In our previous post we introduced a 17.5 day withdrawal and deallocation delay for stakers and Operators, respectively. This post explains how that number was determined through the lens of slashable stake guarantees. Additionally, we show that, with reasonable assumptions, this window can be reduced to 14 days.

AVS Sync

Assume there is an Operator Set with a single Operator that allocates slashable stake from one Strategy. The AVS periodically syncs with EigenLayer to update its view of the slashable stake allocated to the Operator Set. Since each sync incurs a cost, the AVS chooses to sync once every AVS Sync Window

days. During this period, the AVS assigns tasks to the Operator Set based on the information from the latest sync. The time from when a task is assigned to an Operator Set to when it can be completed is called the Task Completion Window

. We assume that for any task assigned within an AVS Sync Window, the Task Completion Window ends before the next sync.

Given the monetary cost of syncing, we assume the AVS syncs once every 7 days, and syncing happens instantaneously.

Slashability Window

The Slashability Window

is the period between when an Operator completes a task and when it is slashed for misbehavior. If misbehavior occurs, we assume an observer instantly submits a fraud proof to the AVS contracts which creates a slashing request. This process takes a certain number of days to finalize on-chain, which we refer to as the Fraud Proof Window

. Once the fraud proof is finalized, AVS governance deliberates locally for some time, called the Governance Window

, to decide whether to forward or cancel the slashing request. We assume that whatever mechanism is used for governance is initiated the instant the fraud proof is finalized. Once a decision is made, governance submits a transaction to forward or cancel the slashing request. The amount of time it takes for this transaction to finalize on Ethereum is called the Slashing Request Window

. Thus, the length of Slashability Window is:

$$\text{Slashability Window}= \text{Fraud Proof Window}+\text{Governance Window}+\text{Slashing Request Window}$$

Stake Guarantee Window

Properly parameterized, EigenLayer can guarantee a minimum amount of stake will be available to slash for any task that is completable within an AVS Sync Window. This period, set globally, is called the Stake Guarantee Window

. To determine the length of this period we make assumptions about the lengths of the AVS Sync Window and Slashability Window:

$$\text{Stake Guarantee Window}= \text{AVS Sync Window}+\text{Slashability Window}$$

Therefore, for this guarantee to hold, an AVS must ensure that the sum of its Sync Window and Slashability Window is less than or equal to the globally defined Stake Guarantee Window.

Desired property

To guarantee an amount of slashable stake for a particular task, we need to make an additional assumption about withdrawals and deallocations. A magnitude deallocation and a withdrawal of delegated stake result in an arbitrary reduction of the Operator Set's slashable stake. Therefore, the Stake Guarantee Window needs to be shorter than or equal to the time it takes for a staker's withdrawal to be non-slashable and the amount of time it takes for a deallocation to take effect. Otherwise, an Operator or staker could evade slashing by deallocating or withdrawing, respectively, before the Stake Guarantee Window ends:

$$\text{Stake Guarantee Window} \le \min(\text{withdrawal delay}, \text{deallocation delay})$$

where withdrawal delay

is the time between a withdrawal being queued by a staker and the corresponding stake no longer being subject to slashing and similarly for the deallocation delay

. For simplicity, we will assume withdrawal and deallocation delays are the same and refer to both as the withdrawal delay. In our planned production implementation, these parameters are identical. Also, to note the obvious tradeoff, the longer our Stake Guarantee Window, the longer the withdrawal and deallocation delays need to be

.

As mentioned here, the protocol allows only one allocation or deallocation at a time, and deposits to Operators are effective immediately. Therefore, the AVS needs to know the pending deallocation, if there is one, and withdrawals from an Operator that have been made between withdrawal delay days ago and today.

Thus, the minimum amount of slashable stake that EigenLayer can guarantee will be available for the time between when an AVS syncs and the end of the Stake Guarantee Window:

\text{stake}*{\text{guaranteed}}=(\text{stake}*{\text{now}}-\text{withdrawals}*{\text{pending}})
\times\frac{\text{magnitude}*{\text{now}}-\text{deallocation}*{\text{pending}}}{\text{total magnitude}*\text{now}}

where \text{withdrawals}_{\text{pending}}

and \text{deallocation}_{\text{pending}}

are the withdrawals and deallocation that can be completed within the Stake Guarantee Window. The total amount of slashable stake that can be guaranteed by EigenLayer is the sum of guaranteed stake across Operators for each strategy. However, this amount is guaranteed when the AVS syncs to EigenLayer. Between syncs, the amount of guaranteed stake is \text{stake}_{\text{guaranteed}}

at the most recent sync minus the amount that it has slashed since.

Duration

Since we assume the AVS syncs once every 7 days, we need to determine an appropriate Slashability Window. We assume that no individual can be censored on Ethereum for more than 3.5 days, possibly discontiguous, within a 7 day period. Thus, we can assume 3.5 days for each Fraud Proof Window, Governance Window, and Slashing Request Window. This is assuming that once the Governance Window ends, governance instantly forwards the slashing request. This results in a Slashability Window of 10.5 days and a Stake Guarantee Window of 17.5 days. We assume the AVS does not consider any fraud proofs that are confirmed after the Fraud Proof Window.

This is how we came up with 17.5 days in our previous forum post on allocating slashable stake. Figure 1 demonstrates how this process would work for two tasks assigned during a particular AVS Sync Window. The first task is assigned immediately after the AVS syncs and a second task is completable immediately before the next AVS sync.

[

figure_1

2128×1286 216 KB

](https://canada1.discourse-cdn.com/flex028/uploads/eigenlayer/original/2X/f/f6858e53156626b75e647cc15cf36fee2bdd05aa.png)

Alternatively, we can loosen our censorship assumption to be that no group

can be censored for more than 3.5 days, possibly discontiguous, within a 7 day period. Also, we can assume that governance is a single centralized entity and can instantly decide whether or not to cancel or forward the slashing request. Thus, the Governance Window is equal to 0. Also, our new assumption guarantees that the Fraud Proof Window and Slashing Request Window take 3.5 days, collectively. The Stake Guarantee Window is then 10.5 days as shown in Figure 2.

[

figure_2

2098×1350 212 KB

](https://canada1.discourse-cdn.com/flex028/uploads/eigenlayer/original/2X/1/1876badffa4c8dfae6f36bf1ef9e0096fb63bf19.png)

Suggested duration

However, we propose a more reasonable set of assumptions that yields an additional beneficial property. We assumed that governance deliberates and forwards or cancels the slashing request. If governance is not the entity that submitted the fraud proof, then it requires two parties to submit a transaction: the fraud proof submitter and governance. However, let us assume that governance does not need to sign a transaction to forward the slashing request in the event of legitimate misbehavior. Once the governance period is over and it is decided that they will not cancel the slashing request, the fraud proof submitter can then forward the slashing request to EigenLayer. This only requires the weak assumption that no individual can be censored for more than 3.5 days, possibly discontiguous, within a 7 day period. Then, the Slashability Window would include 3.5 days for the fraud proof submitter to sign the fraud proof and forward the slashing request,

collectively, plus the 3.5 days it takes for governance to decide to cancel the slashing request or not. Thus, the Slashability Window is 7 days, resulting in a Stake Guarantee Window of 14 days.

With a Slashability Window of 7 days, if there is a bug in the fraud proof allowing an attacker to generate a false positive, the attacker would try and get their fraud proof confirmed right before the end of the Fraud Proof Window in which case governance would need to instantly submit a transaction cancelling the slashing request which, given our censorship assumption, would take at most 3.5 days.

We suggest setting the Stake Guarantee Window, and thus the withdrawal and deallocation delays, equal to 14 days.

Other thoughts & future directions

There are several other issues and features to consider. For example, rate limiting withdrawals and allowing AVSs to set custom withdrawal delays. Also, at any point in time, there can be a difference in the guaranteed minimum amount of slashable stake and the actual amount of slashable stake, with the latter always being greater than or equal to the former by definition. So, if AVSs make decisions based on the guaranteed minimum amount then slashable stake is not being used efficiently. We also assumed that AVSs sync every 7 days, but they can sync more or less often than that.

Additionally, perhaps more intuitively, instead of referring to the window between syncs as the AVS Sync Window, we can just call that whole period the Task Completion Window. For example, reframing would result in the Stake Guarantee Window being the sum of the Task Completion Window and the Slashability Window.

Inquiry

We have outlined how we think about determining the Stake Guarantee Window and how it relates to other time parameters. We encourage feedback and alternative perspectives. For example: how long does the Slashability Window have to be for your specific use case? Or how do you plan on involving governance to cancel slashing? Any thoughts and discussion are encouraged!

Disclaimer

The Eigen Labs Research Team uses the Forum as a space to share research on the protocol, to preview elements and ideas that may or may not become part of upcoming releases, and to explore ways of using, analyzing, and thinking about the EigenLayer protocol and ecosystem. Unlike our blog posts and social media announcements, which focus on formal updates and decisions, the Forum will be more interactive and experimental, allowing us to exchange ideas and gather feedback from a wider group.