Securing Cross-Blockchain Oracles with Restaking

Max Kobelt

2025

Cross-blockchain oracles (CBOs) are systems that bridge information between otherwise isolated blockchains by replicating (a subset of) the state from a source blockchain on a target blockchain. The availability of the source blockchain's state on the target blockchain opens up a variety of use cases, for example proving that a payment was made on the source blockchain or enabling cross-chain smart contract interactions. Since neither of the two participating blockchains is aware of their counterpart, a CBO has to be constructed such that applications on the target blockchain using the transmitted state can be confident that the data they operate on is truthful.

Restaking, introduced by EigenLayer [1], is the method of reusing already locked stake by extending the range of protocols it secures. Validators can opt into securing any restaking-compatible protocol by posting an on-chain registration and downloading additional software to participate in the protocol. Broadening the range of protocols secured by a validator's stake improves the stake's capital efficiency and increases staking rewards. On the other hand, validators are exposed to higher operational expenses and more slashing risks.

Proof-of-stake (PoS) CBOs are instances of multi-party CBOs which assign voting power to validators based on their stake locked with the protocol. The security of these CBOs is derived from the high attack costs of acquiring enough stake to corrupt the system by overturning the votes cast by honest validators. The initial attraction of stake is difficult: potential validators may have already locked their stake in other protocols and users refrain from interacting with the CBO because of the low value of its locked stake.

By applying restaking to PoS CBOs, the bootstrapping problem can be alleviated due to the ability of validators to secure more than one protocol with the same stake. In this report, we explain our solution of a restaking-based CBO, starting with its architecture in Section 1. Afterward, we describe our prototype implementation in Section 2 and finish with the problems we encountered during the research and development in Section 3.

1 Architecture & Protocol Flow

The system that we present here uses the work of [2] (*zkOracle* from here) and modifies it to support restaking. zkOracle is a PoS CBO that collects votes concerning the state of a source blockchain and publishes the majority decision on-chain on a target blockchain. For efficiency reasons, zkOracle uses an off-chain aggregation mechanism to collect the votes and only posts a cheaply verifiable zero-knowledge proof on-chain, proving the protocol's correct execution.

The smart contracts of zkOracle are intended to be deployed on just one blockchain, the target blockchain. In our system, though, registration and deregistration, as well as stake slashing, are supposed to be handled on a potentially distinct blockchain.

Three blockchains are participating in our proposed system:

- The source blockchain on which transactions are made that are of interest to others
- The target blockchain which contains the main oracle functionalities, including the replication of the source blockchain's state
- The restaking blockchain, hosting a smart contract for managing the stake and handling validator registration and deregistration

Since the protocol state is split up across two blockchains, we need a mechanism to ensure that it remains synchronized. Therefore, the restaking blockchain and the target blockchain are also connected via a unidirectional relay, such that the state of the restaking blockchain can be verified on the target blockchain.

When a validator wants to register themself, they begin by assigning their stake to the protocol on the restaking blockchain. Similarly to zkOracle, this process only succeeds when the validator set is not at its capacity yet or the new validator's locked stake is higher than the stake of some other active validator, leading to a replacement.

Once a validator is registered, their stake is assigned to the oracle. At this point, the validator is still unable to receive validation rewards since the state submission and reward distribution take place on the target blockchain. To get registered on the target blockchain, they have to prove their registration on the restaking blockchain by leveraging the relay.

As long as all validators behave honestly (that is, their responses to the same request are equivalent) the submission procedure only takes place on the target blockchain and follows the same logic as zkOracle. If a validator misbehaves, two balances need to be slashed: The accumulated rewards of the validator on the target blockchain and the locked stake of the validator on the restaking blockchain. For both cases, the current aggregator (the validator collecting votes from its peers) computes and publishes a zero-knowledge proof on the target and restaking blockchain, leading to the respective slashes.

When a validator wants to exit the oracle and withdraw their stake and accumulated rewards, they first have to announce this intention on the restaking

blockchain. As with zkOracle, this action does not lead to an immediate withdrawal but marks the beginning of an exit period upon whose end the validator can withdraw their stake. The usage of an exit period is important because a malicious validator could repeatedly replace a validator and exit the oracle to gain control over the complete validator set. Additionally, since the restaking and the target blockchain operate asynchronously, there must be enough time for the exiting validator to get removed from the validator set on the target blockchain. Otherwise, the exiting validator could continue accumulating rewards while having no stake locked on the restaking blockchain, thus increasing the likelihood of misbehaving.

To ensure that the validator set remains synchronized between the two parts of the oracle, propagating changes to the validator set on the restaking blockchain to the target blockchain needs to be incentivized. For registration (and replacement), the newly registered validator is incentivized to relay the change to the target blockchain since otherwise their stake can not be used to generate validation rewards. When exiting the oracle, as previously explained, the action has to be announced on the restaking blockchain. From this point on, any validator on the target blockchain can prove the intention of the exiting validator by using the relay and receive a fixed reward for reporting the exit. Therefore, all validator set changes on the restaking blockchain should be propagated to the target blockchain quickly, assuming that initiators of such state changes behave rationally.

2 Prototype

The prototype we provide contains zero-knowledge arithmetic circuits and Solidity smart contracts to implement the proposed system. In comparison to the zkOracle prototype[1], our prototype adds a circuit for proving validator misbehavior on the restaking blockchain and splits up the logic of zkOracle across two blockchains, as explained in the previous section.

While zkOracle stores validator states in one Merkle tree, our prototype needs two Merkle trees: One for validators on the restaking blockchain (storing the locked stake) and one for validators on the target blockchain (storing the current accumulated rewards). Consequently, a given zero-knowledge proof valid on one blockchain can not be reused on the other because the Merkle trees (which are part of the proof inputs) store different balances.

The arithmetic circuit proving validator misbehavior on the restaking blockchain is a combination of the aggregation circuit (proving a majority vote) and the slashing circuit of the target blockchain. Since the restaking blockchain is not aware of user requests submitted to the target blockchain, the circuit first has to assert that a majority of validators agreed on a specific decision and, afterward, that there is one validator that posted a diverging response.

To communicate the occurrence of a certain change to the validator set from the restaking blockchain to the target blockchain, we make use of EVM event

[1] `https://github.com/soberm/zkOracle`

logs which can be used in conjunction with the relay on the target blockchain to prove the change. Each of these events contains the data required to also apply the change on the target blockchain and a nonce. The nonce is used to ensure that the events are applied in the correct order on the target blockchain.

3 Encountered Problems

Initially, the system was intended to be designed such that it could be integrated into EigenLayer. As it turned out though, EigenLayer expects actively validated services (AVSs), the systems in need of stake to support their security, to submit their rewards to the EigenLayer core contracts which would complicate the reward distribution process[2]. Additionally, EigenLayer distinguishes between operators and delegators, where operators are the parties directly participating in AVS operations and delegators only delegate their stake to a specific operator. Operators participate directly in our CBO, thus, rewarding them on the target blockchain would be possible. On the other hand, delegators should not be required to have an identity on the target blockchain which further complicates the integration into EigenLayer. A complicated solution to this problem could consist of bridging funds from the target blockchain to the restaking blockchain and distributing these funds accordingly to operators and delegators. For the aforementioned two reasons, EigenLayer integration is not easily achievable with the current design.

It also seems difficult to find a good exit procedure. As we have implemented it currently in the prototype, each validator needs to store a minimum amount `MIN_BALANCE` of funds in the oracle contract on the target blockchain such that rational actors relay an exit and get rewarded with `MIN_BALANCE`. Since `MIN_BALANCE` is fixed though, in extreme situations the incentive for relaying exits may not be sufficient. Another solution would be to let validators sign a message confirming that an exit took place on the target blockchain. The validator that just exited could then post the other validators' signatures on the restaking blockchain and claim their stake. The problem with this approach is that validators could arbitrarily decide to censor exits. Another solution, which is similar to the previous one, would make use of a relay from the target to the restaking blockchain. We think though that requiring the existence of another relay for just this purpose has too much overhead compared to the value that it provides.

References

[1] EigenLayer Team. *EigenLayer: The Restaking Collective*. 2022. URL: `https://docs.eigenlayer.xyz/assets/files/EigenLayer_WhitePaper-88c47923ca0319870c611decd6e562ad.pdf` (visited on 02/23/2025).

[2] `https://github.com/Layr-Labs/eigenlayer-contracts/blob/decf99caab298592d157a454c225286bd4491093/docs/core/RewardsCoordinator.md`

[2] Michael Sober, Giulia Scaffino, and Stefan Schulte. "Cross-Blockchain Communication Using Oracles With an Off-Chain Aggregation Mechanism Based on zk-SNARKs". In: *Distributed Ledger Technologies: Research and Practice* 3.4 (Dec. 2024). DOI: 10.1145/3678187.

Glossary

AVS actively validated service. 4

CBO cross-blockchain oracle. 1, 2, 4

PoS proof-of-stake. 1, 2