Pistis Oracle Litepaper

- What is Pistis Oracle?
    - Pistis Oracle is a ZK oracle implementation based on EigenLayer Shared-Security.
    - Pistis Oracle enables modular blockchain without the need for developers to build their own node networks. It utilizes the shared security of EigenLayer and Ethereum to ensure the security of the oracle network, while leveraging ZK technology to ensure reliable and stable output of oracle data. The shared security implementation of ZK oracle based on EigenLayer also addresses the challenges of price oracles.
- Why build with EigenLayer?
    - With EigenLayer, Ethereum stakers can help secure many services by restaking their staked ETH and opting-in to many services simultaneously, providing pooled security**.** Reusing ETH to provide security across many services reduces capital costs for a staker to participate and significantly increases the trust guarantees to individual services.
    - Anyone building a new decentralized service for Ethereum must bootstrap a new trust network to secure their system, fragmenting security. EigenLayer solves this problem by enabling any service, regardless of its composition (e.g. EVM-compatibility), to tap into the pooled security of Ethereum's stakers, creating an environment for permissionless innovation and free-market governance.
- What problems there are in the current oracle landscape?
    - Need to build a new blockchain trust network to maintain decentralization and counter single point failures. It is difficult to bootstrap a new trust network.
    - Need significant initial funding to maintain token market value for oracle security. If token market value is too low, it can be easily manipulated to cause a sharp decline in price. This can lead to panic selling of the token

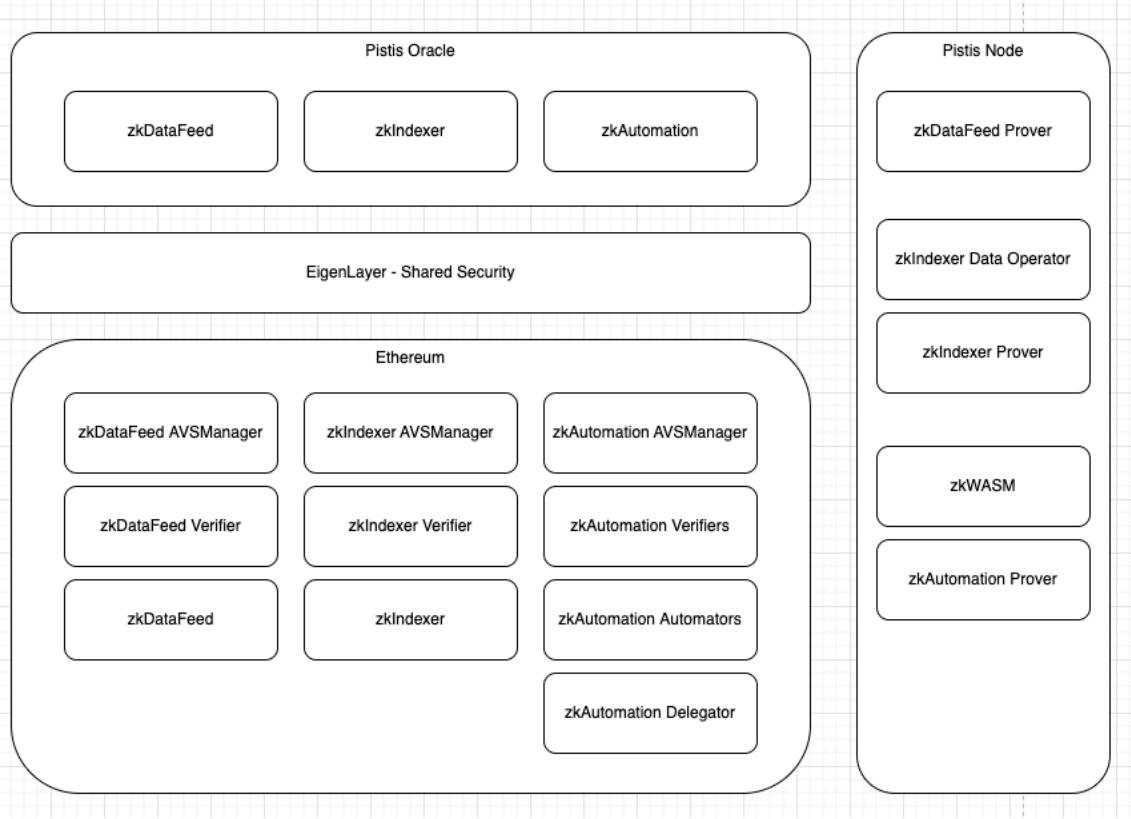
by node validators and their withdrawal from the network, resulting in instability of the oracle network.

    - Nodes may be controlled by a centralized organization with whitelist. If the centralized organization is compromised, the oracle network is compromised.
    - The data that collected by oracle may itself have been damaged or corrupted due to data source forgery or attacks.
    - The data reported by an oracle needs to be generated through the consensus of the blockchain, and achieving consensus among the various nodes of the blockchain takes time. According to the blockchain's "impossible trinity" theory, in order for the oracle to reliably and consistently report data, it would result in high latency.

- How does Pistis Oracle solve the problems?
    - Implements a modular blockchain layer based on EigenLayer, oracle off-chain nodes only needs to focus on executing layer logic and interacting with on-chain contracts, the trust network is entrusted to EigenLayer, making it more secure and reliable.
    - Utilizes EigenLayer Shared-Security, oracle economic security can be bootstrapped with existing Ethereum validators, without requiring significant initial funding to maintain token market value for oracle security.
    - Any EigenLayer Operator can register as a validator node and perform tasks for oracle off-chain nodes without any permission requirements. To enhance decentralization, we will employ a token incentive strategy to encourage EigenLayer Operators from different regions and with different staked assets to register as validator nodes. For example, if a node is located in a region or holds staked assets that are relatively scarce among all validator nodes, that node will receive more token rewards upon registration. The reward ratio will be dynamically adjusted until a balance is achieved among all parties.
    - Utilizes ZKP to verify the data generation mechanism and authenticity to ensure the integrity and reliability of the data.

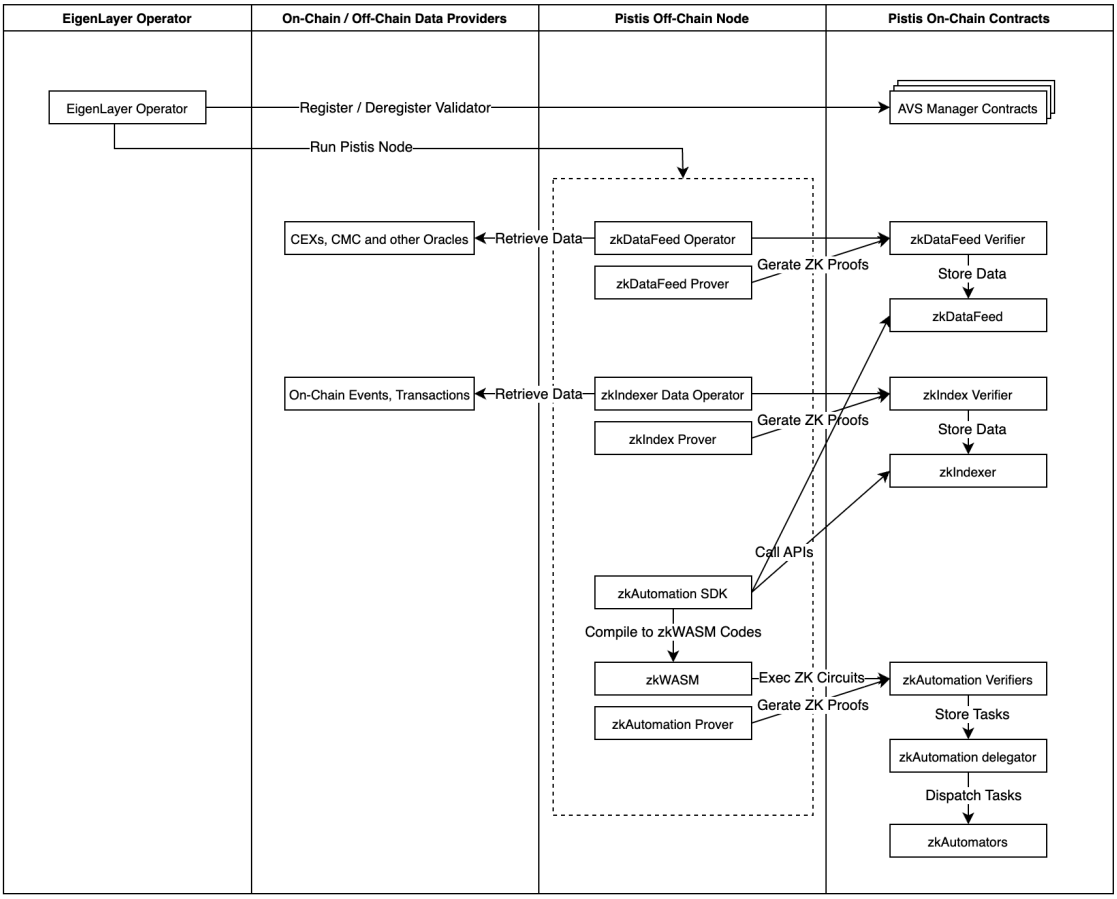
- Data is collected and ZK proofs are generated by off-chain nodes, then verified on-chain by smart contracts, enabling minimal trust without the need for a consensus network and achieving ultra-low latency. By leveraging parallel processing of ZKP, high throughput is achieved.

- How is Pistis Oracle implemented?
  - Architecture



    - Pistis Node: Retrieve data from on-chain / off-chain, compute ZK proofs for data generating, and save data and proofs to chain.
    - AVSManager Contracts: Manage EigenLayer operator's register/deregister actions, Pistis Node work payments, and EigenLayer operator slashing conditions.
    - zkDataFeed, zkIndexer and zkAutomation Contracts: Provide data feed, indexing and automation features on-chain.

- Verifiers Contracts: Verify the data that outputted by Pistis Node with ZK Proofs on-chain.

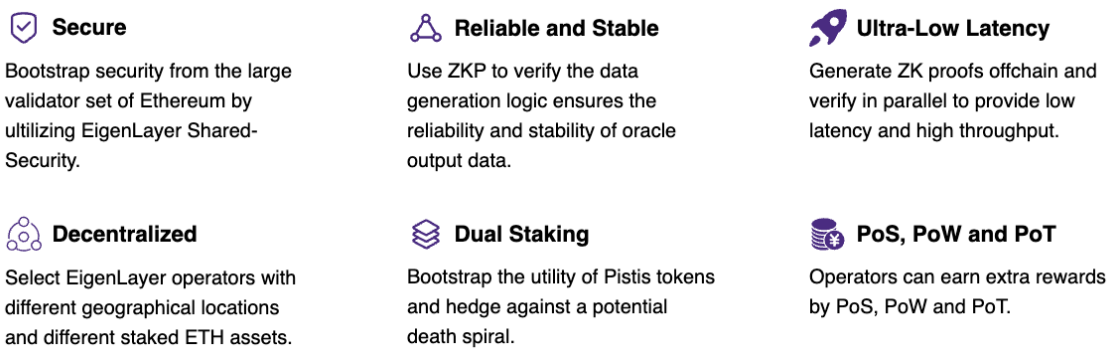


- Features

  Pistis Oracle contains three main features: ZK Data Feed, ZK Indexer, and ZK Automation.

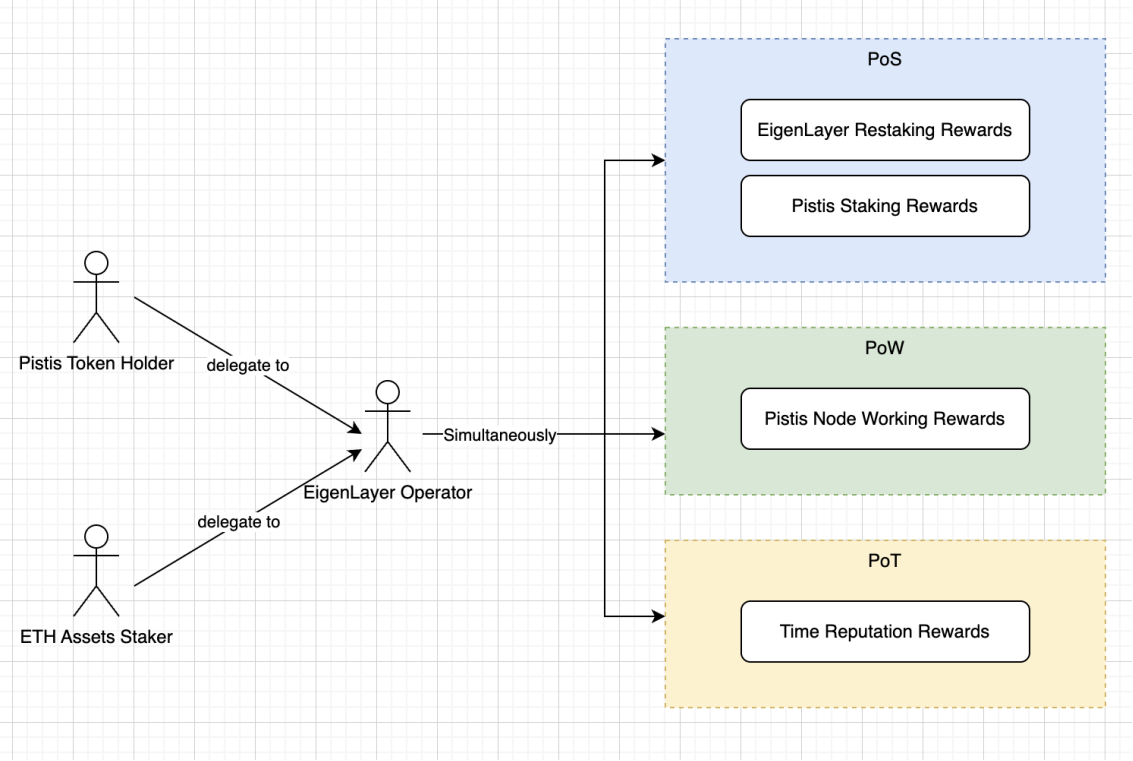
  - ZK Data Feed
    - Obtains real-time prices from decentralized oracles such as Deco based on TLS, as well as from data sources like CEXs and CMC.
    - Retrieves real-time prices from on-chain interfaces of platforms such as ChainLink, PythOracle, and SupraOracle.
    - Computes the weighted average price using ZK calculations and stores the computation result along with the proof on the blockchain.

        - ZK Indexer
            - Generates event circuits and on-chain verification contracts using ZKP to provide event data querying functionality for other dApps.
            - Generates transaction, receipt, account circuits, and on-chain verification contracts using ZKP for providing historical data querying functionality (transaction, receipt, account) to other dApps.
        - ZK Automation
            - Provides a programmable ZK execution environment for users through zkWASM.
            - Generates ZK circuits and proofs for the logic of querying prices, events, transactions, receipts, accounts from ZK Data Feed and ZK Indexer, as well as for executing calculations based on query results, and stores the circuits and proofs on the blockchain.
            - Automatically calls on-chain verification logic and executes on-chain automators when the off-chain logic satisfies certain trigger conditions and passes the verification successfully.
    - Advantages



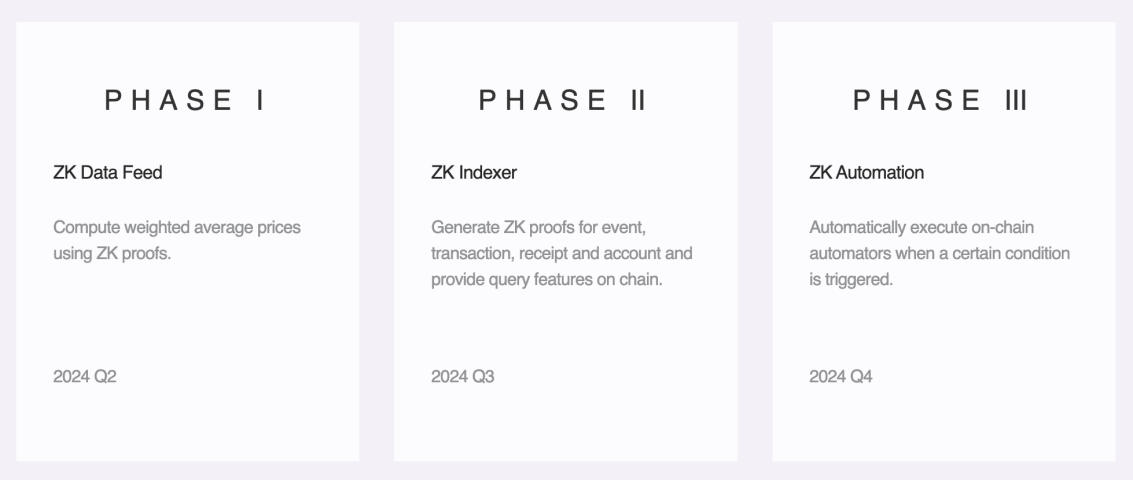
- Incentive mode introduction
    - Different roles in Pistis ecosystem
        - ETH Assets Staker: Restake LST / LRT assets to EigenLayer, operate nodes directly or delegate to other EigenLayer Operators to provide Shared-Security and node task executing services to Pistis Oracle.

        - EigenLayer Operator: Delegate ETH Assets Staker to participate in Pistis Staking, and execute Pistis node tasks.
        - Pistis Consumer: Call Pistis ZK Data Feed, ZK Indexer and ZK Automation services, and pay ETH assets for the services.
    - PoS, PoW and PoT
        - PoS
            - ETH Assets Staker: Restake LST / LRT assets to EigenLayer to earn EigenLayer restaking rewards.
            - EigenLayer Operator: Delegate ETH Assets Staker to participate in Pistis Staking to earn extra staking rewards, and and distribute rewards proportionally to ETH Assets Staker.
        - PoW
            - EigenLayer Operator: Execute Pistis Node tasks to earn working rewards. The node that completes the task first will receive the work reward.
        - PoT
            - EigenLayer Operator: If operators can provide stable services for an extended period, they will receive additional time reputation rewards.



- How does an EigenLayer Operator unbond?
  1. Submit a quit request on-chain.
  2. Wait for two weeks and continue to maintain stable work before the quit.
  3. If there are too many people applying to quit, there will be a queue for the quit process.
- EigenLayer Operator slashing conditions
  - Cannot provide continuous stable services before the quit.
  - Provide false data or generate fake ZK proofs.

- Future plan
  - Timeline & milestone



- Team
  - Team introduction
    - Extensive experience in ZKP.
    - Involved in multiple projects for ZK implementation and optimization.
    - Dedicated to building the Web3 infrastructure.
    - Located in Singapore and China.
  - Team members
    - Albert He, Project Manager.
      - Results-driven and highly skilled Web3 Project Manager with 15 years of experience in leading and delivering successful blockchain projects.
      - Managed project timelines, allocated resources, and monitored project progress, ensuring adherence to budget and timeline constraints.
      - Facilitated effective communication and collaboration among cross-functional teams, including developers, designers, and blockchain experts.
    - Alex Lee, Architect.
      - Highly skilled and experienced Web3 Architect with 10 years of expertise in designing and implementing decentralized solutions.

Proficient in blockchain technology, smart contracts, dApps, and Web3 frameworks.

    - Led a team of architects and developers, providing technical guidance, reviewing code, and ensuring best practices in Web3 development.
    - Developed smart contract architectures, ensuring optimal gas usage, code efficiency, and security.
    - Designed and implemented off-chain scaling solutions, leveraging technologies like state channels, sidechains, or layer-2 protocols.
- Joe King, Contract Developer.
    - Highly skilled and experienced Web3 Contract Developer with 3 years of expertise in developing smart contracts.
    - Implemented token contracts, including ERC-20, ERC-721, and custom token standards, with a focus on functionality, security, and gas optimization.
    - Designed and implemented contract logic for decentralized finance (DeFi) applications, such as lending, staking, and automated market makers (AMMs).
- Philip Smith, ZK and Backend Developer.
    - Highly skilled and experienced Web3 ZK Developer with 5 years of expertise in developing and implementing ZK proofs and privacy-enhancing technologies in blockchain applications. Proficient in zk-SNARKs, zk-STARKs, and other ZK frameworks.
    - Developed and deployed zk-SNARKs and zk-STARKs circuits for various use cases, such as confidential transactions, identity management, and data privacy.
    - Optimized and fine-tuned ZKP implementations for efficiency, reducing computational and storage costs.
- Steven Lau, Security Engineer.

- Highly skilled and experienced Web3 Security Engineer with 4 years of expertise in identifying and mitigating security vulnerabilities in decentralized systems. Proficient in blockchain security, smart contract auditing, and best practices in Web3 security.
- Performed smart contract audits to identify coding vulnerabilities, logic flaws, and potential attack vectors.
- Conducted audits and security reviews of ZKP systems, identifying and addressing potential vulnerabilities and attack vectors.