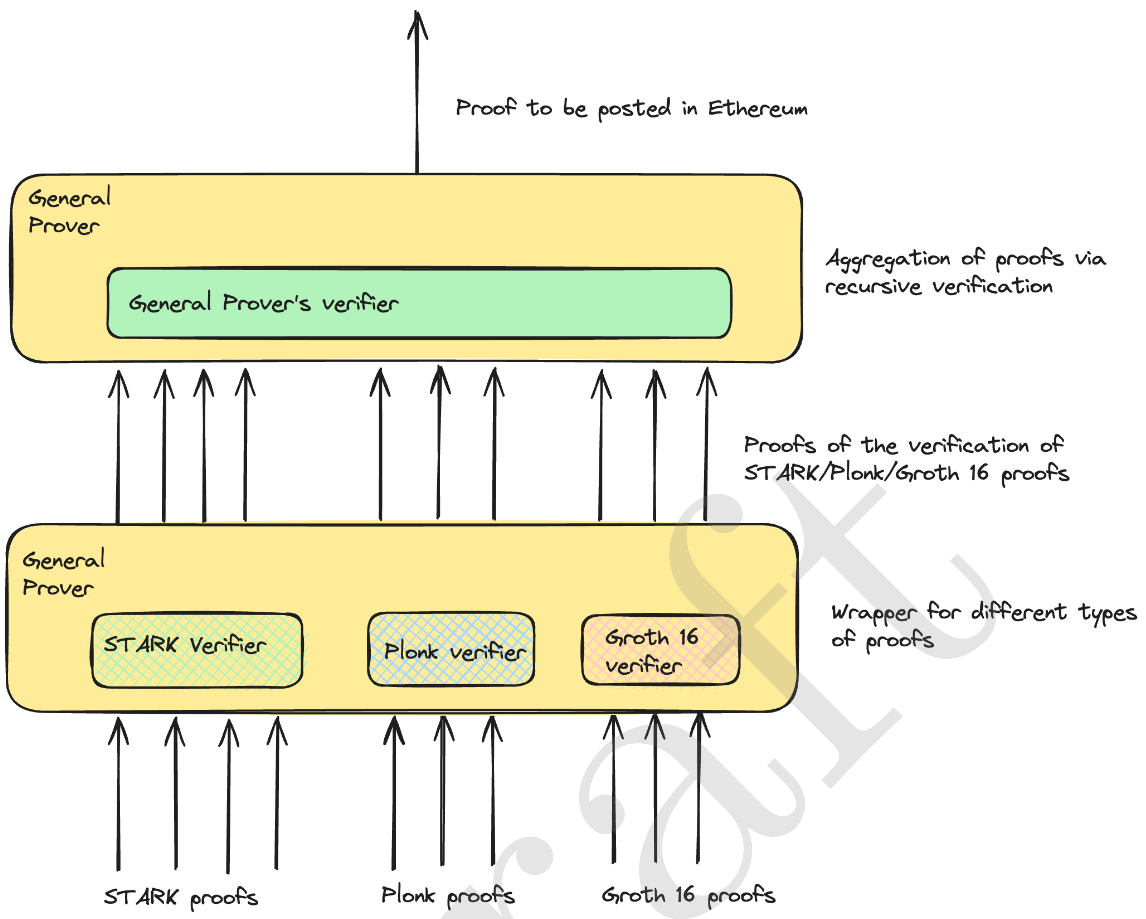
Proof to be posted in Ethereum
General Prover
General Prover's verifier
Aggregation of proofs via recursive verification
Proofs of the verification of STARK/Plonk/Groth 16 proofs
General Prover
STARK Verifier
Plonk verifier
Groth 16 verifier
Wrapper for different types of proofs
STARK proofs
Plonk proofs
Groth 16 proofs

Figure 2: General prover's flow

2.4 Dual Staking

We are proposing a dual staking model. First, we need to boostrap the proof of stake network using ETH with restaking from EigenLayer. In a second stage, as with any piece of critical infrastructure, we need a native token for governance and sovereignty. We want that the costs for compromising both liveness and security to be high. In particular governance will be necessary to decentralize the decision making of some important architectural changes such as which proof systems to include in a new version of the protocol.

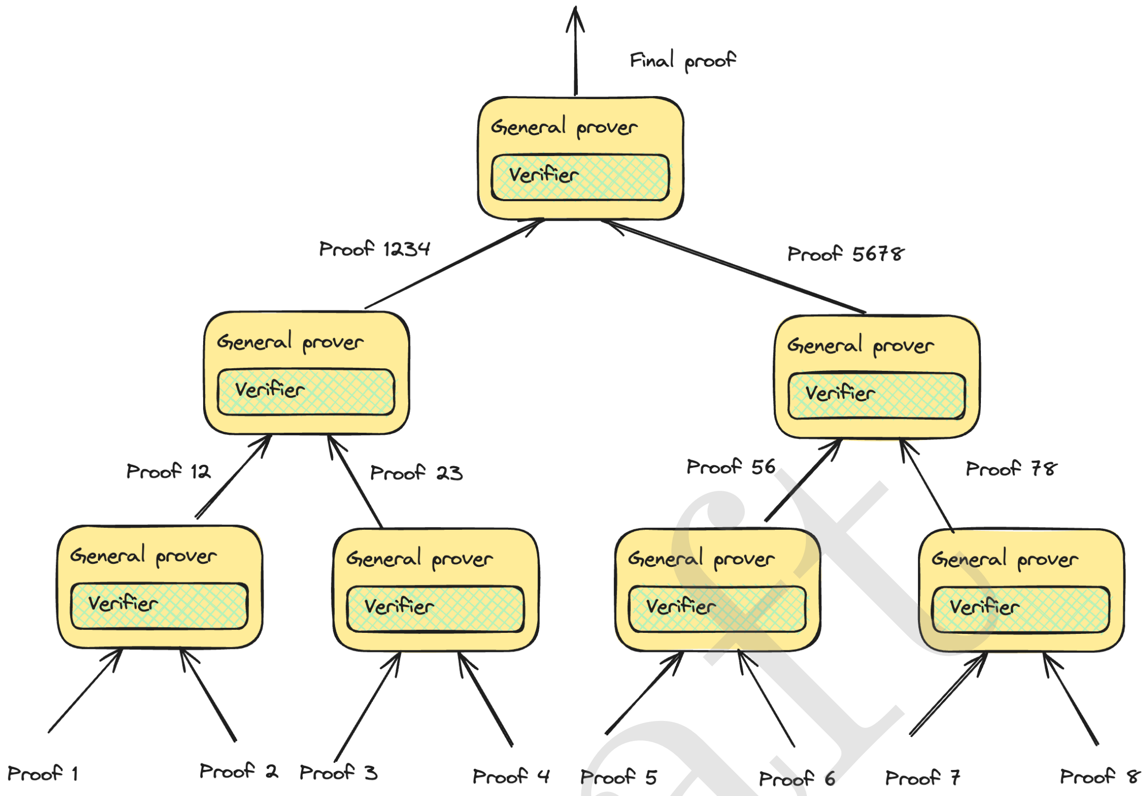

Figure 3: Recursion tree

2.5 Basic flow for Aligned Layer

The diagram in figure 4 shows a basic flow for Aligned Layer. The task manager posts the proof in the DA Layer and creates a new task in Ethereum, sending the hash of the proof and required metadata. The operators fetch the task from Ethereum and the proof from the DA Layer. The operators then send the proof verification results to the aggregator. The aggregator validates the results, aggregates them, and posts them on Ethereum and could also post them to other chains.

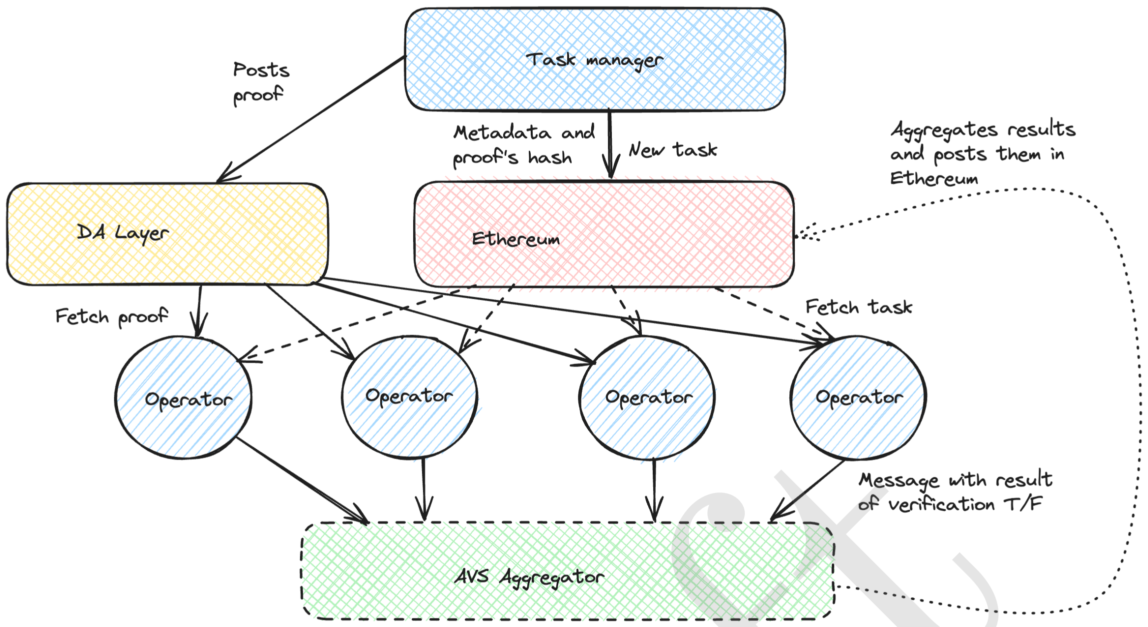

Figure 4: Basic flow for Aligned Layer

2.6 Comparing verification in Aligned and Ethereum

Table 1 compares verification in Aligned using soft finality, Aligned with hard finality in Ethereum and Ethereum. Soft finality is the cryptoeconomic finality, whereas hard finality is the cryptographic finality in Ethereum by verifying the aggregated proof. Aligned offers a more cost-effective alternative with practically the same security guarantees as Ethereum. Using soft finality reduces latency, making the user experience better and bridging different chains simpler and faster.

Variable	Aligned with soft finality	Aligned with hard finality in Ethereum	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

3 Summary on proof systems

Even though zero-knowledge proofs first appeared in 1985 [6], it was not until 2014 that we had practical systems for general computation [7]. Initially, these proving systems relied on pairing-friendly elliptic curves and needed a trusted setup per program. The appearance of transparent proof systems (STARKs [8]) removed the obstacle of the trusted setup at the expense of larger proofs. We will try to summarize the characteristics of several proof systems in table 2.

It is important to note that the table shows only the original version in the paper; prover times were estimated from Thaler [14] and is just a rough estimate. However, by choosing different

System	Groth 16[9]	Plonk+KZG[10]	STARKs[8]	HyperPlonk[11]	Binius[12]
Proof size	Constant	Constant	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{n})$
Verification time	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(\ell)$	$\mathcal{O}(\sqrt{n})$
Arithmetization	R1CS	Plonkish	AIR	Plonkish	Plonkish
Polynomials	univariate	univariate	univariate	multivariate	multivariate
Field	Curve-specific	Curve-specific	small fields	Curve-specific	Binary
Post quantum?	No	No	Yes	No	Yes
Prover time	$\mathcal{O}(n \log n)$[13]	$\mathcal{O}(n \log n)$[11]	$\mathcal{O}(n \log n)$[11]	$\mathcal{O}(n)$[11]	$\mathcal{O}(n)$[11]

Table 2: Comparison proof systems. ℓ is the size of the public inputs and n the length of the program/number of gates

polynomial commitment schemes, we can have conjectured post-quantum secure proof systems (such as Plonk [10] with FRI [8] as commitment scheme) or use different fields.

To generate proofs, one needs to transform the computation into a system of polynomial equations that must be fulfilled. At least 3 different types are commonly used:

- Quadratic Rank One Constraint System (R1CS)
- Plonkish
- Algebraic Intermediate Representation (AIR)
- Customizable Constraint System (CCS) [15], generalizing the above

To check the enforcement of the constraints, the variables are encoded either as univariate polynomials, $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ or multivariate polynomials, $p(x_0, x_1, x_2) = a_0 + a_1 x_0 + a_2 x_1 + a_3 x_0 x_1 + \cdots + a x_0^a x_1^b x_2^c$

The polynomial's coefficients live in some finite field, $\mathbf{F}$. The field's size is related to security and performance. The field's size should be large enough so that we get enough security (at least 128 bits), but large fields hurt performance for two reasons:

1. We have to represent every variable as a field element, which means that if we have boolean variables, we have to represent them by a field element.
2. Operations tend to be slower for larger fields.

When using elliptic curves, the underlying curve determines the fields over which the equations hold. We can choose smaller fields for hash-based systems, especially with nice reduction formulae, such as $2^{64} - 2^{32} + 1$. Mersenne primes have very efficient arithmetic but lack some nice structure for fast Fourier transform (FFT). Recent developments in field agnostic [16] proving systems and circle STARKs[17] could leverage the advantages of Mersenne primes for proving, such as $2^{31} - 1$. Binary fields have also gained popularity due to their hardware friendliness. Binius [12] is a new proving system that works with these fields, achieving fast provers, at the expense of larger proofs.

Several advances have also been made in the field of lookup arguments [18], with new techniques improving performance and achieving table-independent lookups. The use of classical protocols like GKR could boost the performance of these lookups [19], which will make proving complex operations cheaper.

All projects have been developing protocols for proof aggregation [20] to reduce costs: having just one proof whose verification is equal to the verification of all separate proofs. The simplest way to achieve this is via proof recursion: we prove that we verified some proofs. STARKs and SNARKs based on cycles of curves [21] have some advantages for recursion, especially after the introduction of lookup arguments. They often use one final recursive SNARK (a constant-sized SNARK such as Groth 16) to compress the proof size further.

It is clear that we have a wide range of options for proving, involving different tradeoffs in terms of proof size, proving and verifying times, ease of development, etc. It is not the same if we want to prove general programs as just having a few programs. Having dedicated and optimized circuits can be advantageous in terms of performance but they require expert knowledge and take more time to develop. We think developers should be free to use the proof system that better suits their needs and not be constrained by the settlement layer. Aligned Layer solves these problems by handling verification in any proof system, offering fast, soft finality, and aggregating proofs for final verification in Ethereum.

4 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 re-execution: verification of short cryptographic proofs. Leveraging this technology, rollups helped solve the scalability issues, while introducing fragmented liquidity and users, requiring bridging and making verification costs dependent on gas prices. EigenLayer allows developers to create new protocols that can tap into Ethereum's trust network. We propose Aligned Layer, an infrastructure dedicated to proof verification and aggregation, tapping into Ethereum's security via EigenLayer. Aligned Layer allows for the easy introduction of different proof systems, making it easy for developers to use proving systems taylored to their needs. The system allows for fast, soft finality, which reduces latency, lower bridging costs, making user experience simpler and better.

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] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, page 21–32, New York, NY, USA, 1991. Association for Computing Machinery.

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

[5] Daniel Marin, Michel Abdalla, Dan Dore, Paul Govereau, Jens Groth, Samuel Judson, Kristian Sosnin, Guru Vamsi Policharla, and Yinuo Zhang. Nexus 1.0: Enabling verifiable computation, 2024. `https://www.nexus.xyz/whitepaper.pdf`.

[6] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[7] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. Cryptology ePrint Archive, Paper 2013/279, 2013. `https://eprint.iacr.org/2013/279`.

[8] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

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

[10] 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.

[11] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. Cryptology ePrint Archive, Paper 2022/1355, 2022. https://eprint.iacr.org/2022/1355.

[12] 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.

[13] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Paper 2019/1047, 2019. https://eprint.iacr.org/2019/1047.

[14] Justin Thaler. *Proofs, Arguments and Zero-Knowledge.* 2023.

[15] Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Paper 2023/552, 2023. https://eprint.iacr.org/2023/552.

[16] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic snarks for r1cs. Cryptology ePrint Archive, Paper 2021/1043, 2021. https://eprint.iacr.org/2021/1043.

[17] Ulrich Haböck, David Levit, and Shahar Papini. Circle starks. Cryptology ePrint Archive, Paper 2024/278, 2024. https://eprint.iacr.org/2024/278.

[18] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Paper 2020/315, 2020. https://eprint.iacr.org/2020/315.

[19] Shahar Papini and Ulrich Haböck. Improving logarithmic derivative lookups using gkr. Cryptology ePrint Archive, Paper 2023/1284, 2023. https://eprint.iacr.org/2023/1284.

[20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. Cryptology ePrint Archive, Paper 2020/499, 2020. https://eprint.iacr.org/2020/499.

[21] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. Cryptology ePrint Archive, Paper 2014/595, 2014. https://eprint.iacr.org/2014/595.