The DendrETH project has implemented a zero-knowledge circuit capable of proving successful state transitions based on the beacon chain light client sync protocol. The project is currently aiming to implement a zero-knowledge circuit capable of proving the Casper finality conditions by processing the attestation messages of the entire Ethereum validator set. This work has led us to explore efficient constructs for implementing recursive zero-knowledge circuits for BLS aggregation, validator set membership tracking, validator duties tracking and more.

We believe that these building blocks are well-suited for solving a pressing problem that all liquid staking protocols face: How can a smart contract keep track of the on-chain performance of the created validators without relying on a set of trusted oracles that inevitably introduce additional security risks and economic overhead to the protocol?

In particular, after the introduction of EIP-4788, all liquid staking protocols will benefit from the presence of security-audited zero-knowledge circuits providing the following proofs:

1. Proof of Total Locked Value

Given a particular withdrawal address used by the liquid staking protocol, provide a proof for the sum of the balances of all active validators on the beacon chain that use the specified withdrawal address.

With support from Diva Staking, the DendrETH team has developed an end-to-end prototype for this proof, using a highly-efficient recursive circuit based on PLONKY2. Please see the “Technical Details” section of this post for further details.

2. Proof of Total Rewards Potential

Given a merkle accumulator containing a set of validator public keys (this scales from a single validator to the entire set of validators created by the liquid staking protocol), provide a proof that indicates the maximum number of rewards that the set of validators were eligible for within the canonical finalized history. This takes into account the block proposal duties and the sync committee duties of the validators.

Since the maximum possible profit from a block cannot be determined without knowledge of all attestations that were broadcast in the network, we assume that the potential reward is equal to a running average of the last N blocks leading up to the slot of the proposal.

Similarly, since the maximum MEV profit from a block cannot be known, we can model it as a public input for the circuit which can be set by the liquid staking protocol (and potentially managed dynamically).

Please note that the circuit can verify the presence of a transaction within the block that distributes the MEV rewards to the liquid staking protocol, but unfortunately this doesn’t rule out the possibility that the proposer was paid some additional sum by the builder out-of-band. Some of these difficulties in the tracking of MEV rewards are likely to be resolved in the planned Proposer-Builder Separation upgrade of Ethereum that will enshrine the MEV distribution within the base protocol.

3. Proof of Poor Validator Performance

Given a Proof 2

obtained for a particular validator as described above, provide a proof that the validator has earned less than a target

percentage of the maximum rewards (e.g. 90%). Such a proof can be used to penalize or evict particular operators from the protocol. The circuit compares all recorded withdrawals of the validator (while taking into account any deposit top-ups) to the total rewards potential

to determine whether the validator is meeting the target performance.

Rationale

The development of the proposed zero-knowledge circuits is motivated by several key factors and considerations.

Firstly, the reliance on trusted oracles in liquid staking protocols poses a significant risk. These oracles have the power to influence token prices within the protocols, potentially favoring certain users and shifting profits disproportionately. This centralized control contradicts the fundamental principles of decentralization and trustlessness that underpin blockchain technology.

Secondly, the oracles in question control a substantial amount of funds within the protocol. To ensure their continued participation and prevent potential defection, it is essential to provide them with adequate compensation. Additionally, given the substantial power they possess, the oracles become prime targets for hackers. This necessitates significant investment in maintaining their security infrastructure, further contributing to the overhead associated with the existing liquid staking protocols.

In practice, the current costs associated with oracles in the already deployed liquid staking protocols have been subject to criticism, raising concerns regarding the economic efficiency and viability of the protocols. Therefore, it becomes crucial to

explore alternative approaches that can address these concerns and potentially reduce costs while enhancing overall security and trustlessness.

The zero-knowledge circuits envisioned in this proposal can be developed as a public good - a reusable solution that can potentially replace the use of trusted oracles in all existing and future liquid staking protocols. When leveraged, this solution will eliminate the variable costs associated with trusted oracles in favor of the fixed cost of verifying zero-knowledge proofs on chain (expected to be less than 500K gas per update).

By introducing these trust-minimized oracle alternatives, the DendrETH team aims to enhance the security, decentralization, and economic efficiency of liquid staking protocols. The reduction in reliance on trusted oracles and the associated costs can lead to a more robust and sustainable ecosystem, fostering greater user confidence and participation.

Technical Details

Integration through a Fixed Set of Withdrawal Credentials

If the liquid staking protocol uses a fixed address in the withdrawal credentials of all created validators (or a small fixed set of addresses), then no modifications are necessary to the smart contracts responsible for deposits. The DendrETH zero-knowledge circuits will scan the entire set of active validators, identifying all validators using the specified withdrawal credentials.

Integration through a Merkle Accumulator for the Validator Set

If the liquid staking protocol leverages multiple validator operators and certain proofs needs to be applicable to a particular operator, the smart contracts responsible for executing deposits can to be modified to keep track of the validator set of each operator by recording the root hash of a binary SHA256 merkle tree constructed over an append-only list of (validator_public_key, deposit_index)

tuples. To compute this value incrementally, the smart contract needs to store only log(N)

SHA256 values (tree branches) in a very similar fashion as theofficial deposit contract.

The DendrETH team offers a reference reusable implementation here:

github.com

metacraft-labs/DendrETH/blob/main/beacon-light-client/solidity/contracts/validators_accumulator/ValidatorsAccumulator.sol

// SPDX-License-Identifier: MIT pragma solidity 0.8.18;

// This interface is designed to be compatible with the Vyper version. /// @notice This is the Ethereum 2.0 deposit contract interface. /// For more information see the Phase 0 specification under https://github.com/ethereum/eth2.0-specs interface IDepositContract { /// @notice A processed deposit event. event DepositEvent(bytes pubkey, bytes withdrawal_credentials, bytes amount, bytes signature, bytes index);

/// @notice Submit a Phase 0 DepositData object. /// @param pubkey A BLS12-381 public key. /// @param withdrawal_credentials Commitment to a public key for withdrawals. /// @param signature A BLS12-381 signature.

This file has been truncated. show original

Proof of Total Locked Value

Public Inputs

The following public inputs must be supplied on-chain during the verification of the proofs:

- withdrawalCredentials

: The fixed set of withdrawal addresses, used by the protocol.

- stateRoot

: The current beacon state root against which the proof was generated (obtained through the mechanism described in EIP-4788).

- totalLockedValue

: The total locked value amount that will be verified by the circuit (this value is expected to be stored in the contract and

updated with a transaction that also submits a corresponding ZK proof).

Circuit Logic

The circuit performs the following verification steps:

1. BeaconState Fields Verification

: The circuit uses SSZ merkle proofs to verify the following additional private inputs against the provided stateRoot

:

- currentEpoch

matches the slot

field of the state (by dividing by SLOTS_PER_EPOCH

)

- currentValidatorsRoot

matches the validators

field of the state (as a SSZ hashTreeRoot

)

- currentBalancesRoot

matches the balances

field of the state (as a SSZ hashTreeRoot

)

1. Commitment Mapping

: A general-purpose commitment mapping circuit establishes a correspondence between currentValidatorsRoot

value (a SSZ hashTreeRoot

) and a Poseidon root hash of an equivalent tree of records (currentValidatorsRootPoseidon

). Similarly, we perform an incremental computation of a Poseidon commitment map for the validatorAccumulator

(validatorAccumulatorPoseidon

).

1. Validator Status and Balance Verification

: The circuit receives up-to-date validator details as private inputs and verifies them with Poseidon merkle proofs against the currentValidatorsRootPoseidon

value. A validator is relevant for the total locked value computation only if its withdrawal credentials match the withdrawalCredentials

public input and its activation and exit epochs enclose the currentEpoch

. Similarly, for each validator we process a SSZ merkle proof against currentBalancesRoot

and the balance of the relevant validators is summed up in rangeTotalValue

. In the array of validator details, we require that the validator indices are strictly increasing and the circuit produces a tuple with the following values:

- rangeStart

(the first validator index in the array of supplied validator details)

- rangeEnd

(the last validator index)

- rangeTotalValue

(the summed balance of all relevant validators)

1. Range Aggregation

: The circuit combines a number of ranges by verifying that the provided indices are strictly adjacent. rangeTotalValue

is summed up until we construct a range spanning the entire validator set.

1. Hashing and Output Generation

: The public inputs are hashed. The circuit outputs the first 253 bits of the SHA-256 hash of the public inputs to make circuit verification more cost-effective.

Experimental Results

We tested our prototype circuits on a desktop computer with Intel i9-12900K CPU and on various cloud instances. The proving time is dominated by steps 3 and 4 which are embarrassingly parallel problems. On the i9 CPU, the Validator Status and Balance Verification

proof requires 700 ms

for 8 validators

, while each Range Aggregation

takes 1200 ms

(the depth of the tree of recursive calls is 38 levels).

Thus, with unlimited parallelism the complete proof can be computed in less than 1 minute theoretically, but we will aim for a more practical iterative computation where each update is delivered with up to 10 minutes of latency

.

With the current validator set on mainnet, the compute time on the i9 CPU for a single proof is less than 60 hours. Assuming a single update per day, we will aim to create operator software that leverages the modern cloud offerings to achieve operational costs of less than 50 USD per day

.

For the final verification of the PLONKY2 proofs on chain, we plan to use a verifier implemented as a PLONK circuit in order to obtain a low gas cost per update (less than 500K

) without requiring a trusted setup. Please note that the gas costs will be fixed and not dependant of the size of the Ethereum validator set.

Project Timeline

- August 2023
- Functional proof of concept implementation of the “Total Locked Value” circuit that provides realistic estimates for the operational latency and cost of the final solution.
- September 2023
- Final version of the “Total Locked Value” circuit with suitable documentation for audit purposes. Early prototype of the operator software.
- October 2023
- First production-ready version of the operator software, capable of creating proofs for total locked value. The software is likely to depend on the services of cloud operators such as AWS.
- Rest of 2023
- Audit and further improvements to the operator software.

- 2024 Q1/Q2
- Development of the “Total Rewards Potential” and “Poor Validator Performance” proofs.

Q&A

Will the resulting project be open source?

All components of the system will be fully open source (GPLv3). We will strive to provide comprehensive guides for operating instances of our proof generators and relay nodes, as well as easy-to-use packages for most operating systems (i.e. docker images).

Who is doing the work?

The work will be carried out by the existing DendrETH team, consisting of (in alphabetical order):

- Dimo Dimov
- Emil Ivanichkov
- Kristin Kirkov
- Petar Kirov
- Simeon Armenchev
- Stefan Nikolov
- Yordan Miladinov
- Zahary Karadjov

What is the background of the person(s) doing the work? What experience do they have with such projects in the past?

Zahary Karadjov is the current team lead of the Nimbus project. Petar Kirov is a former CTO of Jarvis Network. The rest of the team has accumulated a lot of relevant experience during their work on the DendrETH project so far.

How will the work be maintained after delivery?

We believe the components developed by the DendrETH team will be widely used in critical cross-blockchain communication infrastructure in the future, which will ensure their continued development and maintenance.

What is the proposed grant amount?

We are requesting 50,000 DAI from the LIDO community (50% up-front and 50% upon reaching the October 2023 milestone). We are deeply grateful for the support we received so far from Diva Staking and the Ethereum Foundation. We believe we are working in extremely cost-effective manner, seeking synergies between multiple project goals and optimizing our work for as wider adoption as possible (in order to reduce the burden on each individual community that benefits from it).

What is the acceptance criteria?

There is ready-to-use and well documented software for the creation and verification of the proofs proposed in this document. The proof latency time and the gas costs are the same or better as the experimental results shared in this proposal.

The availability of a proof can be verified through a simple smart contract that can be queried for a particular slot number, returning the verified total locked value.

How will the community verify that the work delivered matches the proposed cadence?

We will produce regular technical demos and proof-of-concept deployments of our work as soon as we reach each milestone.

What other protocols, DAOs, projects, or individuals, would stand to benefit from this grant?

We aim to create a solution that will be applicable in all current and future liquid staking protocols.

Furthermore, certain components of our solution such as the generalized commitment mapper will enable the development

of a wide range of projects that aim to take advantage of the functionality introduced in EIP-4788.

Finally, the work delivered under this grant will certainly enhance the quality of all components developed by the DendrETH team in the pursuit of wider goals such as implementing zero-knowledge proofs for the Casper finality conditions in order to create more secure cross-chain bridges and one-shot syncing solutions for highly convenient and secure Ethereum clients.