Summary: Lido Oracle contract is trustful - it relies on trusted third-parties and quorum mechanism for reports to maintain its state. It is possible to make the contract trustless, allowing any third party to adjust the contract state, and utilizing Zero Knowledge proofs to mathematically secure the correctness and validity of the change. This should lead to better security, cost savings and scalability.

Longer: Lido oracle contract is trustful - it relies on external oracle(s) to honestly calculate the total ethereum value committed by Lido validators (aka TLV), and uses quorum mechanism and a list of trusted members to protect from malicious actors.

This leads to three consequences:

- Security: A dedicated and resourceful attacker can work towards acquiring control of the majority of oracle members - and abuse the oracle contract when it is achieved.

** practically this is a 51% attack, however at the moment there are only 5 trusted oracles (see getOracleMembers in the contract), so 51% attack essentially boils down to overtaking just 3 entities. There's a proposal to increase the number of oracle members to 11 - this should make it considerably harder (need overtaking 6 entities), but still within a realm of practically feasible.

- Cost: Contract requires a considerable amount of expensive storage read/write operations to manage members, check if reports are coming from a trusted source, and keep track of reports while quorum is being accumulated.
- Scalability: With the network growing. the cost of quorum calculation grows linearly (O(N)) with the number of trusted oracle members

A trustless, ZK-proof secured approach can address these shortcomings in the following way:

- Security: With proper construction, zk-proof can ensure that only honest calculations produce input that would pass validation. This will make quorum and membership management unnecessary - any input that passes validation can be trusted to be coming from an accurate and honest calculation.
- Cost: With ZK-based solution, input validation can be limited to a small number (2-4) checks against Merkle Tree Root keccak hashes (obtained from a 1st party trustworthy contract(s) or provided by Ethereum network) + L1 contract invocation to confirm ZK-proof validity.
- Scalability: since all honest and accurate calculations should produce the same outcome (there's only way to sum all staked balances), it is sufficient for the contract to receive a single valid update (per some time frame - e.g. epoch), and don't need multiple parties to report. This removes the need to scale the number of trusted parties as the network grows.

More concretely, this can be implemented utilizing StarkWare'sCairo programming language and verifier. The solution will contain three pieces:

- Cairo program to produce total value locked report and necessary ZK-proofs (Merkle Tree Roots)
- L1 Ethereum contract to replace current oracle contract; most likely not much more complex than a "toy" contract used in one of the Cairo tutorials - essentially just a few lines of code.
- Trustless oracle - it would gather the input for the Cairo program, run it, obtain proof of correctness from SHARP, and send them to the contract.