Building off Plasma Next

Abstract

Presenting a novel blockchain scaling solution by integrating the concepts of instant finality, similar to the Lightning Network, with the EigenLayer approach for efficient use of locked funds. This combination addresses various drawbacks associated with current blockchain scaling solutions, enhancing capital efficiency, reducing operational complexity, minimizing operator risk, and extending the scope of statelessness while providing instant, secure transaction finality. We provide rigorous mathematical formalisms, proofs, algorithms, and detailed analysis to demonstrate the viability and advantages of this approach.

Instant Finality

The system ensures that transactions are confirmed instantly, akin to the Lightning Network, providing immediate transaction finality and security. This is achieved through the use of off-chain payment channels and conditional payments verified by zero-knowledge proofs (ZKPs).

Definition 1 (Payment Channel):

A payment channel is a bilateral off-chain mechanism that allows two parties to conduct multiple transactions without committing each transaction to the blockchain. The initial and final states of the channel are recorded on-chain.

Definition 2 (Conditional Payment):

A conditional payment is a transaction where the transfer of funds is contingent upon the satisfaction of a predefined condition, which can be verified using a ZKP.

Unified Staking Pool

A unified liquidity pool is utilized, where funds are staked by operators and users and dynamically allocated across multiple channels and transactions, optimizing the use of locked funds. This approach improves capital efficiency and reduces the risk of liquidity shortages.

Definition 3 (Unified Staking Pool):

A unified staking pool is a smart contract that aggregates staked funds from multiple participants and dynamically allocates these funds across various payment channels based on demand.

Recursive Proofs and Smart Contracts

Recursive zero-knowledge proofs (ZKPs) and smart contracts are employed to securely manage and verify cross-channel transactions, ensuring efficient and scalable verification.

Definition 4 (Recursive ZKP):

A recursive ZKP is a proof system that allows for the aggregation of multiple proofs into a single proof, enabling efficient verification of cross-channel transactions.

Prerequisites

To fully understand the proposed solution, readers should be familiar with the following concepts:

- Blockchain technology and its scalability issues
- The Lightning Network and off-chain payment channels
- Zero-knowledge proofs (ZKPs) and their applications in blockchain
- Smart contract functionality and deployment

Implementation Details

Instant Finality Mechanism

Payment Channels:

We implement payment channels similar to the Lightning Network, where transactions between parties are confirmed off-chain instantly. The channel balance can be formally expressed as:

$$\text{Balance}_{\text{final}} = \text{Balance}_{\text{initial}} + \sum_{i=1}^{n} \text{Transaction}_i$$

Theorem 1 (Payment Channel Security):

Given a secure signature scheme and a reliable blockchain, payment channels guarantee that the final balance reflects the sum of all valid transactions, preventing unauthorized transfers.

Proof 1:

The proof follows from the security of the underlying signature scheme and the immutability of the blockchain. Each transaction is signed by both parties, ensuring authenticity and non-repudiation. The initial and final states of the channel are recorded on the blockchain, providing a tamper-proof ledger. Therefore, the final balance is secure and reflects the sum of all valid transactions.

Conditional Payments with ZKPs:

We utilize ZKPs to create conditions for payment finality that are verified instantly. A ZKP for a conditional payment can be represented as:

$$\text{ZKP}(\text{Transaction}_i, \text{Condition}) \Rightarrow \text{Transaction}_i \text{ is final}$$

Theorem 2 (Conditional Payment Security):

Given a secure ZKP system, conditional payments ensure that transactions are finalized if and only if the predefined conditions are met, preventing unauthorized transfers.

Proof 2:

The proof relies on the soundness and completeness of the ZKP system. Soundness ensures that a proof can only be generated if the condition is indeed satisfied, preventing false positives. Completeness guarantees that a valid proof can always be generated when the condition is met, preventing false negatives. Therefore, conditional payments are secure and finalize transactions only when the predefined conditions are fulfilled.

Unified Staking Pool and Dynamic Allocation

Stake Management:

Operators and users stake their funds into a unified pool managed by a smart contract. The dynamic allocation of staked funds can be formalized as:

Algorithm 1 Dynamic allocation of staked funds

Require: StakedFunds, Channels, Demand 1: for each Channel $\in$ Channels do 2: Allocate(Channel, Demand(Channel)) 3: end for

Lemma 1 (Allocation Efficiency):

The dynamic allocation algorithm ensures that staked funds are allocated to channels based on their demand, minimizing idle capital and optimizing liquidity utilization.

Proof 3:

The allocation algorithm iterates over all channels and allocates funds based on the demand for each channel. By dynamically adjusting the allocation according to demand, the algorithm minimizes the amount of idle capital and optimizes the utilization of staked funds. This improves overall liquidity efficiency.

Liquidity Sharing

The unified pool allows for dynamic allocation of liquidity to various channels and bulk-token-transfers based on demand, optimizing the use of staked funds.

Theorem 3 (Liquidity Sharing Security):

The unified staking pool and dynamic allocation mechanism ensure secure and efficient sharing of liquidity across channels, preventing unauthorized access to staked funds.

Proof 4:

The security of liquidity sharing relies on the integrity of the smart contract managing the unified staking pool. The smart contract enforces access control mechanisms, ensuring that only authorized entities can allocate or transfer funds. Dynamic allocation is performed based on predefined rules and demand metrics, preventing arbitrary or malicious allocations.

Therefore, liquidity sharing through the unified staking pool is secure and efficient.

Cross-Channel Guarantees and Security

Recursive ZKPs for Aggregation:

We aggregate proofs from multiple transactions and channels into a single proof using recursive ZKPs. This ensures efficient and scalable verification, expressed as:

\text{RecursiveZKP} = \bigoplus_{i=1}^{n} \text{ZKP}(\text{Transaction}_i)

where (\bigoplus)

denotes the recursive aggregation operation.

Theorem 4 (Recursive ZKP Security):

Recursive ZKPs provide secure and efficient aggregation of proofs from multiple transactions and channels, ensuring the integrity of cross-channel verifications.

Proof 5:

The security of recursive ZKPs stems from the properties of the underlying ZKP system. Recursive composition preserves the soundness and completeness of individual proofs, ensuring that the aggregated proof is valid if and only if all constituent proofs are valid. Moreover, recursive aggregation reduces the verification complexity, as the verifier only needs to check a single aggregated proof instead of verifying each proof independently. Therefore, recursive ZKPs enable secure and efficient cross-channel verification.

Smart Contract Hierarchies

We implement a hierarchy of smart contracts to manage the allocation and reallocation of funds, ensuring secure and efficient cross-channel transactions.

Definition 5 (Smart Contract Hierarchy):

A smart contract hierarchy is a layered structure of interconnected smart contracts that manage the allocation and reallocation of funds across channels. Higher-level contracts control the overall liquidity distribution, while lower-level contracts handle channel-specific operations.

Theorem 5 (Smart Contract Hierarchy Security):

The smart contract hierarchy ensures secure and efficient management of cross-channel transactions, preventing unauthorized access and enforcing consistent state transitions.

Proof 6:

The security of the smart contract hierarchy relies on the correctness and integrity of the individual smart contracts. Each contract is carefully designed and audited to ensure that it performs its intended functions and maintains a consistent state. The hierarchical structure allows for modular control and error containment, as issues in one contract can be isolated and resolved without affecting the entire system. Access control mechanisms and state transition rules are enforced at each level, preventing unauthorized modifications. Therefore, the smart contract hierarchy provides a secure and efficient framework for managing cross-channel transactions.

O(1) State Growth Mechanism

To achieve O(1) state growth, we leverage Merkle proofs to manage state updates efficiently. Instead of recording each transaction directly on-chain, we record the Merkle root of bulk-token-transfers. This approach ensures that the state growth per block is constant, regardless of the number of transactions.

Process:

When a set of transactions occurs, they are organized into a Merkle tree. The Merkle root of this tree is then recorded on-chain. Subsequent state updates only need to include the new Merkle root, significantly reducing the data stored on-chain.

Mathematical Formalism:

Let (T_1, T_2, \ldots, T_n)

be a set of transactions. These transactions are used to construct a Merkle tree (M)

, with the Merkle root (R)

. The state update process can be described as:

\text{State}*{\text{new}} = \text{State}*{\text{old}} \cup {R}

where (R)

is the Merkle root of the current set of transactions. This ensures that the state growth is limited to the size of the Merkle root, typically 32 bytes, achieving O(1) state growth.

Example Scenario:

Consider a block containing 1000 transactions. Instead of recording each transaction on-chain, we construct a Merkle tree with these transactions and record only the Merkle root. This reduces the on-chain data from 1000 transactions to a single Merkle root, illustrating the efficiency of this approach.

Implementation Details

Implementing Merkle proofs in a blockchain system involves integrating them into the transaction processing pipeline and state management infrastructure.

Integration:

The Merkle tree construction and root calculation are performed off-chain. Once the Merkle root is computed, it is included in the block header, ensuring minimal on-chain data.

Algorithms and Data Structures:

Algorithm 2 Merkle Root Calculation

Require: Transactions T1, T2, . . . , Tn Ensure: Merkle root R 1: Initialize leaf nodes with hashes of transactions 2: while more than one node in the tree do 3: for each pair of nodes do 4: Compute parent node as hash of the two child nodes 5: end for 6: end while 7: return Root node

Elimination of Continuous Online Presence

In traditional blockchain systems, participants often need to remain continuously online to ensure their transactions are confirmed and to participate in consensus mechanisms. This requirement can be burdensome and impractical for many users, leading to inefficiencies and increased operational costs.

Off-Chain Payment Channels

Off-chain payment channels enable participants to conduct transactions without needing to record each one on the blockchain. These channels allow for instant transaction finality and reduce the need for continuous online presence.

Mechanisms:

Participants open a payment channel by locking funds on-chain. They can then transact off-chain by exchanging signed messages that update the channel balance. The final state is recorded on-chain when the channel is closed.

Instant Finality with Zero-Knowledge Proofs

Instant finality ensures that transactions are immediately confirmed and final once the conditions are met. Zero-knowledge proofs (ZKPs) play a crucial role in achieving this by providing a way to verify transactions without revealing sensitive information.

Example Use Cases:

- Conditional payments where funds are transferred only if certain conditions are met.
- Micropayments and streaming payments where instant confirmation is essential.

Design and Implementation

Our system architecture supports offline participation through the use of off-chain payment channels and ZKPs.

Architectural Design:

- Off-chain transactions are facilitated by payment channels.
- ZKPs are used to verify the conditions for transaction finality.

Technical Implementation:

Algorithm 3 Off-Chain Transaction with ZKP

Require: Initial channel state S0, transactions T1, T2, . . . , Tn Ensure: Final channel state Sn 1: for each transaction Ti do 2: Verify ZKP for Ti 3: if ZKP is valid then 4: Update channel state Si 5: end if 6: end for 7: Record final state on-chain

Security and Reliability

Ensuring the security and reliability of offline participants is crucial. Our approach includes several mechanisms to protect users and ensure transaction integrity.

Security Measures:

- Secure channel opening and closing procedures.
- ZKPs to verify transaction conditions.

Dispute Resolution:

- Disputes are resolved by submitting ZKPs to the blockchain, which verifies the validity of the disputed transaction.

Benefits and User Experience

Our approach offers several benefits to users, including reduced operational burdens and increased flexibility.

Advantages:

- Operational Efficiency:

Users do not need to remain online continuously.

- Flexibility:

Participants can transact at their convenience.

Capital Efficiency

The unified liquidity pool ensures efficient use of locked funds, as they can be dynamically allocated where needed most. This reduces the amount of idle capital and optimizes liquidity management.

Lemma 2 (Capital Efficiency Improvement):

The unified staking pool and dynamic allocation mechanism improve capital efficiency compared to traditional channel-specific staking approaches.

Proof 7:

In traditional channel-specific staking, funds are locked in individual channels and cannot be reused or reallocated, leading to idle capital and suboptimal liquidity utilization. The unified staking pool, on the other hand, aggregates funds from multiple participants and dynamically allocates them across channels based on demand. This approach minimizes idle capital and optimizes liquidity utilization, as funds can be efficiently moved to channels with higher demand. Therefore, the unified staking pool and dynamic allocation mechanism improve capital efficiency compared to traditional approaches.

Instant Finality

Transactions are confirmed instantly off-chain, similar to the Lightning Network, making the system suitable for real-time payments. Zero-knowledge proofs ensure that transactions are final and secure without the need for trust.

Lemma 3 (Instant Finality Advantage):

Instant finality provided by off-chain payment channels and ZKP-based conditional payments enables real-time transactions and reduces confirmation latency compared to on-chain settlements.

Proof 8:

On-chain transactions require multiple block confirmations to be considered final, introducing significant latency. In contrast, off-chain payment channels enable instant confirmation of transactions between channel participants. Conditional payments verified by ZKPs further enhance the security and finality of these transactions. As a result, the instant finality mechanism reduces confirmation latency and enables real-time payments, providing a significant advantage over on-chain settlements.

Complexity for Operators

Smart contracts handle dynamic allocation and verification, reducing the operational burden on operators. The use of recursive ZKPs simplifies the verification process, making it easier to manage multiple channels.

Lemma 4 (Operator Complexity Reduction):

Smart contracts and recursive ZKPs reduce the operational complexity for channel operators compared to manual liquidity management and individual proof verification.

Proof 9:

In a manual liquidity management setup, operators need to continuously monitor channel balances and manually allocate funds, which is time-consuming and error-prone. Similarly, verifying individual proofs for each transaction across multiple channels is computationally intensive. By employing smart contracts for dynamic allocation and recursive ZKPs for aggregated verification, the operational burden on operators is significantly reduced. Smart contracts automate liquidity management based on predefined rules, while recursive ZKPs simplify the verification process by aggregating proofs from multiple transactions and channels. This reduction in complexity allows operators to manage more channels efficiently and focus on other aspects of the system.

Operator Risk

By sharing liquidity across multiple channels and dynamically allocating funds, the risk of liquidity shortages and insolvency is minimized. Automated safeguards provided by smart contracts protect operators.

Lemma 5 (Operator Risk Mitigation):

The unified staking pool and dynamic allocation mechanism mitigate the risk of liquidity shortages and insolvency for channel operators compared to channel-specific staking.

Proof 10:

In a channel-specific staking setup, operators face the risk of liquidity shortages if the locked funds in a particular channel are insufficient to handle incoming transactions. This can lead to delays, failed transactions, and potential insolvency. The unified staking pool mitigates this risk by aggregating funds from multiple participants and dynamically allocating them across channels based on demand. If one channel experiences a shortage, funds can be quickly reallocated from other channels to meet the demand. Moreover, smart contracts implementing the unified staking pool can enforce automated safeguards, such as minimum balance thresholds and emergency liquidity provisions, further protecting operators from liquidity risks.

Challenge Periods and Dispute Resolution

Aggregated proofs and instant verification reduce the need for lengthy challenge periods and simplify dispute resolution. Recursive ZKPs ensure that the state across interconnected channels is consistently maintained and verifiable.

Lemma 6 (Simplified Dispute Resolution):

Aggregated proofs and instant verification using recursive ZKPs simplify dispute resolution and reduce the reliance on lengthy challenge periods compared to traditional cross-channel dispute mechanisms.

Proof 11:

In traditional cross-channel dispute resolution, each

channel maintains its own state, and disputes are resolved through a lengthy challenge-response process. This involves providing evidence for individual transactions and waiting for a challenge period to expire before the dispute is settled. With aggregated proofs and instant verification using recursive ZKPs, the state across interconnected channels is consistently maintained and can be efficiently verified. If a dispute arises, the aggregated proof can be quickly checked to determine the validity of the disputed transaction, without the need for a lengthy challenge period. This simplified dispute resolution process reduces the time and complexity associated with resolving cross-channel disputes.

Dependence on ZKPs

While reliance on ZKPs remains, their use in aggregation and instant verification can optimize computational efficiency. Developing scalable infrastructure for ZKP generation and verification can mitigate the overhead.

Lemma 7 (ZKP Efficiency Optimization):

The use of recursive ZKPs for aggregation and instant verification optimizes the computational efficiency of ZKP-based systems compared to individual proof generation and verification.

Proof 12:

Generating and verifying ZKPs for each individual transaction can be computationally expensive, especially as the number of transactions and channels grows. By employing recursive ZKPs, multiple proofs can be aggregated into a single proof, reducing the overall proof size and verification complexity. Instead of verifying each proof separately, the verifier only needs to check the aggregated proof, which is more efficient. Moreover, instant verification using ZKPs eliminates the need for lengthy waiting periods, further optimizing the computational overhead. While the reliance on ZKPs remains, the use of recursive ZKPs and instant verification helps mitigate the computational burden associated with ZKP-based systems.

Limited Scope of Statelessness

The unified liquidity pool and dynamic allocation extend the benefits of statelessness to more complex transactions and applications. The approach supports a wider range of stateless applications beyond simple payments.

Lemma 8 (Extended Statelessness Scope):

The unified liquidity pool and dynamic allocation mechanism extend the scope of statelessness to support more complex transactions and applications compared to traditional stateless payment channels.

Proof 13:

Traditional stateless payment channels are limited to simple payment transactions between two parties, where the channel state is not maintained. The unified liquidity pool and dynamic allocation mechanism introduced in this paper extend the scope of statelessness to support more complex transactions and applications. By aggregating funds from multiple participants and dynamically allocating them across channels, the system enables stateless execution of cross-channel transactions and multi-party interactions. The use of recursive ZKPs and smart contract hierarchies further enhances the capabilities of stateless operations, allowing for efficient verification and management of complex state transitions. As a result, the proposed approach expands the range of stateless applications beyond simple payments, enabling the development of more sophisticated stateless protocols and services.

Forced Inclusion Mechanism

With instant finality, the need for forced inclusion mechanisms is reduced, as transactions are confirmed off-chain instantly. Zero-knowledge proofs ensure that once conditions are met, transactions are final and included without the risk of censorship.

Lemma 9 (Reduced Reliance on Forced Inclusion):

Instant finality provided by off-chain payment channels and ZKP-based conditional payments reduces the reliance on forced inclusion mechanisms compared to on-chain transaction inclusion.

Proof 14:

In on-chain transaction inclusion, there is a risk of censorship or delayed inclusion, as miners or validators may prioritize certain transactions over others. Forced inclusion mechanisms, such as transaction pinning or priority fees, are often employed to ensure that transactions are included in a timely manner. However, with instant finality achieved through off-chain payment channels and ZKP-based conditional payments, transactions are confirmed instantly without relying on the underlying blockchain. Once the conditions specified in the ZKP are met, the transaction is considered final and irreversible, eliminating the need for forced inclusion. This reduces the reliance on additional inclusion mechanisms and provides a more censorship-resistant and predictable transaction confirmation process.

LA FIN