This Whitepaper is Alpha Version 0.1

All of the information contained in this early Alpha release of the Anzen Protocol Whitepaper is subject to change prior to the launch of the protocol. The Whitepaper is released in this early state for research and educational purposes, and covers only a limited set of possible implementations and features of the Anzen Protocol.

Anzen Protocol: Optimizing Operator Payments on EigenLayer

Hydrogen Labs

January 16, 2024

Abstract

In EigenLayer, Actively Validated Services (AVS) are applications that require economic security to perform certain functions. Economic security is reached when the Cost of Corruption (CoC) is greater than the potential Profit from Corruption (PfC) [1]. AVSs achieve economic security by sufficiently incentivizing operators to restake value such that the CoC is greater than the PfC. However, because both the CoC and PfC are constantly changing, dynamic adjustment of incentives is required to maintain a sufficient but not excessive security budget. In order to achieve this, we first propose a framework for measuring the economic security of a given AVS. Using this framework, Anzen Protocol calculates the economic security of AVSs and triggers adjustment of incentive rates towards the optimal level. Active management of incentives offers enhanced security and long-term savings on incentive emissions for AVSs. Anzen Protocol is itself designed as an AVS, benefiting from the decentralization and security of EigenLayer operators.

1 Introduction

Because blockchains are pseudonymous, censorship-resistant and irreversible, AVSs can only rely on correctly structured economic incentives to ensure security, as opposed to chargebacks, escrow, social consensus, or legal threats [3]. Decentralized consensus based on economic security, such as Proof of Stake, powers many protocols, including Ethereum [2]. The core concept of Proof of Stake is that participants in the networks stake assets, which can be subject to slashing in the event a participant attempts a malicious action [4]. A system is considered economically secure when the locked assets available for slashing exceed the potential profit from malicious behavior.

EigenLayer introduces the concept of Shared Security, where AVSs are secured by ETH, rather than solely by the issuance of their own native assets. Ethereum validators run AVS modules simultaneously as they validate Ethereum. AVS modules impose a set of conditions under which non-compliant validators running the AVS may be slashed. In return, validators receive additional revenue in the form of incentives. This mechanism is called restaking. Validators who restake are termed 'operators' or 'restakers.'

EigenLayer creates an open marketplace for economic security. AVSs rent economic security from operators in exchange for incentives. Operators may choose whether to opt in or out of restaking for each AVS. When incentives are adjusted, operators naturally join as profitability increases or leave as it decreases, relative to the perceived risks of a given AVS. EigenLayer reduces the barrier to entry for new protocols by bootstrapping an existing validator set and pool of staked assets.

Additionally, EigenLayer reduces the technical complexity of creating economic security mechanisms. Previously, protocols were required to design and implement their own economic security mechanisms in their protocol's codebase, often building from scratch. Nethermind's open-source AVS node specification offers a framework for any protocol to use as a starting point to create an economically secure protocol [5]. With EigenLayer, bootstrapping security becomes readily accessible.

2 Motivation

One of the core motivations of dynamically adjusting incentives for an AVS is to protect against a situation where the CoC drops below the PfC. If the CoC is less than the PfC, it is profitable to attack the AVS and claim the difference. This vulnerability can occur when the CoC decreases, the PfC increases, or both happen simultaneously. Another core motivation is to prevent overpaying for economic security when the CoC is excessively higher than the PfC. AVSs are naturally averse to overpaying incentives in their native token or treasury assets, as security is an ongoing expense.

3 Defining the Economic Security Challenge

In the context of an EigenLayer AVS, the CoC is directly proportional to the amount of restaked value allocated to the AVS. A variety of unpredictable factors may result in operators opting in or out of restaking at any given moment, thereby changing the CoC. Some of these factors include changes in the value of incentives offered to operators (i.e. token price fluctuations), and changes in the perception of risks associated with a particular AVS. Fluctuations in the price of ETH, ETH LSTs, and other assets used for restaking also continuously change the CoC.

The PfC is defined as the maximum profit that can be gained from an AVS if economic security is corrupted. The implementation of a PfC calculation must be customized for each AVS. Typically, it is based on the value of assets owned by the protocol that could be extracted if economic security is compromised. The PfC is also constantly changing, largely due to normal fluctuations in the usage of the AVS. For example, rollups and app-chains have variable amounts of value at risk based on the price and the amount of assets bridged in/out. Another example is price feed oracles, which have a variable PfC based on the cumulative value that could be extracted by an attacker. The cumulative value at risk includes all of the third party integrations relying on the oracle, which can vary based on inflows/outflows in each protocol or new third party integrations.

3.1 PfC Risk Analysis & CoC Buffer

For simple protocols with few third party integrations, the PfC is largely equivalent to Total Value Locked (TVL). For more complex and deeply integrated protocols, the PfC is more intricate, requiring extensive risk analysis. First, we need to determine how much value an attacker could directly siphon by transferring assets out of the protocol. Then, we must consider the potential value extracted via manipulation of other protocols that rely on the AVS. Furthermore, additional value could potentially be realized by arbitraging or shorting various affected assets and protocols.

Given the intricate interactions among protocols, ecosystems, and stakeholders, crafting an all-encompassing PfC formula for a complex protocol requires careful analysis. Similar to the existing process for managing complex risk parameters in DeFi governance, AVSs may take advantage of risk and market analyses, simulations, and other methodologies. Additionally, the deliberate buffer between the CoC and PfC ensures that AVSs maintain a high degree of safety in all conditions. This buffer, analogous to the pattern of over-collateralization widely utilized in decentralized finance, helps ensure protocol security during periods of high volatility.

4 AVS Safety Factor

We propose a unified metric for measuring AVS economic security, termed the Safety Factor (SF). To calculate the SF for a given AVS, we first need to calculate two input parameters:

1. CoC: The value of restaked capital required to gain control of a given AVS
2. PfC: The maximum profit that can be gained from an AVS if economic security is corrupted

Combining these 2, we create the SF:

$$SF = \frac{CoC - PfC}{CoC} \tag{1}$$

The SF represents the net profit a malicious actor can expect as a result of attacking the protocol. The SF represents the buffer between PfC and CoC that an AVS maintains for security in the event of a sudden upward change in the PfC or sudden downward change in the CoC.

$$1 > SF > 0 : \text{Profitable Corruptibility} \tag{2}$$

A SF greater than 0 means that an AVS is economically secure. A SF below 0 means that the AVS is at risk. As the SF value approaches 1, it signals that the AVS is overpaying for security.

4.1 Example SF: A Bitcoin Bridge AVS

For example, suppose there is a Bitcoin bridge to Ethereum secured by EigenLayer restaking. In this example, we assume that the bridge is secured by a 66% honest-majority consensus to manage bridged funds [6].

The PfC is the value of the Bitcoin in the bridge.

The CoC for this example is expressed as:

$$\frac{ETH_{Attack}}{ETH_{Restaked} + ETH_{Attack}} = \frac{2}{3} \geq ETH_{Attack} = 2 \times ETH_{Restaked} \tag{3}$$

In this situation, if the TVL of the Bitcoin in the bridge is greater than twice the total restaked value, it is then profitable to attack the protocol.

$$SF = \frac{2 \times ETH_{Restaked} - BTC_{Bridged}}{2 \times ETH_{Restaked}} \tag{4}$$

Maintaining the SF above 0 is crucial to the AVS. If it is trending towards 0, then incentives must be increased to encourage more restakers to sufficiently secure the AVS.

5 Anzen Protocol Overview

To address the market dynamics of running an AVS on EigenLayer, we introduce Anzen Protocol. Anzen is designed to optimize operator payments in a way that avoids periods of insecurity as well as periods of overpaying. The protocol contains a suite of modules that calculate the SF for a particular AVS. Once the SF is calculated for a given AVS, it is consumed by an Incentive Reserve which adjusts the incentive emission rate. Anzen Protocol is run as an AVS, secured by EigenLayer operators. Operators validating Anzen Protocol must arrive at consensus before SF data is delivered.

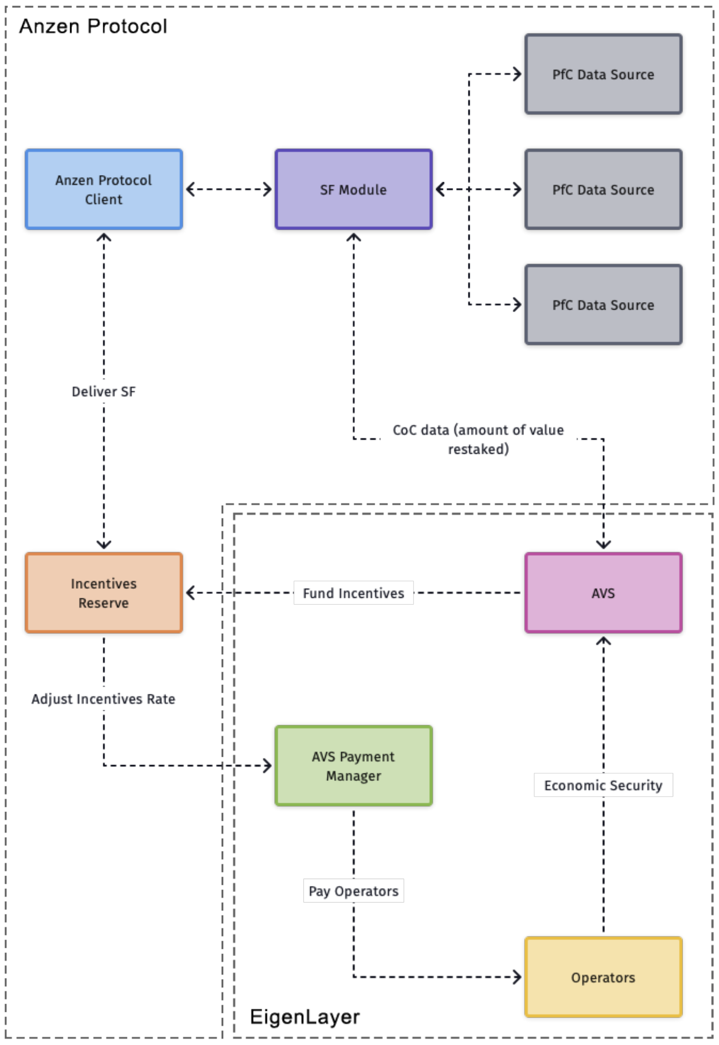

Figure 1: Protocol Architecture

5.1 SF Modules

Within Anzen Protocol, each integrated AVS has a separate SF Module. The Module contains the code to calculate and return the PfC, CoC, and SF. The Module performs any queries and calculations needed to obtain the required result.

5.2 Protocol Client

The Anzen Protocol Client is responsible for querying all SF Modules, participating in consensus, and delivering the updated data to the destination contract. The Client must perform these duties correctly as determined by consensus, or it may be subject to slashing conditions. If an operator running the Client sends malicious responses to queries, they are subject to slashing.

5.3 Protocol Governance

Before an SF module is added to the protocol, the AVS submits an initial governance proposal. Thereafter, the module undergoes review and audit by third parties in a multi-stage process (outlined in figure 2). Once the governance proposal passes, the SF module is added to Anzen Protocol.

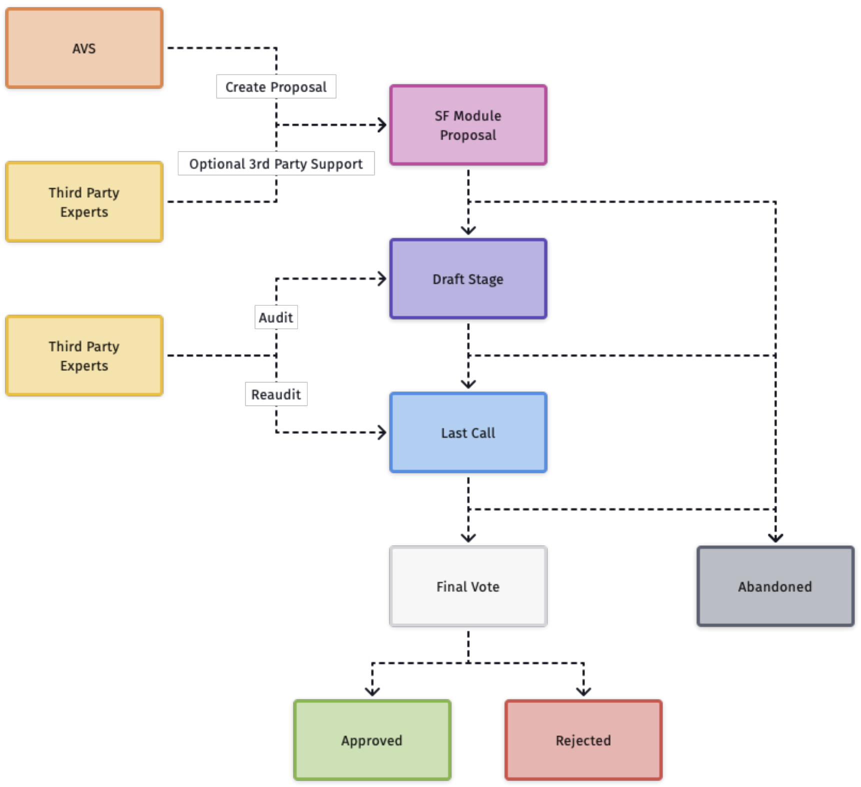

Figure 2: Governance

6 Operator Payments

In this section we explore a model for paying operators to complete tasks based on a flow rate of tokens. The entities and functions involved are formally defined in this section. As value is generated and accrued by the normal business operations of an AVS, token incentive reserves are replenished. The rate of token distribution optimizes long-term protocol sustainability and fair compensation for operators.

- **Tasks:** $\{t_1, t_2, \ldots, t_n\}$
- **Operators:** $\{o_1, o_2, \ldots, o_k\}$
- **Flow Epochs:** $\{f_1, f_2, \ldots, f_l\}$

6.1 Task Share s_{ij}

s_{ij} defines the share of task t_i allocated to operator o_j. It satisfies the following conditions:

$$0 \leq s_{ij} \leq 1, \quad \forall i \in [1, n], \ \forall j \in [1, k]$$

$$\sum_{j=1}^{k} s_{ij} = 1, \quad \forall i \in [1, n]$$

6.2 Task Weight w_i

w_i defines the weight of task t_i. It can take any non-negative value.

$$w_i \geq 0, \quad \forall i \in [1, n],$$

6.3 Flow Epoch Definition

Each flow epoch f_j defines the number of tokens being distributed and contains a subset of tasks. Let T_j be the set of tasks in flow epoch f_j.

$$T_j \subseteq \{t_1, t_2, \ldots, t_n\}, \quad \forall j \in [1, l]$$

It is Anzen Protocol's stated goal to optimize f_j balancing protocol sustainability, security, and operator compensation.

6.4 Operator Payment Calculation

The amount of tokens A_{ij} distributed to operator o_j during flow epoch f_i is defined as:

$$A_{ij} = f_i \times \sum_{t_i \in T_j} \frac{w_i \times s_{ij}}{w_i} \tag{5}$$

6.5 Aggregated Operator Payment Across Multiple Epochs

To evaluate the cumulative token distribution to an operator o_j for a subset of flow epochs from f_m to f_n, we can extend the previous payment formula. Let the set of flow epochs in this range be $F_{m:n}$:

$$F_{m:n} = \{f_m, f_{m+1}, \ldots, f_n\}$$

The aggregated amount $A'_{m:n,j}$ of tokens distributed to operator o_j for the tasks in epochs $F_{m:n}$ can be calculated as:

$$A'_{m:n,j} = \sum_{f_i \in F_{m:n}} \left(f_i \times \sum_{t_i \in T_j} \frac{w_i \times s_{ij}}{w_i} \right) \tag{6}$$

In this equation, $A'_{m:n,j}$ represents the total amount of tokens earned by operator o_j across multiple epochs $F_{m:n}$. The inner sum computes the number of tokens earned by the operator in a single epoch f_i, following the original formula. The outer sum aggregates these amounts over the range of epochs considered, from f_m to f_n.

7 Optimization of Payments f_{n+1}

In this model, the number of tokens f_n distributed in each epoch n is a critical parameter to optimize. This parameter is influenced by the Safety Factor (SF), which measures the robustness and sustainability of the system. In this section, we define the constants and variables involved in the adjustment of f_{n+1}, the tokens to be distributed in the next epoch:

- $SF_{\text{desired\_lower}}$: Lower bound of the desired Safety Factor range.
- $SF_{\text{desired\_upper}}$: Upper bound of the desired Safety Factor range.
- f_n: Tokens distributed in the current epoch n.
- f_{n+1}: Tokens to be distributed in the next epoch $n+1$.
- $ReductionFactor$: Factor by which f_n is reduced when SF is excessive.
- $MaxRateLimit$: Maximum allowable rate of increase for f_{n+1} to preserve reserves.

7.1 Case 1: Adequate Safety Factor

In this scenario, the Safety Factor is within the desired range, indicating that the protocol is operating safely and sustainably with the current f_n. There's no immediate need to change f_{n+1}.

$$\text{If } SF_{\text{desired\_lower}} \leq SF \leq SF_{\text{desired\_upper}}, \text{ then } f_{n+1} = f_n \tag{7}$$

7.2 Case 2: Excessive Safety Factor

When the Safety Factor is higher than necessary, it suggests that the protocol could reduce f_n without compromising security. Lowering f_{n+1} will lead to cost savings and increased sustainability.

$$\text{If } SF > SF_{\text{desired\_upper}}, \text{ then } f_{n+1} = f_n \times \text{Reduction Factor} \tag{8}$$

7.3 Case 3: Inadequate Safety Factor

If the Safety Factor falls below the desired range, it indicates a need to increase f_n. However, increasing f_n rapidly could deplete reserves quickly. We propose using a reverse Dutch auction mechanism to gradually increase f_n.

$$\text{If } SF < SF_{\text{desired\_lower}}, \text{ then initiate reverse Dutch auction for } f_{n+1}$$

In this mechanism, f_{n+1} will start at a f_n and will increment at a controlled rate (Δf) until the SF returns to the desired range. To prevent rapid depletion of reserves, the rate of increase is capped by predefined limits.

$$f_{n+1} = f_n + \Delta f, \quad \frac{\Delta f}{f_n} \leq \text{Max Rate Limit} \tag{9}$$

This ensures that f_{n+1} doesn't increase too rapidly, providing a controlled method to restore the Safety Factor to an acceptable range.

8 Incentive Reserves

Anzen Protocol leverages the SF to trigger adjustments to the rate of incentives in response to SF deviations. The incentive rate adjustments are such that the SF progresses towards the desired level (see Figure 3 below). The desired SF and the formula to adjust the incentive emissions rate in response to a given actual SF are programmatic, allowing for a methodical approach to incentive management. The Incentive Reserves are funded by the AVS and interface with EigenLayer's Payment infrastructure.

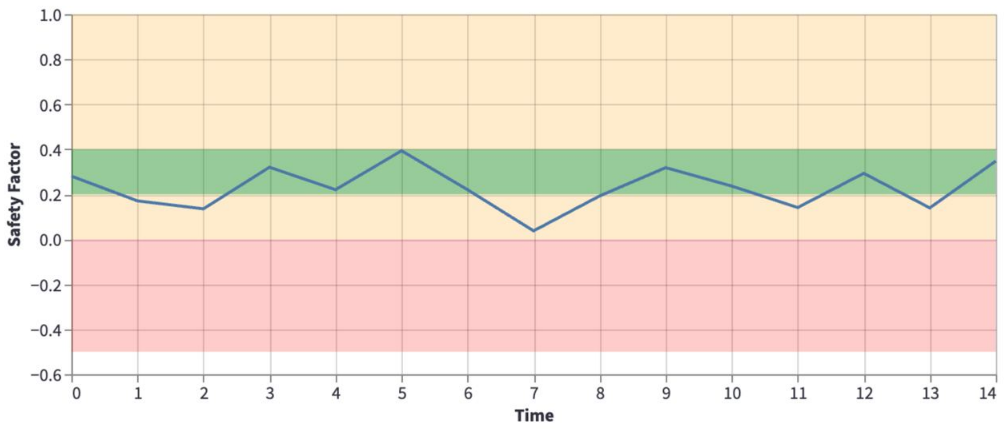

Figure 3: Safety Factor Optimization

8.1 Rate Limiting

The Incentive Reserves have rate limits enabled to set an upper and lower boundary for changes in the flow of incentive emissions. For instance, an Incentive Reserve can set a limit where no more than a 20% change in incentives per day may be triggered. On the lower bound, these rate limits are particularly important to secure the protocol and prevent a situation in which an attacker drastically reduces incentives, leaving the AVS vulnerable to economic attack. Defining these boundaries is part of the onboarding process for AVSs.

8.2 Fee Model

At launch, Anzen Protocol will not have fees enabled. The protocol may turn on the fee switch in the future based on protocol governance. Anzen's proposed fee model is to take a performance-based fee which is applied when savings are realized in the security budget of an AVS. The performance fee would only be applied when Anzen Protocol adjusts excessive incentives downward. The Protocol calculates the realized savings in emissions and charges a fee equal to a small percentage of savings. Additionally, there may be a small flat fee on top of the performance-based fee for each successful adjustment to ensure minimum costs are covered. Fees will be used to incentivize operators to validate Anzen Protocol.

9 Protocol Benefits

Anzen Protocol offers several advantages, with enhanced economic security being foremost among them. The protocol aims to reduce the likelihood of economic vulnerabilities that could be exploited. Additional benefits include:

9.1 Robust Security

When economic security decreases due to normal fluctuations, it is important for AVSs to adjust incentives and maintain economic security close to the SF. Real time automated adjustment of incentives helps ensure continuous economic security.

9.2 Monitoring

It is important for AVSs to have the ability to forecast how their incentive parameters can affect the overall estimated economic security they receive. This can greatly help with onboarding new AVSs onto EigenLayer. AVSs should be able to monitor detailed metrics regarding their SF, CoC, PfC, rate limits, and prior incentive adjustments. Additionally, SF data may be displayed in the Nethermind AVS dashboard. It is also important for AVSs to receive notifications in the event of unusual activity, so they can respond directly.

9.3 Protocol Sustainability

Part of the monitoring process involves comparing protocol revenue to incentive emissions spent on EigenLayer economic security. This ensures that protocols are distributing incentives sustainably and that the protocol remains economically viable over the long term, without excessively depleting its reserves.

9.4 Transparency

Users can monitor the current Safety Factor (SF) at any time, allowing them to assess the level of economic security provided by the protocol. This transparency fosters trust and enables users to make informed decisions about their continued participation in the system. A greater understanding of the underlying safety metrics can improve user confidence and protocol growth.

References

[1] EigenLayer Team (2023) *EigenLayer: The Restaking Collective*, https://docs.eigenlayer.xyz/overview/whitepaper

[2] Vitalik Buterin et al. (2023) *Ethereum Whitepaper*. https://ethereum.org/en/whitepaper/

[3] Hart Lambur, Matt Rice et al. (2020) *UMA Data Verification Mechanism: Adding Economic Guarantees to Blockchain Oracles* https://github.com/UMAprotocol/whitepaper/blob/master/UMA-DVM-oracle-whitepaper.pdf

[4] Sreeram Kannan, Soubhik Deb (2023) *The cryptoeconomics of slashing*, https://a16zcrypto.com/the-cryptoeconomics-of-slashing/

[5] Nethermind Team (2023) *Introduction: EigenLayer AVS Node Specification* https://eigen.nethermind.io/docs/spec/intro

[6] Luís T. A. N. Brandão (2023) *Multi-Party Threshold Cryptography* , https://csrc.nist.gov/Projects/threshold-cryptography