EigenLayer: The Restaking Collective

EigenLayer Team

Abstract

We propose EigenLayer, a restaking collective for Ethereum. EigenLayer is a set of smart contracts on Ethereum that allows consensus layer Ether (ETH) stakers to opt in to validating new software modules built on top of the Ethereum ecosystem. Stakers opt in by granting the EigenLayer smart contracts the ability to impose additional slashing conditions on their staked ETH, allowing an extension of cryptoeconomic security. By opting in to EigenLayer, stakers can validate for many types of modules including consensus protocols, data availability layers, virtual machines, keeper networks, oracle networks, bridges, threshold cryptography schemes, and trusted execution environments. Instead of fragmenting security between modules, EigenLayer aggregates ETH security across all of them. This increases the security of the decentralized applications (DApps) that rely on the modules. In addition, holders of Ether can pursue additional value through the new fee-generating opportunities from these manifold modules. EigenLayer also acts as a staging system for Ethereum, where new innovations such as Danksharding and proposer/builder separation can be battle-tested in manifold variations before the best ideas can be integrated back into Ethereum. Finally, EigenLayer ushers in a new era of permissionless innovation, in which innovators do not need to build their own trust networks in order to implement new distributed validation modules, but can instead rely on the security and decentralization provided by ETH restakers via EigenLayer.

1 The Problem: Fractured Trust Networks

Bitcoin. Bitcoin pioneered the notion of blockchains, heralding an era of decentralized trust [1]. However, Bitcoin was designed to be application-specific, focusing only on peer-to-peer payments. Hence, any new decentralized application also required a new blockchain, with its own trust network. Decentralized trust is very difficult to generate and maintain, and hence, it was very difficult for new decentralized applications to be built in this model.

Ethereum. Ethereum was conceived in 2013 and launched in 2015, and changed the landscape significantly [2]. By being fully programmable on the Ethereum Virtual Machine (EVM) [3], Ethereum **pioneered the concept of modular blockchains**, where distributed applications (DApps) became modules that could be built permissionlessly on top of the Ethereum trust network. The trust network underlying Ethereum provides pooled security to all of the DApps on top of it. This **decoupled innovation and trust**: innovation can come from any DApp developer, but the developer need not be trusted for the safety and liveness of execution of the DApp, since trust is underwritten by the blockchain. The value flow is that the blockchain supplies trust to the DApp, and in exchange receives back fees. We note that this decoupling is a key driving force underpinning the pseudonymous economy, since innovators do not require any reputation or trust, and the DApp can be used by anyone who trusts the underlying blockchain and can verify the DApp code.

Ethereum Layer-2 Era. The set of applications that can be built on top of Ethereum permissionlessly expanded significantly when Ethereum switched to a rollup-centric roadmap [4]. Rollups outsource execution to a single node or a small group of nodes, but can absorb Ethereum trust by proving computation to Ethereum via an EVM contract, either using cryptoeconomic guarantees (via fraud proofs, in which case such rollups are called "optimistic rollups") or cryptographic guarantees (via 'succinct validity proofs,' in which case such rollups are often called ZK-rollups) [5]. This has led to a massively increased rate of permissionless innovation in rollup technology, leading to a proliferation of a variety of proving technologies.

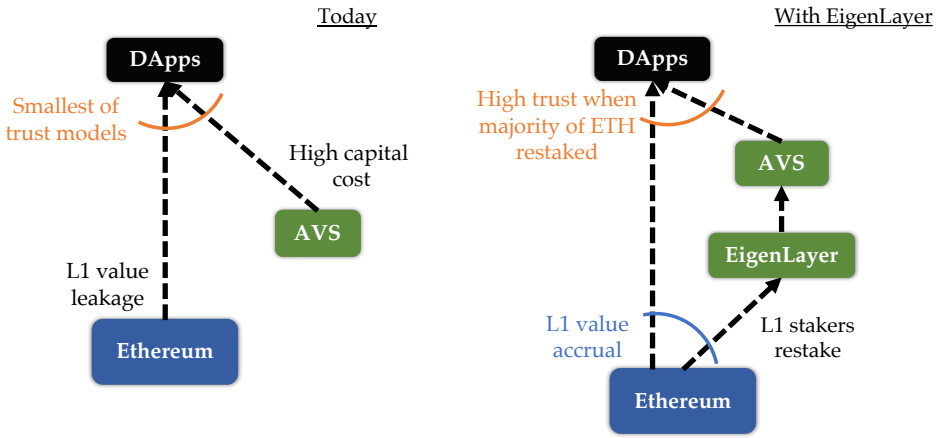


Figure 1: Comparing the ecosystem of actively validated services today and with EigenLayer.

Limitations. However, any module that cannot be deployed, or proven, on top of the EVM, cannot absorb the pooled trust of Ethereum. Such modules involve processing on inputs that are derived from outside Ethereum and as such their processing cannot be validated in-protocol inside Ethereum. Examples of such modules include sidechains based on new consensus protocols, data availability layers, new virtual machines, keeper networks, oracle networks, bridges, threshold cryptography schemes, and trusted execution environments. Typically, such modules require actively validated services that have their own distributed validation semantics to do verification. Usually, these actively validated services ("AVS") are either secured by their own native token, or are permissioned in nature.

There are four basic downsides to the present organization of the AVS ecosystem.

1. **Bootstrapping problem for a new AVS**. Innovators wishing to develop a new AVS have to bootstrap a new trust network in order to get security.
2. **Value leakage**. With each AVS developing its own trust pool, users have to pay fees to these pools on top of transaction fees to Ethereum. This diversion in flow of fees results in value leakage from Ethereum.
3. **The burden of capital cost**. Validators who stake to secure a new AVS must **incur capital cost**, which is equivalent to the opportunity cost and price risk associated with staking in a new system. The AVS must therefore provide a high enough staking return in order to cover this cost. For most AVSs in operation today, the capital cost of staking far dominates any operational cost. For example, consider a Data Availability layer with $\$10B$ of stake securing it, and assume that the annual percentage return (APR) expected by stakers is 5%. This AVS would need to pay back at least $\$0.5B$ annually to stakers in order to compensate for capital cost. This is significantly greater than the operational costs associated with data storage or networking costs.
4. **Lower trust model for DApps**. The current AVS ecosystem gives rise to a highly undesirable security dynamic: generally speaking, any one of a DApp's middleware dependencies may be the target of an attack. Thus, the cost of corruption (see Section 3.1 for a description) of a DApp generally must be taken to be no more than the **minimum** cost to corrupt at least one of its dependencies. See fig. 3a for an illustration. In a world in which an application relies on a critical module such as an Oracle with a small amount of stake securing it, the powerful economic security guarantees provided by Ethereum may not mean much, as the cost to attack the Oracle is much lower than the cost to attack Ethereum.

2 EigenLayer: The Restaking Collective

EigenLayer introduces two novel ideas, **pooled security via restaking** and **free-market governance**, which serve to extend the security of Ethereum to any system and to eliminate the inefficiencies of existing rigid governance structures:

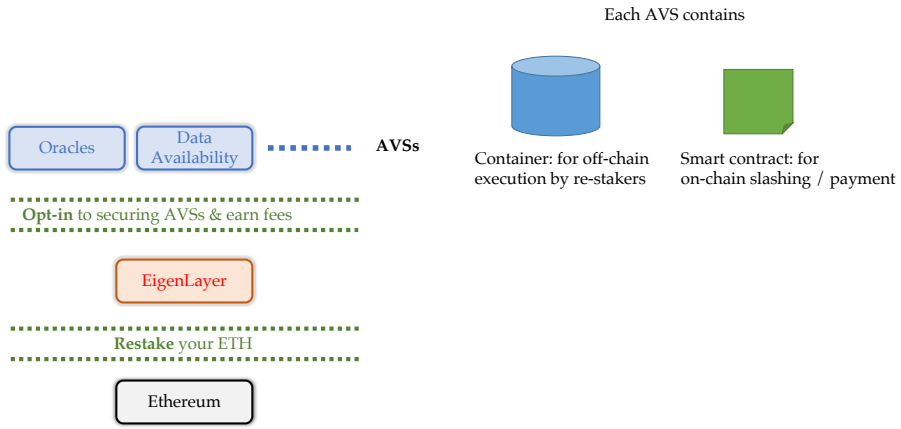


Figure 2: Launching an AVS on top of EigenLayer requires deploying an off-chain container that operators must download, and an on-chain contract that specifies the terms of slashing and payment.

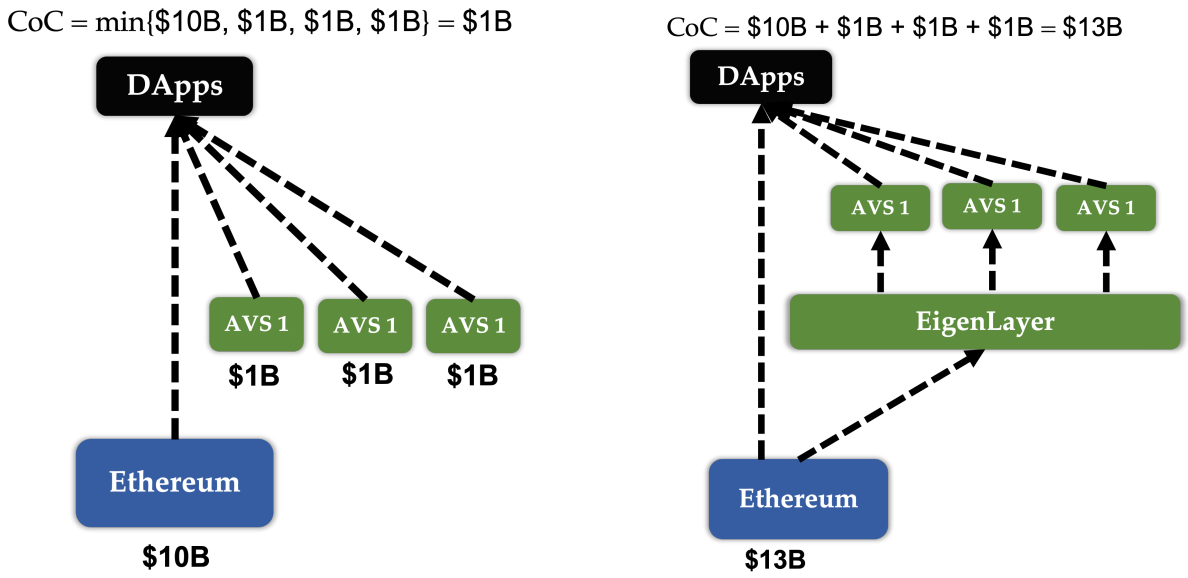
1. **Pooled security via restaking**. EigenLayer provides a new mechanism for pooled security by enabling modules to be secured by restaked ETH rather than their own tokens. In particular, Ethereum validators can set their beacon chain withdrawal credentials to the EigenLayer smart contracts, and opt into new modules built on EigenLayer. The validators download and run any additional node software required for these modules. The modules then have the ability to impose additional slashing conditions on the staked ETH of validators who opted into the module. We call this mechanism restaking. In return, validators gain additional revenue from providing security and validation services to their chosen modules. When coupled with an on-chain verifiable slashing mechanism, this mechanism of restaking allows for a deep transfer of cryptoeconomic security, which we discuss at length in Section A. See Fig. 2 for an illustration. For example, if the module is a Data Availability layer, restakers via EigenLayer will receive a payment whenever data is stored through the module. In return, restakers are subject to slashing conditions that are exercised through proof-of-custody. As illustrated in Fig. 4, restaking vastly expands the space of blockchain applications over which security can be pooled. Thus EigenLayer expands open innovation beyond the smart contract-based DApps enabled by Ethereum, to virtual machines, consensus protocols, and middleware. Any AVS with an on-chain slashing contract can be secured by EigenLayer.

2. **Open marketplace**. EigenLayer provides an open market mechanism which governs how its pooled security is supplied by validators and consumed by AVSs. EigenLayer creates a marketplace in which validators can choose whether to opt in or out of each module built on EigenLayer, as pictured in Fig. 4. The various modules will need to sufficiently incentivize validators to allocate restaked Eth to their module and validators will help determine which modules are worthwhile to assign this additional pooled security given the potential for additional slashing. The opt-in dynamics of EigenLayer have two important benefits: (1) the stable, conservative governance of the core blockchain is complemented with a fast and efficient, free-market governance structure for launching new auxiliary capabilities; and (2) opt-in validation makes it possible for new blockchain modules to exploit heterogeneous resources among validators, resulting in better tuned trade-offs of security and performance. We explore these benefits further in Section 4.

By combining these ideas, EigenLayer serves as an **open marketplace where AVSs can rent pooled security provided by Ethereum validators**.

EigenLayer solves the various problems in the AVS ecosystem highlighted above:

1. **Bootstrapping problem for a new AVS**: A new AVS can bootstrap security from the large validator set of Ethereum.



(a) AVS economics today. To corrupt the AVS, the attacker need only attack one of the modules which are secured by a stake of $1B; that is significantly less than the $10B of stake securing the L1.

(b) AVS economics of pooled security. Corrupting the DApp here would require the attacker to attack the pooled stake of $13B.

Figure 3: Pooled security of EigenLayer.

2. **Capital Cost**: Since ETH stakers reuse their capital across multiple services, their capital cost gets amortized. In particular, the marginal capital cost of native ETH stakers opting in to EigenLayer is minimal (theoretically zero, if there is no risk for an honest node to be slashed).

3. **Trust aggregation**: Since there is a larger pool of capital restaked, the trust model is much better. For example, consider the scenario in Fig. 3b in the EigenLayer world. Assuming that all L1 stake is restaked to all three AVS modules, the cost of corrupting the DApps is the total amount staked in the L1 itself. We note that due to the **additive yield opportunity** from the three AVSs, the total amount staked in the L1 with the presence of EigenLayer is now equal to the separate amounts that were staked with the L1 and each of the AVSs in the world without Eigenlayer. So, in the above example, the total amount staked in the L1 with the presence of EigenLayer is $13B$\$. Thus EigenLayer massively increases the cost-of-corruption, from the minimum-of-the-stake to the sum-of-the-stake.

4. **Value Accrual**: EigenLayer gives ETH stakers several additional revenue streams that they can participate in, and further consolidates the ecosystem's network effects due to the presence of a highly secure AVS ecosystem.

2.1 EigenLayer Enables Multiple Staking Modalities

We compare the various staking related paradigms such as liquid staking, superfluid staking, and restaking here. First, we begin with existing schemes shown in Fig. 5a.

- **Liquid Staking**. Liquid staking services such as Lido [6] and Rocket Pool [7] allow users to deposit their ETH to a staking pool, and receive a liquid staking token (LST) that represents a claim on their ETH and its staking yield. Within the staking pool, the ETH is delegated to one of many validators who are participating in the consensus protocol. LSTs will be redeemable for their underlying ETH value after the Shapella upgrade, though redemption will be subject to a waiting period equal to the ETH staking withdrawal period. LSTs can also be traded in the DeFi ecosystem via exchanges like Uniswap and Curve.

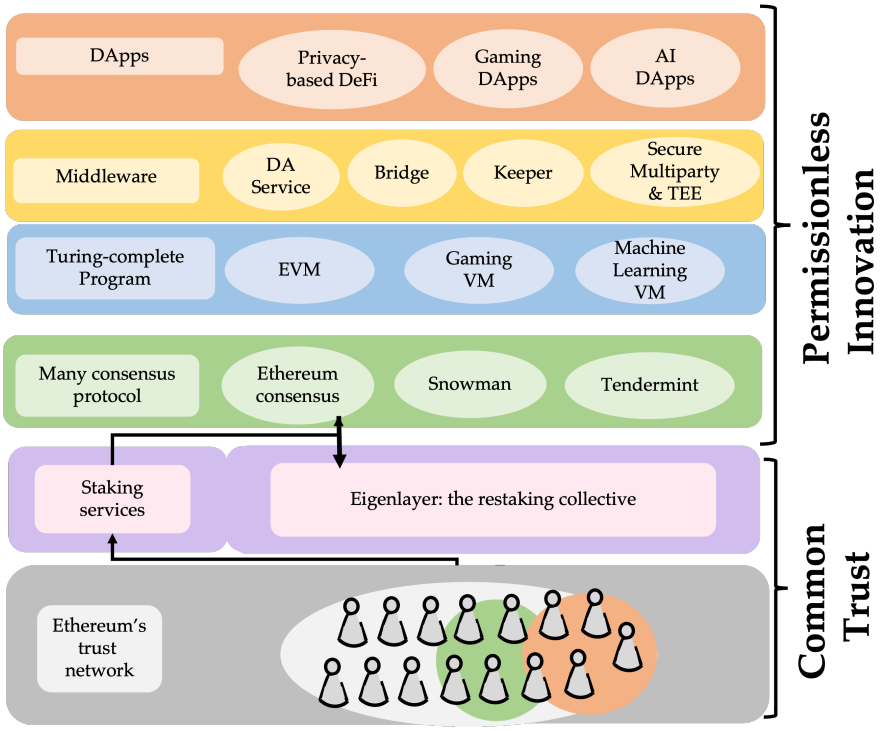


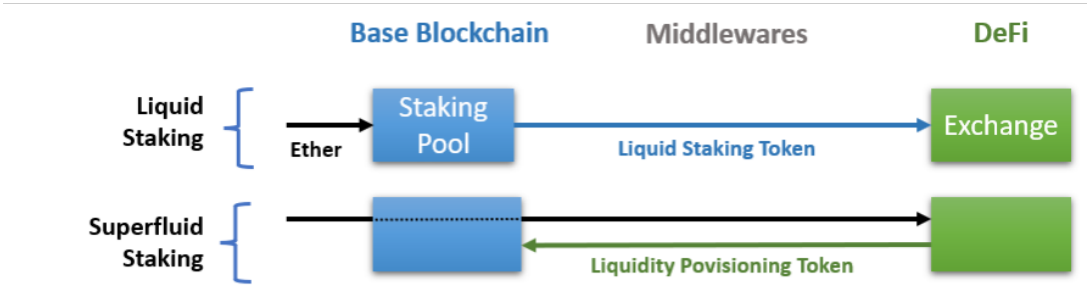
Figure 4: With EigenLayer, permissionless innovation permeates deeper into the blockchain stack.

- **Superfluid Staking**. Superfluid staking reverses the ordering of liquid staking by modifying the core consensus protocol to allow the staking of Liquidity Provisioning (LP) tokens [8]. An LP token represents a share of the total liquidity contained in a DeFi exchange such as Uniswap or Curve.

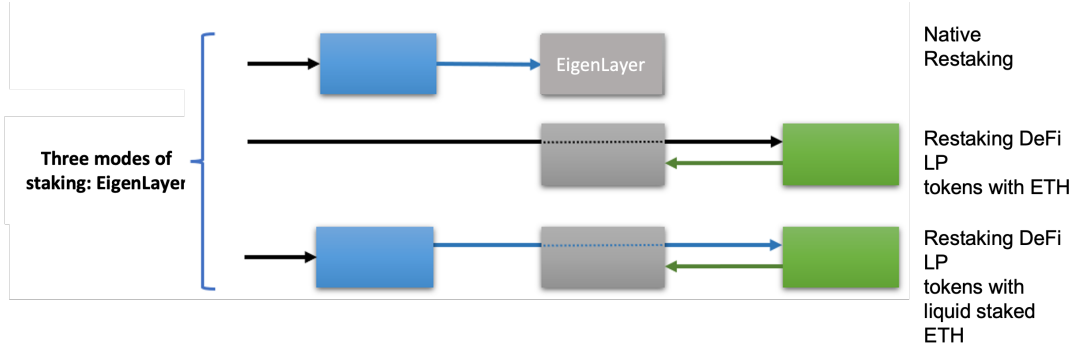
As shown in Fig. 5b, EigenLayer provides multiple pathways for yield stacking which allow stakers to earn additional yield from securing new AVSs. Broadly we can think of three distinct layers of the blockchain: core protocol, AVS, and DeFi. Liquid staking can be thought of as stacking yield by first going to the core protocol and then the DeFi layer. Superfluid staking can be thought of as first going through the DeFi layer before going to the core protocol layer. In EigenLayer, there can be several modalities of restaking:

1. **Native restaking**. Validators can restake their staked ETH natively by pointing their withdrawal credentials to the EigenLayer contracts. This is equivalent to L1 $\rightarrow$ EigenLayer yield stacking.
2. **LST restaking**. Validators can restake by staking their LSTs, ETH already restaked via protocols like Lido and Rocket Pool, by transferring their LSDs into the EigenLayer smart contracts. This is equivalent to DeFi $\rightarrow$ EigenLayer yield stacking.
3. **ETH LP restaking**. Validators stake the LP token of a pair which includes ETH. This is equivalent to DeFi $\rightarrow$ EL yield stacking.
4. **LST LP restaking**. Validators stake the LP token of a pair which includes a liquid staking ETH token, such as Curve's stETH-ETH LP token, thus taking the L1 $\rightarrow$ DeFi $\rightarrow$ EL yield stacking route.

Each of these pathways comes with different types of risks. In keeping with the principle of opt-in governance, EigenLayer outsources the management of such risks to module developers. Developers choose for themselves which tokens to accept as stake for their AVS. They may also choose whether there is preferential weighting for rewards they distribute to different types of staked tokens. For example, a module interested primarily in decentralization may only accept restake in the form of natively restaked ETH.



(a) Liquid staking = L1 → DeFi. Superfluid staking = DeFi → L1



(b) (1)Native restaking or staking of LST token: L1 → EL. (2) Staking LP tokens with ETH: DeFi → EL. (3) Staking LP tokens with LST: L1 → DeFi → EL.

Figure 5: Staking patterns.

2.2 Delegation in EigenLayer

Many of the EigenLayer restakers holding ETH or LSTs may be interested in participating in EigenLayer, but may not want to act as EigenLayer operators themselves. EigenLayer provides an avenue for these restakers to delegate their ETH or LSTs to other entities who are running EigenLayer operator nodes. EigenLayer operators who have stake delegated to them can deposit the delegated stake to spin up new Ethereum validator nodes, and subject the delegated stake to slashing from the modules the operator is participating in. These operators receive fees from both the Ethereum beacon chain and the modules they are participating in via EigenLayer. They keep a fraction of those fees and send through the remainder to the delegators.

Solo Staking: In this model, solo stakers, who are natively restaked, have two options for participating in EigenLayer: (1) solo stakers can can opt in to AVSs on EigenLayer for which they can provide validation services directly; or (2) solo stakers can delegate EigenLayer operations to a different entity, while continuing to validate for Ethereum themselves. This latter option allows for home stakers with non-upgradeable / lightweight setups to continue contributing decentralization and censorship resistance to Ethereum (as well as not sharing the core Ethereum yield with any operator), while receiving additional rewards through EigenLayer via delegation to another operator. We envision that there will be many services built on EigenLayer that are designed to be lightweight, and suitable for home stakers who do not want to delegate their stake to another operator. For example, consider a decentralized price oracle, which is computationally easy to run, but requires high trust.

Delegation Model: The delegation model in EigenLayer requires that delegate their stake to an operator. If their operator doesn't fulfill its obligations in the EigenLayer modules that it is participating in, then their deposited stake will be subject to slashing. Restakers who have delegated their stake with this operator will be slashed too. Hence, EigenLayer restakers should only delegate to trusted operators who have a track record of successfully fulfilling their obligations. There are no incentives built into EigenLayer for delegating, but is possible for others to build innovative delegation frameworks

on top of EigenLayer.

Fee model: In addition, EigenLayer restakers need to take into consideration the ratio of fees that operators share back to delegators. Given that there will be many operators to which a restaker can delegate its ETH or LSTs, this will organically give rise to a free market of delegation between restakers and operators in EigenLayer. Each EigenLayer operator will put up a delegation contract on Ethereum that specifies how fees would be split back to delegators, and that delegation contract will route fees accordingly.

3 EigenLayer: Slashing Analysis

3.1 Slashing Mechanism

Cryptoeconomic security quantifies the cost that an adversary must bear in order to cause a protocol to lose a desired security property. This is referred to as the Cost-of-Corruption (CoC) [9]. When CoC is much greater than any potential Profit-from-Corruption (PfC), we say that the system has robust security. Cryptoeconomic security stands **in contrast with systems that provide majority-trust security guarantees** which only hold under the assumption that at least a threshold percentage of operators are altruistic and will act honestly. A core idea of EigenLayer is to provision cryptoeconomic security through various slashing mechanisms which levy a high cost of corruption.

A key function of the EigenLayer smart contracts is to hold the withdrawal credentials of Ethereum Proof-of-Stake (PoS) stakers. If a staker who is restaked on EigenLayer is proven to have behaved adversarially while participating in an AVS, then that staker's ETH will be subject to slashing and are frozen, that is, prevented from further participation on any AVS on EigenLayer. Since the withdrawal address of the staker is set to the EigenLayer contracts, when the staker withdraws their ETH from participation in Ethereum consensus through EigenLayer, the withdrawn ETH will be slashed according to the on-chain slashing contract of the AVS.

3.2 No Fungible Position on EigenLayer

EigenLayer does not issue a fungible token representing restaked positions, as each restaker may opt in to validating different combinations of modules and thus be subject to different sets of slashing risks. Ensuring that such risks remain transparent to holders of a fungible position is tricky and can create principal-agent problems between holders of the fungible positions and the operators who run the nodes. This needs careful management; therefore, EigenLayer does not plan to issue fungible positions.

3.3 Comparison with Merge Mining

The restaking concept of EigenLayer is similar to the notion of merge mining which has been used for Bitcoin/Namecoin, Bitcoin/Elastos, Bitcoin/RSK and Litecoin/Dogecoin [10]. For Proof-of-Work (PoW) blockchains, the dominant cost to validators is the cost of mining. As illustrated in Fig. 6, merge mining amortizes this cost across many blockchains by using the same cryptographic PoW to mine blocks across the chains simultaneously. For PoS blockchains, the dominant cost to validators is the cost of staked capital. Restaking spreads this cost over multiple modular layers.

However, the similarities between merge mining and restaking end there. With respect to security, restaking and merge mining are very different; while merge mining creates potential security vulnerabilites, restaking **reinforces security**. To understand why this is the case, we need to view PoW and PoS blockchains through the lens of cryptoeconomics. For both PoW and PoS chains, we consider the scenario in which a small subset of main chain validators is also serving as validators for a separate chain — in the case of a PoW chain, via merge mining; and in the case of a PoS chain, via restaking. Suppose that this subset acts maliciously to attack the smaller chain, e.g. by signing an incorrect state

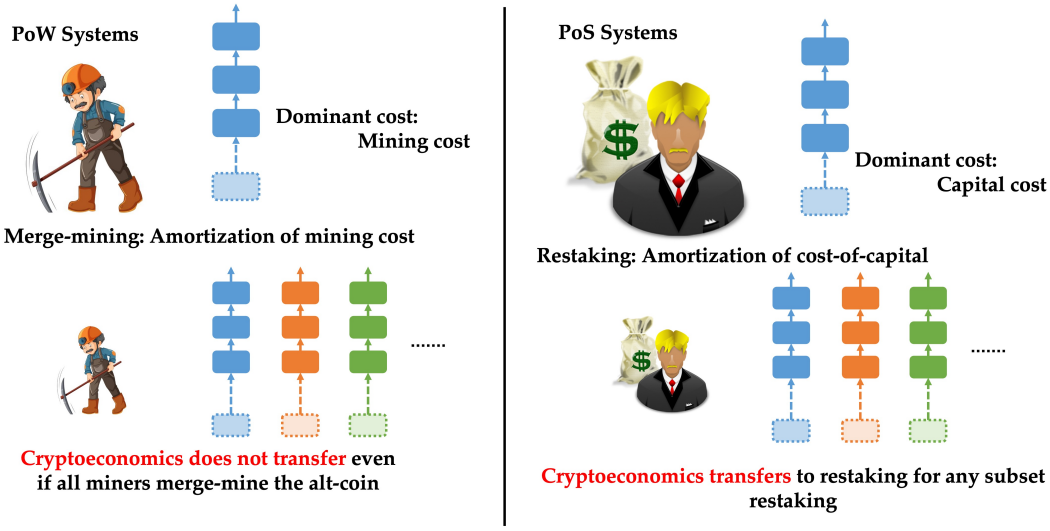


Figure 6: Merge mining vs. restaking.

transition in order to transfer and bridge out some of the chain's assets. Let us consider the outcomes for each type of chain:

- **Restaking transfers cryptoeconomic security to arbitrary subsets**. In the case of PoS chains merged via restaking, the following recourse is possible: The incorrect state transition could be fraud-proofed on the main chain, whereupon the stake held by the malicious validators in the main chain would be slashed. This sort of escalation to the main chain would induce a cost of corruption proportional to the amount restaked into the small chain. We note that we do not require all ETH stakers to participate in EigenLayer to ensure cryptoeconomic security to transfer, since any subset of stakers imposes the corresponding cryptoeconomic cost.
- **Merge mining does not transfer cryptoeconomic security**. For PoW chains, even if all the miners of the main-chain opt in to the merge-mined chain, there is no significant cryptoeconomic security. Firstly, the option of slashing—which would amount to the disabling or removal of the malicious miners' mining hardware — is not available. Moreover, even if the smaller chain loses value due to token toxicity, i.e., there is a loss of value due to the merge-mined chain now losing utility, the hardware capital of miners would continue to have value due to the existence of the main chain whose token would not experience comparable toxicity.

3.4 Risk Management

We address two categories of risks in EigenLayer: (1) many operators may collude to attack a set of AVSs simultaneously; (2) the AVSs built on EigenLayer may have unintended slashing vulnerabilities — this is the risk of honest nodes getting slashed. We analyze these two risks separately.

3.4.1 Operator collusion

We first note that in the ideal case where **all operators restake into all AVSs**, the cost of corrupting any AVS built on EigenLayer is now proportional to the total amount of stake in EigenLayer. This is the best case one can hope for in terms of maximizing the cost of corruption. However, in a realistic case where only a subset of operators opt in to a given AVS, then there are complex attacks where some set of operators may collude to steal funds from a set of AVSs.

Consider an AVS which is secured by $8M of restaked ETH and which contains a total locked value of $2M. With a quorum of 50% required to capture the $2M of locked value, the application appears to be secure, since a successful attack would result in least $4M of the attacker's stake being slashed.

However, this may not be the case if the same set of stakers are also restaking in other AVSs. In the simplest case, exactly the same set of restakers participates in 10 other AVSs, each of which have $2M locked. Thus the total profit from corrupting this group of restakers is 20M$ but the total value at stake is only 8M$ thus making the system cryptoeconomically insecure.

One important fact to observe is that even if the attack is objectively attributable, we cannot expect Ethereum to perform a hard fork in order to compensate affected parties, especially when only a small fraction of ETH stakers are participating in restaking. Of course, if a lot of ETH stakers behave maliciously in a way that causes some critical AVS to fail, then the community may coordinate a hard fork.

One solution is to restrict the PfC of any particular AVS. For example, (1) a bridge can restrict the value flow within the period of slashing, (2) an oracle can have bounds on the total value transacted within the period, etc. However, this solution depends on the designer of those AVSs. The other solution is where EigenLayer can actively increase the CoC for corrupting AVSs. Here, we have a generalized analysis of restaking security where we consider that any set of stakers can collude. If the set forms a majority quorum on certain AVSs, they can potentially extract a PfC from those AVSs. We have a mechanism to determine whether an operator or a set of operators, who are restaked with EigenLayer, are potentially at risk of creating a security vulnerability via some collusion or not. By creating an open-source cryptoeconomic dashboard, we will allow AVSs built on EigenLayer to monitor whether the set of operators participating in their validation tasks is entrenched across many other AVSs or not. If so, the AVS can put a specification in its service contracts that incentivizes only EigenLayer operators who are participating in only a low number of AVSs. We can thus think of EigenLayer as having elastic security. We refer the reader to Section B for a detailed analysis of the cryptoeconomic risk.

3.4.2 Unintended slashing

We note that, like any well-developed protocol, the goal of AVSs built on EigenLayer is to gradually ossify once the AVS is battle-tested. Once an AVS ossifies, we assume that the risk of unintended slashing will be minimal. However, before the AVS and its related infrastructure and contracts are battle-tested, there are various slashing risks that need to be mitigated in EigenLayer in order to avoid risk cascades. One risk is the case in which an AVS is created with an unintentional slashing vulnerability (for example, a programming bug) which gets triggered and causes loss of funds to honest users.

We propose two lines of defense here: (1) security audits; and (2) the ability to veto slashing events. For security audits, we recognize that AVS codebases must be audited just like smart contracts are audited. While AVS codebases can be more complex than the average smart contract, an extenuating circumstance is that unlike smart contract audits, which are focused on protecting consumer users (members of the general populace), AVS audits are targeted at stakers and operators, who tend to be more sophisticated participants in the blockchain ecosystem. Given this sophistication and risk/reward profile, we expect a proper audit to be necessary for any AVS to get opt-in from stakers and operators. The second line of defense before an AVS ossifies is that there is a governance layer in Eigenlayer comprised of prominent members of the Ethereum and EigenLayer community, which has the ability to veto slashing decisions via a multisig. We think of the slashing veto process as similar to training wheels that will be eventually removed.

3.5 Governance

Given its role in having the power to veto slashing, it is important to carefully consider how to constitute this multisig veto committee. One way to select this veto committee is to use a token-based quorum. However, using a token-based quorum makes governance susceptible to a centralized entity buying up a majority of tokens and then being able to dictate its own rules or even worse, attack the system directly. In order to avoid such a scenario, we use a reputation-based committee comprised of reputed individuals in the Ethereum and EigenLayer community. This committee will hold the responsibility of enabling upgrades to the EigenLayer contracts, reviewing and vetoing slashing events, and admitting new AVSs into the slashing review process. Note that the veto committee has no power

to maliciously trigger slashing by itself, and that any upgrade to the EigenLayer contracts comes with a time lag.

This veto committee can be considered to be training wheels for new AVSs onboarding onto EigenLayer. The AVSs can use this veto committee to assure EigenLayer restakers that they will not be subject to malicious slashing or inaccurate slashing due to bugs, as there is always a fallback to the committee which can veto the slashing. Meanwhile, AVS developers can battle-test the codebase associated with the AVS. Once mature and getting enough trust from restakers, an AVS can stop using the veto committee as a fallback. The trust assumptions for using the veto committee are that the AVS must trust the veto commitee will not veto a correct slashing, while the stakers restaking in EigenLayer must trust the veto committee will veto any unjustified slashing by the AVS.

AVSs that want to build on top of EigenLayer and use the veto committee must be admitted by the veto committee. The onboarding process may require security audits and other diligence by the members of the committee, including checking the system requirements for operators to serve the AVS.

3.6 Designing Modules for Maximal Security Also Minimizes Centralization Risk

We note that the most security is obtained when all of the ETH that has been restaked with EigenLayer is used to secure a given AVS. However, there are two impediments to this: (1) whether the expected revenue of the AVS outweighs the operational costs for operators; and (2) whether operators have enough computational resources to participate in validating for the AVS. We propose two possible patterns by which a module can be designed in order to alleviate these concerns.

1. Hyperscale AVS: In a hyperscale AVS, the total computational workload is divided across all N participating operator nodes. This feature is called horizontal scaling, or scale-out in distributed computing. One prominent example of a hyperscale module is a hyperscale data availability protocol, where the total amount of data is chunked into N chunks each of size $2/N$ of the original data. Thus the total cost of storing the data is as though only 2 nodes store it. We note that in a hyperscale AVS, the throughput requirements of each node can be small while the system itself can achieve high throughput by aggregating performance across many nodes. **Furthermore, hyperscale AVSs have lesser incentives for centralized validation than existing blockchain systems today.** This is because, in existing blockchains (which are not horizontally scaled), the validation cost can be fully amortized by a central operator, as a state transition once known to be valid can be used by all operator keys. However, in a horizontally scaled module, it requires the centralized party to perform different validation per key, thus minimizing the amortization gain.
2. Lightweight: There are also examples of tasks which are redundantly performed by all the operators yet the cost of performing these tasks is very low, and the computing infrastructure required is also very low. For example, attesting on a certain message based on running a light client, verifying zero-knowledge proofs, running light nodes of other blockchains, and running oracle price feeds are lightweight actively validated services that can be run on EigenLayer.

We note that by designing a suite of hyperscale and lightweight AVSs which produce most of the yield available on EigenLayer, we can ensure that even home validators in Ethereum can obtain most of the economic benefit of EigenLayer, thus minimizing centralization pressures on Ethereum staking.

4 A World with EigenLayer

4.1 EigenLayer Enables New Applications

The set of new AVSs enabled by EigenLayer is quite broad and encompasses new blockchains, middleware, and modular blockchain layers such as data availability layers. We list here some of the possibilities, many of which are also exciting directions of ongoing and future research:

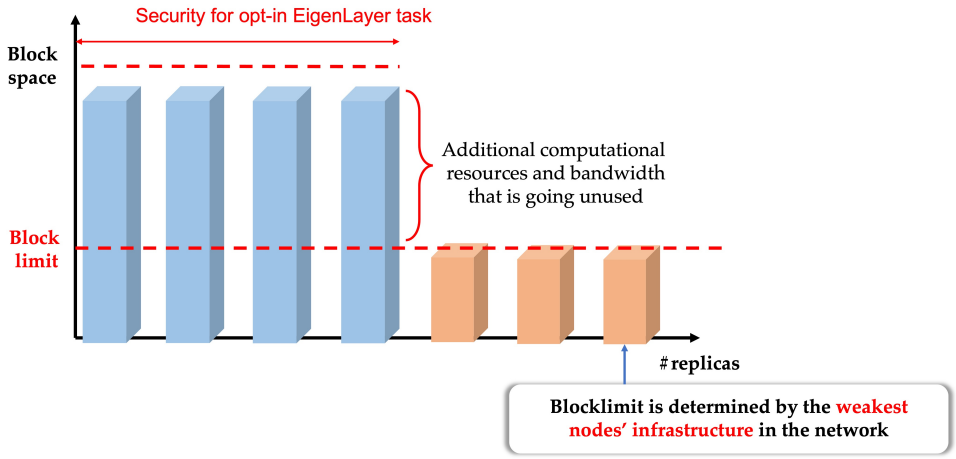


Figure 7: Blockspace is the most valuable commodity in blockchains [11]. In today's ecosystem, blocklimit on this blockspace is determined by weakest validator's infrastructure. This limit arises from Ethereum's focus on decentralization. However, EigenLayer can leverage the extra capabilities of validators with additional resources to provide guarantees of cryptoeconomic security to applications that desire them.

1. **Hyperscale Data Availability Layer** (Hyperscale AVS): We can build a hyperscale Data Availability (DA) layer providing a high DA rate and low cost by utilizing EigenLayer restaking and some of the cutting edge ideas in DA developed in the Ethereum community, including Danksharding.

2. **Decentralized Sequencers** (Lightweight / hyperscale AVS): Many rollups require decentralized sequencers for managing their own MEV and censorship resistance. These sequencers can be built on EigenLayer with a quorum of ETH stakers — there can be a single decentralized sequencer quorum that performs the service for many rollups. We note that the decentralized sequencer does not have to perform execution and can be only an ordering layer in which there is no state growth problems. So it is possible to make it either lightweight or even horizontally scaled (by choosing a random subset of consensus nodes to order different subsets of transactions).

3. **Light-Node Bridges** (Lightweight AVS): It is easy to build light-node bridges to Ethereum using EigenLayer. For example, the Rainbow Bridge between NEAR and Ethereum is based on an optimistic pattern but experiences high latency due to the high gas cost of verification [12]. It is possible for restakers to verify off-chain whether bridge inputs are correct, and if a strong cryptoeconomic quorum signs off on a bridge input, then the bridge input is considered accepted. If someone challenges, then the bridge input can be verified and the validators in EigenLayer can be slashed in the slow (non-optimistic) mode.

4. **Fast-Mode Bridges for Rollups** (Lightweight AVS): For ZK rollups, since the proof verification expense on Ethereum is still high, rollup sequencers write to Ethereum somewhat infrequently, affecting composability and delaying confirmation guarantees. It is possible for a quorum of operators on EigenLayer with a large amount of ETH restaked to participate in ZK proof verification off-chain, and to certify that proofs are correct on-chain. If the claims of the fast-mode bridge turn out to be wrong, a slower slashing path can be triggered. For optimistic rollups, EigenLayer can enable a much larger collateral pool to participate in certifying stateroots under risk of slashing.

5. **Oracles** (Lightweight AVS): There have been proposals for enshrining price feeds into Ethereum, or using the Uniswap token quorum for providing price feeds [13]. Such oracles can potentially be built via Eigenlayer if all we require is majority trust on ETH restaked with EigenLayer, and it is an opt-in layer.

6. **Opt-in Event-Driven Activation** (Lightweight AVS): Event-driven activations such as liquidations and collateral transfers are currently not available natively on Ethereum. While they can be built on a separate layer such as a keeper network, keeper nodes which do not manage blockspace cannot make strong guarantees of inclusion of event-driven actions. In EigenLayer, Ethereum validators who happen to be block proposers and are also opted-in to restaking on EigenLayer for an event-driven activation AVS can provide strong guarantees on inclusion of event-driven actions, at the risk of getting slashed.

7. **Opt-In MEV Management** : We note that a manifold variety of opt-in MEV management methods become feasible under EigenLayer, including Proposal-Builder Separation [14], MEV smoothing [15], and threshold encryption for transaction inclusion [16, 17]. As a simple example, MEV smoothing can be built on top of EigenLayer by a group of restakers that decides to share MEV equally among its members. Any restaker who deviates from the prescribed MEV smoothing behavior can be slashed. Since only block proposers need to do the specific action when they are triggered, it is naturally horizontally scaling.

8. **Settlement Chains with Ultra-Low Latency**: Ethereum has high latency to get economic finality (up to 12 minutes), and hence it may be useful to have fast settlement with high economic finality. EigenLayer allows the creation of restaked sidechains where ETH restakers can participate in new consensus protocols, some of which have very low latency and very high throughput. We note that settlement layers do not need to have state growth as settling ZK proofs is nearly stateless (some recent stateroots can be maintained as contract state). Furthermore, settlement layers can have a high degree of parallelization as many ZK proofs can be verified in parallel. This can be a lightweight AVS.

9. **Single-Slot Finality** (Lightweight AVS): It is possible to imagine single-slot finality, where nodes sign off on the finality of a block via an opt-in mechanism on EigenLayer [18]. The core idea would be that nodes who have restaked can now attest that they will not build on a chain that does not include the testified block, thus creating a potential finality pathway. Designing this so that it is truly opt-in and does not break the consensus protocol is an important direction of research.

4.2 EigenLayer Leverages Staker Heterogeneity, and Massively Expands Blockspace

Any blockchain decides its block limit based on the weakest node infrastructure that it is willing to admit to its network. Figure 7 shows an example of how resources are highly varied across nodes. We note that stakers who have more computational resources than the blockchain's requirements cannot utilize excess resources in the present homogeonous architecture that is prevalent across all blockchains. EigenLayer can take advantage of computational heterogeneity by creating **opt-in** validation tasks that can only be performed by higher-capacity nodes. This leads to lower cryptoeconomic safety, but this is fully measurable as the value at risk. Liveness, which may depend on decentralization, may not be easily quantifiable, but we note that systems built on top of EigenLayer can leverage the Ethereum main chain as a backup for liveness and censorship resistance much like rollups do.

Heterogeneity across stakers goes far beyond computational heterogeneity. There are also significant differences between stakers based on their risk preferences. For example, some restakers may be willing to accept being slashed via participation in an oracle module, when their oracle inputs differ from those of a well-known exchange; for other restakers, this risk may be considered too high. There are also strong differences in reward preferences. Some restakers will want to take a positional bet on say, decentralized social networks, and allocate any excess computational resources they have to helping validate and bootstrap such social networks. These restakers might receive fees for their services in the form of illiquid tokens. On the other hand, other restakers may only consider immediately transferable and liquid rewards as having value. We note that restakers may also have a variety of distinguishing traits based on ownership of or interaction with Verifiable Credentials, Soul-Bound Tokens [19], and other technologies and assets. These identifiable traits may serve as criteria by which restakers are recruited to perform certain module validation tasks. Thus EigenLayer allows the expression of heterogeneity across restakers in computational capacity, risk preferences, reward preferences, and

identity, while also allowing modules to recruit restakers based on a combination of these preferences and traits.

We emphasize that the notion of retuning the performance/security parameters of DApps does not indicate a difference in philosophy between EigenLayer and Ethereum from the perspective of decentralization. Decentralization remains an important part of ensuring liveness in Ethereum, on which both EigenLayer and its modules directly depend. Furthermore, as previously discussed, by building hyperscale and lightweight validation tasks which produce much of the fee revenue on EigenLayer, even homestakers can continue to enjoy a large fraction of the yield available on EigenLayer, and thus the network can avoid centralization pressure.

4.3 EigenLayer Breaks the Trade-Off between Democracy and Agility

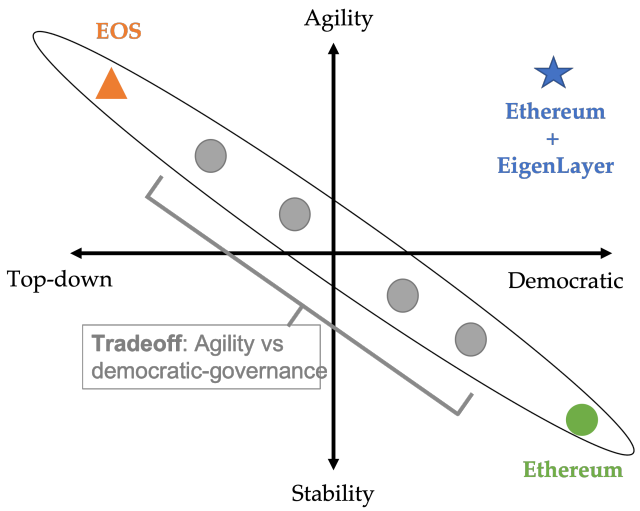


Figure 8: Ethereum, in its present form, features a democratic governance but slow pace of innovation. With the addition of EigenLayer, Ethereum can enjoy the best of both worlds: democratic governance with agility in innovation.

The Ethereum protocol is upgraded through a democratic mode of off-chain governance that is robust, and as a consequence, necessarily slow-moving. As shown in Fig. 8, competing protocols such as Binance Smart Chain (BSC) have sought to achieve faster decision making and updates by adopting a more top-down, corporate type of governance. While such decision making is efficient, it is less democratic. All protocols today end up making some trade-off between democratic governance and the rate of innovation.

The emergence of EigenLayer challenges the existence of any trade-off between democratic governance and agile innovation. In fact, EigenLayer allows agile innovation to be built on top of the Ethereum trust network, while leaving the core of Ethereum alone to continue upgrading in a cautious and stable manner. This achieves the best of both worlds. Furthermore, the rate of permissionless innovation possible on EigenLayer far exceeds the rate of innovation possible in any top-down governance model, where decisions are limited to only a small group of entities or individuals.

In this configuration, long-term stability is provided by the Ethereum base layer which is upgraded carefully, as it has always been, while agile innovations that are needed in the short-term are distributed via EigenLayer by the free market. **In fact, EigenLayer can act as a staging network for Ethereum, where competing ideas are implemented in a permissionless manner before being integrated and absorbed back into Ethereum.** We are already working on implementing a Danksharding-based data availability layer in this staging network, so that some of the lessons learned

can inform future Ethereum designs.

Thus Ethereum plus EigenLayer allows for **both cautious, conservative governance for Ethereum, and agile, permissionless innovation** simultaneously.

4.4 EigenLayer Can Incentivize Ethereum Staker Decentralization

There is a natural tendency for stake and validator nodes to centralize in any distributed system. However, many AVSs with distributed systems rely on having their stake and validator nodes be as decentralized as possible. One example is a threshold encryption system for on-chain privacy. If majority of the keys are held by one centralized entity, then privacy can be broken by this centralized entity before the stipulated period in a non-attributable manner. Therefore, there is a need for the set of threshold key holders to be as decentralized as possible so that no one can know the actual content before the stipulated period.

With EigenLayer, an AVS can explicitly specify that the stake/nodes participating in its validation tasks must be part of a decentralized quorum. More explicitly, the AVS can specify in its contracts while integrating with EigenLayer that only Ethereum home validators can participate in its tasks, thus contributing to keeping the AVS decentralized. This ability to permissionlessly specify who can participate in the validation tasks of an AVS is akin to programming decentralization.

At an abstract sense, EigenLayer enables a marketplace for AVSs to buy decentralization. As more and more AVSs specify that only home validators can participate in their tasks on EigenLayer, this makes it more profitable to run home validator nodes on Ethereum which incentivizes decentralization.

4.5 Multi-Token Quorums

EigenLayer offers flexibility for AVSs to define their own quorum alongside a quorum comprised of restaked ETH, and require the final response to its validation tasks to be a function of responses from a majority of each quorum. For example, an AVS can specify two quorums, an Ethereum restakers quorum and an $AVS quorum (where $AVS is the AVS's token). In order for any operator on EigenLayer to perform services for this AVS, they either have to restake their ETH or they have to stake $AVS token. The AVS can treat both quorums as two independent quorums and use an AND clause to combine the majority response from both the quorums.

This flexibility to define multiple quorums offers an opportunity to the AVS to bootstrap its own token as a utility token and accrue value to its protocol, while using the restaked ETH quorum to hedge against a death spiral of its own token.

4.6 Business Models on EigenLayer

The following are business models that can be built by AVSs on top of EigenLayer:

1. Pure wallet. Under this model, the company deploys an AVS on top of EigenLayer as a commercial service. Users utilizing the AVS need to pay fees in a neutral denomination: (1) a fraction of those fees goes to the company wallet in return for their service, and (2) the rest of the fees goes to ETH restakers in EigenLayer and the EigenLayer protocol. This enables pure company-based business models and allows SaaS economics to be built by AVSs inside the crypto space.

2. Tokenize the fee. Under this model, the AVS is being operated as a protocol (instead of being as a commercial service). Users utilizing the AVS need to pay fees in a neutral denomination: (1) a fraction of those fees goes to a quorum of token holders of $AVS (the native token of AVS) as specified by the protocol, and (2) the rest of the fees goes to ETH restakers in EigenLayer and the EigenLayer protocol.

3. Pay in the native token of the AVS. Under this model, the AVS is being operated as a protocol and users of the protocol need to pay fees in the denomination of a specific token $AVS issued by the AVS. The value of this token $AVS is contingent on the expectation of continuous profitable operation of the AVS in the future. The fee consists of two parts: (1) a fraction of the fees goes to a quorum of token holders as specified by the protocol, and (2) the rest of the fees goes to ETH restakers in EigenLayer and the EigenLayer protocol.

4. Dual staking utility. Under this model, the AVS also issues its own token $AVS and EigenLayer features two quorums: the first quorum is composed of ETH restakers, and the second quorum is composed of $AVS stakers. In the dual quorum model, safety is the better of the two quorums, and liveness is the worst of the two quorums. Now, anyone with either ETH or $AVS can participate in providing security to the AVS via EigenLayer by restaking ETH or staking ($AVS) in their respective quorums. This leads to dual staking in EigenLayer, which can drive utility to the AVS's token $AVS, while using ETH restaking to hedge against a potential death spiral of $AVS which would compromise the security of the AVS.

5 Summary

1. EigenLayer creates a free market for decentralized trust, delivered from Ethereum stakers to modules that desire stake and validation services.

2. By restaking via EigenLayer, Ethereum stakers can opt into providing security and validation services to modules of their choice, either by operating nodes directly, or via delegation to other EigenLayer operators.

3. A variety of lightweight and hyperscale modules can be built on EigenLayer, which can be designed for widespread participation from solo stakers.

4. Modules may also take advantage of significant heterogeneity between stakers, which may differ on axes of computational capacity, risk/reward preferences, and identity.

5. EigenLayer seeks to enable more agile, decentralized, and permissionless innovation on blockchains.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[2] Vitalik Buterin et al. Ethereum white paper.

[3] Ethereum: A secure decentralized generalized transaction ledger. `https://ethereum.github.io/yellowpaper/paper.pdf`.

[4] Rollup-centric ethereum roadmap. `https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698`.

[5] An incomplete guide to rollups. `https://vitalik.ca/general/2021/01/05/rollup.html`.

[6] Lido.fi. `https://lido.fi/`.

[7] Rocketpool. `https://rocketpool.net/`.

[8] Superfluid staking. `https://docs.osmosis.zone/osmosis-core/modules/superfluid/`.

[9] The cryptoeconomics of slashing. `https://a16zcrypto.com/the-cryptoeconomics-of-slashing/`.

[10] Merged mining. `https://developers.rsk.co/rsk/architecture/mining/`.

[11] Blockspace: An introduction with chris dixon. https://www.generalist.com/briefing/blockspace.

[12] Rainbow bridge. https://near.org/bridge/.

[13] Uni should become an oracle token. https://gov.uniswap.org/t/uni-should-become-an-oracle-token/11988.

[14] State of research: increasing censorship resistance of transactions under proposer/builder separation (pbs). https://notes.ethereum.org/@vbuterin/pbs_censorship_resistance.

[15] Committee-driven mev smoothing. https://ethresear.ch/t/committee-driven-mev-smoothing/10408.

[16] Shutter - in-depth explanation of how we prevent front running. https://blog.shutter.network/shutter-in-depth-explanation-of-how-we-prevent-frontrunning/.

[17] Removing trusted relays in mev-boost using threshold encryption. https://ethresear.ch/t/removing-trusted-relays-in-mev-boost-using-threshold-encryption/13449.

[18] Paths toward single-slot finality. https://notes.ethereum.org/@vbuterin/single_slot_finality.

[19] Soulbound. https://vitalik.ca/general/2022/01/26/soulbound.html.

A Appendix: Restaking Security at the System Level

We have seen that restaking allows for a transfer of cryptoeconomic security from a base layer blockchain such as Ethereum to other modules such as middleware and protocols. This enables the pooled and shared security that lies at the core of EigenLayer. In Section , we alluded to the benefits of this pooled security with respect to system-level security characteristics. In this section, we will review these benefits in greater detail.

Fig. 3a provides a picture of a system of today, in which a single DApp relies on a network of blockchain and middleware dependencies to provide the full set of required trusted services, such as execution, ordering, oracle queries, etc. Generally speaking, any one of a DApp's middleware dependencies may be the basis of an attack. Thus, the cost of corruption (see Section 3.1) of a DApp generally must be taken to be no more than the **minimum** of the costs to corrupt each of its dependencies. That is, for an application i serviced by a set of modules, M_i, the cost of corruption is given by

$$\mathrm{CoC}_i = \min_{j \in M_i} \{CoC_j\}. \tag{1}$$

In such a world, the powerful security guarantees provided by Ethereum may be meaningless when an application relies on a middleware module such as an oracle with a much smaller amount of securing stake.

We can compare this cost of corruption to that of a restaked world in which all modules are secured by a single network of stakers, as shown in Fig. 3b. To first order, we can assume that when modules merge together, their separate total market caps (the total values of their staking tokens) sum together. Assuming this behavior, we see that for a DApp i built on this network, the cost of corruption is now given by

$$\mathrm{CoC}_i = \sum_{j \in M_i} CoC_j. \tag{2}$$

B Appendix: Cryptoeconomic Risk Analysis

Let us formally describe the desired cryptoeconomic security guarantees for EigenLayer. Let T denote the set of tasks secured by EigenLayer, and let S denote the set of stakers securing these tasks. Let s_i denote the stake held by a staker $i \in S$. For a task $i \in T$, let S_j denote the set of stakers who have restaked for that task, and let α_j denote the fraction of this stake which is required to corrupt the task.

We say that set of validators $U \subseteq T$ is sufficient to corrupt a task $j \in T$ if their combined stake surpasses the fraction α_j. We let $\mathcal{S}_j^c$ denote the collection of staker sets which are sufficient to corrupt j:

$$\mathcal{S}_j^c = \left\{ U \subseteq S : \sum_{i \in S_j \cap U} s_i \geq \alpha_j \sum_{i \in S_j} s_i \right\}. \tag{3}$$

Likewise, for a set of tasks $V \subseteq T$ we let $\mathcal{S}^c(V)$ denote the collection of staker sets which are sufficient to corrupt all tasks $j \in V$: $\mathcal{S}^c(V) = \bigcap_{j \in T} \mathcal{S}_j^c$. We can now define the cost of corruption for a set of tasks $V \subseteq T$ to be minimum amount of stake held by a set of stakers which is sufficient to corrupt V:

$$c(V) = \min_{S \in \mathcal{S}^c(V)} \sum_{i \in S} s_i. \tag{4}$$

Similarly, for each task $i \in T$, we suppose that there exists some well defined profit from corruption p_j, which is the maximum amount of value which can be extracted from the task by a set of stakers which is sufficient to corrupt j. We assume that p_j does not depend on $S \in \mathcal{S}_j^c$ or on the actions take by stakers to corrupt other tasks. We denote the total profit from corruption for a set of tasks $V \subseteq T$ as

$$p(V) = \sum_{j \in V} p_j. \tag{5}$$

A task j is secure if and only if for each set of tasks containing j, the cost of corrupting that set is greater than the profit from corrupting that set. That is,

$$j \text{ is secure} \iff \forall\ V \ni j,\ \ c(V) > p(V). \tag{6}$$

Thus, for all tasks to be secure, we need to have that for all $V \subseteq T$, $c(V) > p(V)$.

From a different perspective, for each set of stakers, $U \subseteq S$, we can also define the set of tasks $T^c(U)$ which this set of stakers is able to corrupt:

$$T^c(U) = \left\{ j \in T : \sum_{i \in S_j \cap U} s_i \geq \alpha_j \sum_{i \in S_j} s_i \right\}. \tag{7}$$

An equivalent condition for the security of all tasks is that no set of stakers can corrupt a set of tasks whose profit from corruption is greater than their total stake, that is,

$$\text{all tasks are secure} \iff \forall\ U \subseteq S, \sum_{i \in U} s_i > \sum_{j \in T^c(U)} p_j. \tag{8}$$

In the next two subsections, we will consider possible approaches for both monitoring whether this condition is met and providing inputs to stakers and module developers on actions needed to restore cryptoeconomic security when this condition is not met, or only marginally met. EigenLayer will provide a risk management dashboard for supplying this advisory information to stakers and module developers.

B.1 Sufficient Conditions for Cryptoeconomic Security

The preferred approach for ensuring that all tasks are cryptoeconomically secure is to ensure that all stakers satisfy a simple condition, given by

$$s_i \geq \sum_{j \in T_i} \gamma_{ij} \frac{p_j}{\alpha_j} \tag{9}$$

where $\gamma_{ij} = s_i / \sum_{k \in S_j} s_k$ gives the fraction of task j's securing stake which is held by staker i. This condition is sufficient to ensure security for all tasks since for any $U \subseteq S$, we have,

$$\sum_{i \in U} s_i \geq \sum_{i \in U} \sum_{j \in T_i} \gamma_{ij} \frac{p_j}{\alpha_j} = \sum_{j \in T} \sum_{i \in U \cap S_j} \gamma_{ij} \frac{p_j}{\alpha_j} \geq \sum_{j \in T^c(U)} \frac{\sum_{i \in U \cap S_j} s_i}{\sum_{i \in S_j} s_i} \frac{p_j}{\alpha_j} \geq \sum_{j \in T^c(U)} \frac{\alpha_j}{\alpha_j} p_j = \sum_{j \in T^c(U)} p_j, \tag{10}$$

satisfying the right-hand side of (8).

We define the overcollateralization oc_i of a staker i to be given by

$$oc_i = s_i - \sum_{j \in T_i} \gamma_{ij} \frac{p_j}{\alpha_j} \tag{11}$$

We say that a staker is undercollateralized $oc_i > 0$ and overcollateralized if $oc_i < 0$. When there exists an undercollateralized staker i, one of several actions should happen: 1, the undercollateralized staker i can increase their amount of stake; 2, the undercollateralized staker can deregister or be deregistered from some set of modules; or 3, other stakers can adjust their own registrations.

B.2 Continuous Relaxation

We now present a slightly modified formation which allows for more flexibility in finding secure allocations. We allow each staker to allocate a portion of their stake to each module. This is useful for when allocating their entire stake will result in undercollateralization.

Specifically, the scheme is as follows. We let $\sigma_{ij} \in [0, 1]$ denote the fraction of a validator i's stake which as been assigned to task j. This has the following meaning: Validator i's *influence* on task j is determined using the stake amount $\sigma_{ij} s_i$. However, importantly, if a validator acts maliciously in any task for which their allocation is non-zero, *all* of their stake will be slashed.

Now, the condition for security can be restated. The collection of tasks which, given a set of stakers, U, is able to corrupt is given by

$$T^c(U) = \left\{ j \in T : \sum_{i \in U} \sigma_{ij} s_i \geq \alpha_j \sum_{i} \sigma_{ij} s_i \right\}. \tag{12}$$

With this modification, condition (8) continues to hold. Corresponding to this relaxation, we have a sufficient condition for security:

$$s_i \geq \sum_j \frac{\sigma_{ij} s_i}{\sum_k \sigma_{kj} s_k} \frac{p_j}{\alpha_j}. \tag{13}$$

When this condition is met for all i, we have for any $U \subseteq S$ that

$$\sum_{i \in U} s_i \geq \sum_{i \in U} \sum_j \frac{\sigma_{ij} s_i}{\sum_k \sigma_{kj} s_k} \frac{p_j}{\alpha_j} = \sum_{j \in T} \frac{\sum_{i \in U} \sigma_{ij} s_i}{\sum_k \sigma_{kj} s_k} \frac{p_j}{\alpha_j} \geq \sum_{j \in T^c(U)} \frac{\sum_{i \in U} \sigma_{ij} s_i}{\sum_k \sigma_{kj} s_k} \frac{p_j}{\alpha_j} \geq \sum_{j \in T^c(U)} \frac{\alpha_j}{\alpha_j} p_j = \sum_{j \in T^c(U)} p_j. \tag{14}$$

The benefit of the relaxation is that it allows for an efficient algorithm for finding a secure allocation given fixed stake amounts (s_i's) and profits from corruption (p_j's). This is a achieved by posing the problem of finding a good allocation $\sigma \in [0,1]^{S \times T}$ as an optimization problem. We will show that it is possible to formulate this problem as a maximization of quasi-concave objective subject to linear constraints.

We first introduce a function $f : [0,1]^{S \times T} \to \mathbb{R}^{S \times T}$ defined by

$$f_{ij}(\sigma) = \frac{\sigma_{ij} s_i}{\sum_k \sigma_{kj} s_k} \frac{p_j}{\alpha_j}.$$

Each f_{ij} is quasiconvex since its sublevel sets are convex. We wish to find an allocation σ such that each validator is overcollateralized, i.e. for all $i \in S$, $s_i - \sum_j f_{ij}(\sigma) > 0$. For this to be true, it is necessary and sufficient that there exists a set of variables $b_{ij} \in \mathbb{R}$ such that the following conditions hold jointly:

$$\forall i \in S, \quad s_i - \sum_j f_{ij}(\sigma) > 0 \tag{15}$$

$$\forall i, j \in S \times T, \quad b_{ij} - f_{ij}(\sigma) \geq 0 \tag{16}$$

We now reformulate the problem of satisfying these constraints as a problem of finding the allocation which maximizes the smallest slack in (16) while satisfying the condition in (15).

$$\begin{aligned} \max_{\sigma, b} \ & \min_{ij} b_{ij} - f_{ij}(\sigma) \\ s.t. \ & s_i - \sum_j f_{ij}(\sigma) > 0 \ \ \forall i \in S \end{aligned} \tag{17}$$

This is an equivalent problem in the sense that when its solution is positive, both of the preceding conditions will be satisfied. But the objective (17) is the minimum over quasi-concave functions, and is thus quasi-concave itself. Thus the optimization has an efficient approximation algorithm.

The constraints in (17) can be augmented with constraints from the validators indicating whether or not they are willing to participate in validating for a given modules. Since these preferences manifest as linear constraints on σ, the problem remains tractable. This means that we have a general purpose tool for finding secure allocations subject to arbitrary validation preferences.