Private Scalable Fully Decentralized Payment Network (Without Compromise)

Brandon “Cryptskii” Ramsay

Abstract

This research paper presents a novel decentralized payment network model that leverages zero-knowledge proofs (ZKPs) to ensure transaction validity and balance consistency without relying on validators or consensus mechanisms. The network features a fixed token supply, airdropped to participants at inception, eliminating the need for mining and associated costs. The core design focuses on direct mutual verification between sender and receiver, with an extensive exploration of the underlying mathematical foundations, formal proofs, algorithms, and data structures underpinning this system.

The proposed payment network aims to address key challenges faced by existing decentralized payment systems, such as high transaction costs, scalability limitations, and privacy concerns. By employing ZKPs and a unilateral payment channel architecture, the network enables efficient, secure, and privacy-preserving transactions without the need for intermediaries or complex consensus protocols. The paper provides a comprehensive analysis of the system’s security, privacy, and scalability properties, along with detailed comparisons to alternative approaches. The underlying mathematical framework and formal proofs are rigorously defined, ensuring the robustness and correctness of the proposed model.

Introduction

Decentralized payment systems have garnered significant attention due to their potential to provide secure, transparent, and efficient financial transactions without intermediaries. However, existing solutions often face challenges related to high transaction costs, scalability limitations, and privacy concerns. This research introduces a novel decentralized payment network that leverages zero-knowledge proofs (ZKPs) and unilateral payment channels to address these issues.

The proposed network architecture is designed to address specific challenges faced by existing decentralized payment systems:

1. The absence of mining and associated costs solves the issue of high transaction fees and energy consumption in traditional proof-of-work-based systems.
2. The elimination of validators and consensus mechanisms tackles the scalability limitations and potential centralization risks in proof-of-stake and delegated systems.
3. The use of storage partitioning and off-chain payment channels addresses the scalability and privacy concerns associated with storing all transactions on-chain.

By distributing a fixed token supply to participants at the network’s inception, the system eliminates the need for mining and its associated costs. The network focuses on enabling direct mutual verification of transactions between senders and receivers, ensuring the validity of transactions and the consistency of account balances without relying on validators or consensus mechanisms. By leveraging zk-SNARKs, the network allows for direct proof of validity between sender and receiver, as the succinct zero-knowledge proofs inherently prove the correctness of transactions.

To enhance efficiency and scalability, the network uses a multi-tier Merkle tree system with Merkle proofs, ensuring that only a constant succinct size (O(1))

of data is submitted to the blockchain. This design minimizes on-chain storage requirements and ensures data availability.

At the core of this novel payment network lies a comprehensive mathematical framework that leverages zero-knowledge proofs, particularly zk-SNARKs, to validate transactions and generate wallet state proofs. These proofs enable efficient verification of transaction validity and balance updates while preserving user privacy.

The network’s architecture is composed of several key components, including unilateral payment channels, hierarchical smart contracts, and partitioned storage nodes. These components work together to enable scalable, secure, and privacy-preserving transactions, while minimizing on-chain storage requirements and ensuring data availability.

To ensure the robustness and correctness of the proposed model, the paper presents formal algorithms, theorems, and proofs for crucial aspects of the system, such as the Balance Consistency Theorem and the dispute resolution mechanism. These mathematical formalisms provide a solid foundation for the security and reliability of the payment network.

Furthermore, the paper includes an in-depth analysis of the network’s security, privacy, and scalability properties, highlighting its advantages over alternative approaches, such as traditional blockchain-based payment systems and centralized payment networks. The analysis also acknowledges potential limitations and challenges, such as the complexity of zk-SNARK implementations and the need for ongoing optimizations.

The main contributions of this research can be summarized as follows:

1. A comprehensive mathematical framework for ensuring transaction validity and balance consistency using zero-knowledge proofs, particularly zk-SNARKs.
2. A detailed description of the network's architecture, including unilateral payment channels, hierarchical smart contracts, and partitioned storage nodes.
3. Formal algorithms, theorems, and proofs for key components of the system, such as the Balance Consistency Theorem, zk-SNARK proof generation, smart contract verification, and dispute resolution.
4. An in-depth analysis of the network's security, privacy, and scalability properties, along with detailed comparisons to alternative approaches.
5. An exploration of promising use cases that leverage the enhanced privacy features of the proposed system.

The proposed decentralized payment network presents a promising approach to enabling secure, private, and scalable transactions in a decentralized setting, paving the way for more efficient and accessible financial services on the blockchain. The extensive mathematical formalism and rigorous analysis provided in this paper contribute to the growing body of research on decentralized payment systems and demonstrate the potential of zero-knowledge proofs in enhancing the security, privacy, and scalability of blockchain-based financial applications.

[

DALL·E_2024-06-05_20.49.33_-_A_high-level_overview_of_a_decentralized_payment_network_architecture._The_diagram_includes_Off-Cha

1117×970 218 KB

](https://ethresear.ch/uploads/default/original/3X/d/6/d6d467a101f5b9aae81288f1c0d30234082f2bcd.jpeg)

Background

This section introduces the key concepts and technologies used in the proposed decentralized payment network, providing a solid foundation for understanding the system's design and functionality.

Zero-Knowledge Proofs (ZKPs)

Zero-knowledge proofs (ZKPs) are cryptographic protocols that enable one party (the prover) to prove to another party (the verifier) that a statement is true without revealing any information beyond the validity of the statement itself. ZKPs have numerous applications in blockchain technology, particularly in privacy-preserving transactions and scalable off-chain solutions.

One prominent type of ZKP is zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge). zk-SNARKs enable the generation of succinct proofs that can be verified quickly, making them well-suited for blockchain applications where proofs need to be stored on-chain and verified by multiple parties.

Definition

: A zero-knowledge proof for a statement S

is a protocol between a prover P

and a verifier V

such that:

- Completeness

: If S

is true, V

will be convinced by P

with high probability.

- Soundness

: If S

is false, V

will not be convinced by P

except with negligible probability.

- Zero-Knowledge

: If S

is true, V

learns nothing beyond the fact that S

is true.

In the proposed decentralized payment network, zk-SNARKs are employed to prove the validity of transactions and generate wallet state proofs, ensuring the privacy and security of user balances while enabling efficient verification.

System Architecture

The proposed decentralized payment network consists of several key components: off-chain payment channels, hierarchical smart contracts, and partitioned storage nodes. This section provides a detailed description of each component and their interactions within the overall system.

Certainly! Here is the document reformatted in Markdown with low-level LaTeX math included:

Unilateral Payment Channels

Payment channels are a key component of scalable blockchain solutions, enabling off-chain transactions between parties without the need to record every transaction on the main blockchain. In the proposed network, each user has a unilateral payment channel associated with their wallet contract, which holds their tokens off-chain. This design choice simplifies channel management and enables cross-partition transactions.

Definition:

A unilateral payment channel between a user U

and their wallet contract W

is a tuple (B, T_1, T_2, \ldots, T_n)

, where:

- B

is the initial balance of U

in the payment channel.

- T_1, T_2, \ldots, T_n

are the transactions executed within the payment channel.

The final state of the payment channel is determined by the cumulative effect of all transactions T_1, T_2, \ldots, T_n

on the initial balance B

.

To set up a unilateral payment channel, a user creates a wallet contract on the blockchain and transfers the desired amount of tokens to the contract. The wallet contract manages the user's off-chain balance and state updates through zk-SNARK proofs. When a user wants to transfer tokens to another user, they generate a zk-SNARK proof that verifies the validity of the transaction and includes the necessary metadata for the recipient to generate the next transaction proof. This design enables instant transaction finality and eliminates the need for on-chain confirmation.

Example 1: Unilateral Payment Channel Setup and Transactions

Suppose Alice wants to set up a unilateral payment channel with an initial balance of 100 tokens. She creates a wallet contract W_A

on the blockchain and transfers 100 tokens to it. The wallet contract initializes Alice’s off-chain balance to 100 tokens.

Later, Alice decides to send 30 tokens to Bob. She generates a zk-SNARK proof \pi_1

that verifies the validity of the transaction, including the availability of sufficient funds and the correctness of the updated balances. Upon generating the proof \pi_1

, the wallet contract immediately locks 30 tokens, reducing Alice’s available balance to 70 tokens. Alice sends the transaction details and the proof \pi_1

to Bob.

Bob verifies the proof \pi_1

to ensure the transaction’s validity. If the proof is valid, Alice and Bob update their local off-chain balances accordingly. Alice’s balance remains 70 tokens, while Bob’s balance increases by 30 tokens. Both parties sign the proof \pi_1

to authorize the future rebalancing of their respective payment channels.

If Bob does not accept the proof within a specified timeout period, the smart contract automatically releases the locked funds back to Alice’s available balance, ensuring no funds are indefinitely locked.

This example demonstrates how unilateral payment channels enable secure, off-chain transactions between users while preserving privacy and scalability.

Off-Chain Payment Channel Operations

As introduced in Section 2, off-chain payment channels form the foundation of the proposed network’s scalability. Each user has a unilateral payment channel associated with their wallet contract, which holds their tokens off-chain. The channel setup and transaction process can be formally described as follows:

Channel Setup

\begin{algorithm}[H] \caption{Unilateral Payment Channel Setup} \begin{algorithmic}[1] \Procedure{SetupChannel}{U, W, B} \State U creates a wallet contract W on the blockchain \State U transfers B tokens to W \State W initializes the off-chain balance of U to B \State \textbf{return} W \EndProcedure \end{algorithmic} \end{algorithm}

Here, U

represents the user, W

is the wallet contract, and B

is the initial balance in the payment channel.

Off-Chain Transactions

\begin{algorithm}[H] \caption{Off-Chain Transaction} \begin{algorithmic}[1] \Procedure{OffchainTransaction}{S, R, T} \State S generates a zk-SNARK proof π for the transaction T \State S's wallet contract locks the transaction amount \State S sends (T, π) to R \State R verifies π to ensure the validity of T \If{π is valid} \State S updates their local off-chain balance \State R updates their local off-chain balance \State S and R sign π to authorize rebalancing \Else \State S's wallet contract releases the locked amount after timeout \EndIf \State \textbf{return} (T, π) \EndProcedure \end{algorithmic} \end{algorithm}

Here, S

represents the sender, R

is the receiver, and T

is the transaction. The zk-SNARK proof \pi

verifies the validity of the transaction, including the availability of sufficient funds and the correctness of the updated balances. If the proof is valid, both parties update their local off-chain balances and sign the proof to authorize the future rebalancing of their payment channels. If the proof is not accepted within a specified timeout period, the smart contract automatically releases the locked funds back to the sender’s available balance.

Hierarchical Smart Contracts

The hierarchical smart contract structure is a key component of the proposed network, enabling efficient rebalancing of payment channels and management of cross-partition transactions. The structure consists of three layers: root contracts, intermediate contracts, and wallet contracts.

- Root Contracts:

Responsibilities:

- Serve as the entry point for users and maintain a mapping of intermediate contracts.
- Aggregate Merkle roots from intermediate contracts and submit a single, final aggregated Merkle root.
- Submit the final Merkle root to the blockchain at regular intervals, ensuring the global state is verifiable on-chain with minimal frequency and cost being a constant size O(1)

.

- Serve as the entry point for users and maintain a mapping of intermediate contracts.
- Aggregate Merkle roots from intermediate contracts and submit a single, final aggregated Merkle root.
- Submit the final Merkle root to the blockchain at regular intervals, ensuring the global state is verifiable on-chain with minimal frequency and cost being a constant size O(1)

.

- Intermediate Contracts:

Responsibilities:

- Manage liquidity pools for specific transaction types or user groups.
- Maintain a mapping of wallet contracts and are responsible for rebalancing payment channels based on the transaction proofs submitted by users.
- Collect Merkle roots from wallet contracts and aggregate them into a single Merkle tree within their partition.
- Periodically submit the aggregated Merkle root to the root contract.
- Ensure the state within their partition is verifiable on-chain with minimal frequency and cost.

- Manage liquidity pools for specific transaction types or user groups.
- Maintain a mapping of wallet contracts and are responsible for rebalancing payment channels based on the transaction proofs submitted by users.
- Collect Merkle roots from wallet contracts and aggregate them into a single Merkle tree within their partition.
- Periodically submit the aggregated Merkle root to the root contract.
- Ensure the state within their partition is verifiable on-chain with minimal frequency and cost.

- Wallet Contracts:

Responsibilities:

- Represent individual user payment channels and hold the users' off-chain balances.
- Generate zk-SNARK proofs for their state and submit these proofs to the storage nodes.
- Store proofs to the storage nodes for data availability.

- Represent individual user payment channels and hold the users' off-chain balances.
- Generate zk-SNARK proofs for their state and submit these proofs to the storage nodes.
- Store proofs to the storage nodes for data availability.

The hierarchical structure allows for efficient liquidity management and reduced on-chain transaction costs, as rebalancing operations are performed at the intermediate level, and the root contracts only need to process periodic updates.

Example 3: Hierarchical Smart Contract Interaction

Continuing with the previous examples, suppose Alice, Bob, Carol, and David belong to the same user group managed by an intermediate contract IC_1

. The intermediate contract IC_1

is mapped to a root contract RC

.

- When Alice sends 30 tokens to Bob (transaction T_1

), she generates a zk-SNARK proof \pi_1

. Upon generating the proof \pi_1

, Alice's wallet contract immediately locks 30 tokens, reducing her available balance accordingly. The transaction proof \pi_1

is then submitted to the intermediate contract IC_1

. The intermediate contract verifies the proof and updates the balances of Alice's and Bob's wallet contracts accordingly.

- Similarly, when Alice receives 50 tokens from Carol (transaction T_2

), Carol generates a zk-SNARK proof \pi_2

. Upon generating the proof \pi_2

, Carol's wallet contract immediately locks 50 tokens, reducing her available balance. The transaction proof \pi_2

is then submitted to IC_1

, which verifies the proof and updates the balances of Alice's and Carol's wallet contracts.

- Periodically, the intermediate contract IC_1

submits a summary of the balance updates to the root contract RC

, which maintains a global view of the network's state by submitting a single aggregated Merkle root to the blockchain.

This hierarchical structure, with the immediate balance locking mechanism, ensures that all transactions are secure and funds are not double spent, even if there are delays in transaction acceptance or verification.

Storage Nodes and Blockchain Interaction

To ensure data availability and scalability, the proposed network employs storage nodes that store the off-chain transaction history and wallet state proofs. Each storage node maintains a copy of the entire off-chain data, ensuring redundancy and decentralization.

Storage Node Operations:

- Storing Proofs:

Storage nodes store zk-SNARK proofs for individual wallet states. Each wallet maintains its own Merkle tree that includes these proofs.

- Aggregating Data:

At regular intervals, storage nodes aggregate the off-chain data into a single Merkle root, representing the state of all payment channels they manage. This Merkle root is then submitted to the intermediate contracts.

def store_proof(proof, user, wallet): # Store the proof for user and wallet contract # Update the local Merkle tree with the proof

def submit_merkle_root(): # Generate the Merkle root for all stored proofs # Submit the Merkle root to the intermediate contract

The blockchain acts as a secure and immutable ledger, storing the Merkle roots submitted by the root contract. This allows for efficient

verification of the network's global state, as any discrepancies between the off-chain data and the on-chain Merkle roots can be easily detected and resolved.

This hierarchical structure enables efficient verification of individual payment channels and the entire network state without storing the full transaction history on-chain. By leveraging the security and immutability of the blockchain while keeping the majority of the data off-chain, the proposed network achieves a balance between scalability, data availability, and security.

Example 4: Storage Node Operation and Blockchain Interaction

Following the previous examples, suppose storage node SN_1

is responsible for storing the transaction proofs and wallet state proofs for Alice, Bob, Carol, and David.

- When Alice generates a wallet state proof \pi_s

after transactions T_1, T_2,

and T_3

, she submits the proof to the storage node SN_1

. The storage node stores the proof and updates its local Merkle tree with the new proof.

- Similarly, when Bob, Carol, and David generate their wallet state proofs, they submit them to SN_1

, which stores the proofs and updates its local Merkle tree accordingly.

- At the end of each epoch, SN_1

generates a Merkle root R

that represents the state of all payment channels it manages. The storage node then submits the Merkle root R

to the intermediate contract, providing a compact and tamper-evident snapshot of the network's state.

- The intermediate contract aggregates the Merkle roots from all storage nodes within its partition and submits a single final Merkle root to the root contract.
- The root contract aggregates the Merkle roots from all intermediate contracts and submits a single final Merkle root to the blockchain.
- The blockchain stores the submitted Merkle root, allowing for efficient verification of the network's global state. If any discrepancies arise between the off-chain data and the on-chain Merkle roots, they can be easily detected and resolved using the dispute resolution mechanism described in the following section.

This hierarchical structure, combined with immediate balance locking and zk-SNARK proofs, ensures secure, efficient, and scalable off-chain transactions, maintaining the integrity and security of the overall network.

Transaction Validity and Balance Consistency

To ensure the validity of transactions and the consistency of account balances, the proposed payment network employs a combination of zero-knowledge proofs and formal mathematical proofs. This section presents the core theorems and algorithms that underpin the system's security and correctness. (as stated in the abstract, we have a fixed supply released in full at genesis)

Transaction Validity

Each transaction in the proposed network is accompanied by a zk-SNARK proof that verifies the following conditions:

- The sender has sufficient balance to cover the transaction amount.
- The sender's updated balance is correctly computed.
- The receiver's updated balance is correctly computed.

Let T_i

be a transaction in which sender S

transfers \Delta_i

tokens to receiver R

. The accompanying zk-SNARK proof \pi_i

ensures the following conditions:

\begin{align*} B_S &\geq \Delta_i \ B'_S &= B_S - \Delta_i \ B'_R &= B_R + \Delta_i \end{align*}

where B_S

and B_R

are the initial balances of S

and R

, respectively, and B'_S

and B'_R

are the updated balances after the transaction.

Balance Consistency

To prove the consistency of account balances in the presence of valid transactions, we present the following theorem:

Theorem (Balance Consistency)

: Given a series of valid transactions T_1, T_2, \ldots, T_n

between two parties S

and R

, the final balances B'_S

and B'_R

satisfy:

B'_S + B'_R = B_S + B_R

where B_S

and B_R

are the initial balances of S

and R

, respectively.

Proof

: We prove the theorem by induction on the number of transactions n

.

Base case

: For n = 1

, we have a single transaction T_1

with amount \Delta_1

. The updated balances after the transaction are:

\begin{align*} B'_S &= B_S - \Delta_1 \ B'_R &= B_R + \Delta_1 \end{align*}

Adding the above equations yields:

B'_S + B'_R = B_S + B_R

Inductive step

: Assume the theorem holds for n = k

transactions. We prove that it also holds for n = k + 1

transactions.

Let B^{(k)}_S

and B^{(k)}_R

be the balances after the first k

transactions. By the induction hypothesis, we have:

B^{(k)}_S + B^{(k)}_R = B_S + B_R

Now, consider the (k+1)

-th transaction T_{k+1}

with amount \Delta_{k+1}

. The updated balances after this transaction are:

\begin{align*} B^{(k+1)}S &= B^{(k)}_S - \Delta{k+1} \ B^{(k+1)}R &= B^{(k)}_R + \Delta{k+1} \end{align*}

Adding the above equations and substituting the induction hypothesis yields:

\begin{align*} B^{(k+1)}S + B^{(k+1)}_R &= (B^{(k)}_S - \Delta{k+1}) + (B^{(k)}R + \Delta{k+1}) \ &= B^{(k)}_S + B^{(k)}_R \ &= B_S + B_R \end{align*}

Therefore, the theorem holds for n = k + 1

transactions.

By the principle of mathematical induction, the theorem holds for any number of valid transactions

n \geq 1

. \blacksquare

The Balance Consistency theorem ensures that the total balance of the system remains constant throughout a series of valid transactions, providing a fundamental property for the security and correctness of the proposed payment network.

Fraud Prevention Mechanisms

The proposed decentralized payment network integrates multiple layers of fraud prevention mechanisms through its hierarchical smart contract system and the use of zk-SNARKs. These measures ensure the integrity and consistency of transaction states, inherently preventing the submission of outdated or fraudulent states. This section outlines how these mechanisms work in detail.

ZK-SNARK Proofs and State Updates

The network leverages zk-SNARKs to validate each transaction. The key elements include:

- Proof of Validity

: Each transaction within the network must be accompanied by a zk-SNARK proof. This proof verifies several critical aspects:

- The sender has sufficient balance to cover the transaction.
- The sender’s updated balance after the transaction is correctly computed.
- The receiver’s updated balance after the transaction is correctly computed.
- The sender has sufficient balance to cover the transaction.
- The sender’s updated balance after the transaction is correctly computed.
- The receiver’s updated balance after the transaction is correctly computed.
- Consistent State Management

: Each user’s wallet contract maintains a Merkle tree of state proofs. Each state update (i.e., each transaction) is validated through zk-SNARKs, ensuring it is consistent with the previously recorded state. This cryptographic validation prevents unauthorized or incorrect state changes.

Prevention of Old State Submission

The design of the proposed network inherently prevents the submission of outdated or fraudulent states through the following mechanisms:

- Proof Consistency

: Each zk-SNARK proof submitted for a state update must be consistent with the latest recorded state. Intermediate contracts aggregate data from wallet contracts, and root contracts submit these aggregated roots to the blockchain. Any attempt to submit an old state would be detected as it would not match the current aggregated Merkle root.

- On-Chain Verification

: The final aggregated Merkle root submitted by the root contract is stored on the blockchain, providing a tamper-evident record of the global state. During dispute resolution, the submitted state proofs are verified against this on-chain Merkle root to ensure only the most recent valid state is considered.

Mitigated Need for Watchtowers

Due to the robust fraud prevention mechanisms built into the proposed system, the traditional need for watchtowers entities that monitor the blockchain for malicious activities and act on behalf of users is significantly reduced. The hierarchical structure and the use of zk-SNARKs ensure that:

- Each state update is cryptographically verified, preventing unauthorized changes.
- The aggregated Merkle roots provide a consistent and tamper-evident record of the network's state.
- Dispute resolution is handled efficiently and fairly based on the most recent valid state proofs.

The comprehensive fraud prevention mechanisms of the proposed decentralized payment network ensure high levels of security and integrity without the need for external monitoring entities like watchtowers. The hierarchical smart contract system and zk-SNARKs work together to maintain consistent and verifiable transaction states, providing a secure and efficient framework for decentralized financial transactions.

Role of the DAO

While the built-in mechanisms provide robust security and minimize the need for watchtowers, there are scenarios where manual involvement might be necessary. To address these situations, a Decentralized Autonomous Organization (DAO) can be implemented to manage and oversee the network's operations. The DAO would play a crucial role in:

- Handling Exceptional Cases

: Situations that require manual intervention beyond the automated fraud prevention and dispute resolution mechanisms.

- Balancing Automation and Trust

: Ensuring the right mix of automated processes, cryptographic proofs, and trust mechanisms to maintain network integrity.

- Democratic Decision-Making

: Leveraging community governance to make decisions on critical issues, such as protocol upgrades, handling disputes that the automated system cannot resolve, and other governance matters.

DAO Functions

1. Manual Dispute Resolution

: For disputes that cannot be resolved through automated proofs, the DAO can step in to review and make a final decision based on community consensus.

1. Protocol Upgrades

: The DAO can propose and vote on protocol upgrades to enhance the system's functionality and security.

1. Network Oversight

: Providing ongoing oversight and making strategic decisions to ensure the network remains secure and efficient.

The combination of zk-SNARKs, hierarchical smart contracts, and the DAO creates a robust framework for fraud prevention and network governance. The minimized need for watchtowers is achieved through advanced cryptographic verification and efficient dispute resolution mechanisms. However, the DAO ensures that any issues requiring manual involvement are handled with a balance of automation, trust, rigorous mathematical verification, and democratic decision-making. This comprehensive approach provides a secure, scalable, and trustworthy decentralized payment network.

Dispute Resolution

In the event of a dispute between parties, the proposed network employs a dispute resolution mechanism based on the submitted zk-SNARK proofs and Merkle roots. The dispute resolution process can be formally described as follows:

\begin{algorithm}[H] \caption{Dispute Resolution} \begin{algorithmic}[1] \Procedure{ResolveDispute}{S, R, π_S, π_R} \State S submits their final state proof π_S \State R submits their final state proof π_R \State Verify π_S and π_R against the submitted Merkle roots \If{π_S is valid and π_R is invalid} \State Resolve the dispute in favor of S \ElsIf{π_R is valid and π_S is invalid} \State Resolve the dispute in favor of R \Else \State Resolve the dispute based on the most recent valid state proof \EndIf \EndProcedure \end{algorithmic} \end{algorithm}

Here, S

and R

represent the disputing parties, and \pi_S

and \pi_R

are their respective final state proofs. The dispute resolution mechanism verifies the submitted proofs against the Merkle roots stored on-chain and resolves the dispute based on the validity of the proofs. This ensures that the resolution is based on the most recent valid state of the payment channel, preventing fraud and maintaining the integrity of the system.

The dispute resolution process follows these steps:

1. Dispute Initiation:

Either party can initiate a dispute by submitting a dispute request to the relevant smart contract (e.g., the intermediate contract managing their user group).

1. Evidence Submission:

Both parties are required to submit their final state proofs (\pi_S \text{ and } \pi_R)

within a predefined timeframe (e.g., 24 hours). These proofs represent the latest state of their respective payment channels and include the relevant transaction history.

1. Proof Verification:

The dispute resolution mechanism verifies the submitted proofs against the Merkle roots stored on-chain. This verification process ensures that the proofs are valid and consistent with the global state of the network.

1. Resolution:

The dispute is resolved based on the validity of the submitted proofs: * If \pi_S

is valid and \pi_R

is invalid, the dispute is resolved in favor of party S

.

- If \pi_R

is valid and \pi_S

is invalid, the dispute is resolved in favor of party R

.

- If both proofs are valid, the dispute is resolved based on the most recent valid state proof, determined by the timestamp or sequence number associated with the proofs.
- If neither proof is valid or if one party fails to submit their proof within the required timeframe, the dispute can be escalated to a higher-level contract (e.g., the root contract) or a trusted third party for manual review and resolution.
- If \pi_S

is valid and \pi_R

is invalid, the dispute is resolved in favor of party S

.

1. If \pi_R

is valid and \pi_S

is invalid, the dispute is resolved in favor of party R

.

1. If both proofs are valid, the dispute is resolved based on the most recent valid state proof, determined by the timestamp or sequence number associated with the proofs.
2. If neither proof is valid or if one party fails to submit their proof within the required timeframe, the dispute can be escalated to a higher-level contract (e.g., the root contract) or a trusted third party for manual review and resolution.
3. Outcome Enforcement:

Once the dispute is resolved, the smart contracts automatically enforce the outcome by updating the balances of the involved parties according to the resolution decision. This may involve redistributing tokens between the parties' payment channels or applying penalties for fraudulent behavior.

To incentivize honest behavior and discourage frivolous disputes, the network can implement additional mechanisms:

- Dispute Bond:

Parties initiating a dispute may be required to post a bond (in the form of tokens) that is forfeited if their submitted proof is found to be invalid or if they fail to submit their proof within the required timeframe. This bond serves as a deterrent against malicious actors and ensures that disputing parties have a stake in the resolution process.

- Reputation System:

The network can maintain a reputation score for each user based on their history of successful transactions and dispute resolutions. Users with a high reputation score may be given preference in case of ambiguous disputes or may enjoy reduced dispute bond requirements. Conversely, users with a history of fraudulent behavior or frivolous disputes may face higher bond requirements or even temporary suspension from the network.

By combining cryptographic proofs, smart contract automation, and economic incentives, the proposed dispute resolution mechanism ensures that conflicts are resolved fairly and efficiently while maintaining the integrity of the payment network.

Example 5: Dispute Resolution

Suppose a dispute arises between Alice and Bob regarding the final state of their payment channel. Alice claims that her final balance is 100 tokens, while Bob claims that Alice's final balance is 80 tokens.

1. Dispute Initiation:

Alice initiates a dispute by submitting a dispute request to the intermediate contract IC_1

that manages their user group. She deposits the required dispute bond of 10 tokens.

1. Evidence Submission:

Alice and Bob submit their respective final state proofs, \pi_A

and \pi_B

, to the dispute resolution mechanism within the 24-hour timeframe. Alice's proof \pi_A

shows her balance as 100 tokens, while Bob's proof \pi_B

shows Alice's balance as 80 tokens.

1. Proof Verification:

The dispute resolution mechanism verifies the submitted proofs against the Merkle roots stored on-chain. It finds that Alice's proof \pi_A

is consistent with the on-chain state, while Bob's proof \pi_B

is invalid.

1. Resolution:

As Alice's proof \pi_A

is valid and Bob's proof \pi_B

is invalid, the dispute is resolved in favor of Alice. The resolution confirms that Alice's final balance is indeed 100 tokens.

1. Outcome Enforcement:

The intermediate contract IC_1

automatically updates the balances of Alice and Bob's payment channels according to the resolution decision. Alice's balance remains at 100 tokens, while Bob's balance is adjusted based on the discrepancy. Additionally, Bob's dispute bond of 10 tokens is forfeited and distributed as a reward to Alice for submitting a valid proof.

This example demonstrates how the dispute resolution mechanism ensures the integrity of the payment network by resolving conflicts based on the validity of the submitted zk-SNARK proofs and the Merkle roots stored on-chain, while also incentivizing honest behavior through the use of dispute bonds.

Comparison to Alternative Approaches

The proposed decentralized payment network offers several advantages over alternative approaches, such as traditional blockchain-based payment systems and centralized payment networks.

Compared to traditional blockchain-based payment systems, the proposed network provides higher scalability and privacy. The use of off-chain payment channels and zk-SNARKs enables faster and more private transactions, while the hierarchical smart contract structure and partitioned storage nodes enable more efficient processing and storage of transaction data.

Compared to centralized payment networks, the proposed system offers greater security, transparency, and censorship resistance. By leveraging the security and immutability of blockchain technology and the privacy-preserving properties of zk-SNARKs, the network can provide a more secure and transparent payment infrastructure that is resistant to censorship and control by central authorities.

Example 6: Comparison to Centralized Payment Networks

Suppose a centralized payment network relies on a single trusted entity to process transactions and manage user balances. While this approach may offer high transaction throughput, it also presents several risks and limitations:

1. Single point of failure

: If the central entity experiences technical issues or becomes compromised, the entire payment network may become unavailable or vulnerable to fraud.

1. Lack of transparency

: Users must trust the central entity to manage their funds honestly and securely, without the ability to independently verify the state of their balances or the validity of transactions.

1. Censorship risk

: The central entity may choose to block or reverse transactions based on their own criteria, censoring users or restricting access to the payment network.

In contrast, the proposed decentralized payment network addresses these issues through its use of blockchain technology, zk-SNARKs, and a decentralized architecture:

1. Decentralization

: The network is maintained by a distributed network of storage nodes and smart contracts, eliminating the single point of failure and ensuring the availability and resilience of the system.

1. Transparency and verifiability

: Users can independently verify the state of their balances and the validity of transactions using the zk-SNARK proofs and the Merkle roots stored on-chain, providing a high level of transparency and trust in the system.

1. Censorship resistance

: The decentralized nature of the network and the use of zk-SNARKs ensure that transactions cannot be easily censored or reversed by any single entity, preserving the freedom and autonomy of users.

This example highlights the significant advantages of the proposed decentralized payment network over centralized alternatives, providing a more secure, transparent, and censorship-resistant payment infrastructure for users.

Analysis

This section provides a comprehensive analysis of the security, privacy, and scalability properties of the proposed decentralized payment network, and compares it to alternative approaches.

We delve into the technical details of the zk-SNARK implementation, discuss potential challenges and trade-offs, explore additional privacy-enhancing techniques, and consider the governance aspects of the system.

Security Analysis

The security of the proposed network relies on the soundness and completeness of the zk-SNARK proofs, as well as the integrity of the hierarchical smart contract structure. We employ the state-of-the-art zk-SNARK construction proposed by Groth, which offers succinct proofs and efficient verification. The zk-SNARK scheme is built upon the q-Power Knowledge of Exponent (q-PKE) assumption and the q-Decisional Diffie-Hellman (q-DDH) assumption in bilinear groups.

Let \mathcal{G}_1

, \mathcal{G}_2

, and \mathcal{G}_T

be cyclic groups of prime order p

, and let e : \mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}_T

be a bilinear map. The q-PKE assumption states that for any polynomial-size adversary \mathcal{A}

, the following probability is negligible in the security parameter \lambda

:

\Pr\left[\begin{array}{c} g \xleftarrow{\$} \mathcal{G}*1, \quad \alpha, s \xleftarrow{\$} \mathbb{Z}_p, \quad g_2 \leftarrow g^{\alpha},\ (c_1, \ldots, c_q) \xleftarrow{\$} \mathbb{Z}_p^q, \quad t_i \leftarrow g^{c_i} \cdot g_2^{s \cdot c_i},\ (h, \hat{h}) \leftarrow \mathcal{A}(g, g_2, {t_i}*{i=1}^q) : \ h = g^s \wedge \hat{h} \neq \prod_{i=1}^q t_i^{c_i} \end{array} \right]

The q-DDH assumption states that for any polynomial-size adversary \mathcal{A}

, the following probability is negligible in the security parameter \lambda

:

\Pr\left[\begin{array}{c} g \xleftarrow{\$} \mathcal{G}*1, \quad \alpha, s, r \xleftarrow{\$} \mathbb{Z}_p, \quad g_2 \leftarrow g^{\alpha},\ (c_1, \ldots, c_q) \xleftarrow{\$} \mathbb{Z}_p^q, \quad t_i \leftarrow \begin{cases} g_2^{c_i}, & \text{if } b=0\ g_2^{c_i + s}, & \text{if } b=1 \end{cases},\ b \xleftarrow{\$} {0, 1}, \quad b' \leftarrow \mathcal{A}(g, g_2, {t_i}*{i=1}^q) : \ b = b' \end{array} \right]

Under these assumptions, the zk-SNARK construction ensures that the proofs are sound and complete, meaning that a prover cannot create a valid proof for a false statement (soundness) and that a valid proof always verifies successfully (completeness). Consequently, transactions are guaranteed to be valid, and balances are correctly updated, preventing double-spending and other fraudulent activities.

Attack Vector Prevention

The hierarchical smart contract structure, combined with the storage nodes, ensures that the network's global state remains consistent and verifiable, even in the presence of malicious actors. The smart contracts are implemented using the Solidity language and are formally verified using the Oyente and Zeus tools to ensure their correctness and security.

1. Collusion during the trusted setup ceremony:
2. Mitigated by the use of secure multi-party computation (MPC) protocols like ZEXE, ensuring a distributed setup process.
3. Involvement of diverse participants reduces the likelihood of successful collusion.
4. Collusion among users:
5. Prevented by the use of unforgeable and computationally binding zk-SNARK proofs (PLONK), making it infeasible for users to create valid proofs for fraudulent transactions.

6. Smart contracts verify proofs before executing transactions, ensuring only legitimate transactions are processed.
7. Collusion among storage nodes:
8. Mitigated by the distributed storage architecture with multiple nodes maintaining data copies, making it difficult for nodes to collude and provide false data without detection.
9. The use of Merkle trees and hash-based commitments allows smart contracts to verify data authenticity.
10. Smart contract vulnerabilities:
11. Addressed by formal verification tools, independent security audits, secure coding practices, access controls, and error handling mechanisms.
12. Upgradability and emergency stop mechanisms allow for deploying security patches and freezing contracts in case of severe vulnerabilities.
13. Privacy leaks:
14. Mitigated by the use of zk-SNARKs, ensuring transaction privacy.
15. Mixing techniques, anonymity networks, metadata obfuscation, and regular security assessments further enhance privacy protection.
16. Sybil attacks:
17. Inherently resistant due to the use of zk-SNARK proofs, smart contract verification, and the underlying blockchain's consensus mechanism.
18. The system's design, including proof validity and economic disincentives, makes it infeasible for attackers to create and manipulate multiple identities or payment channels.
19. The requirement of fees to set up payment channels and execute transactions further discourages Sybil attacks by making them financially costly for attackers.
20. Denial-of-Service (DoS) attacks:
21. Inherently mitigated by the computational cost of generating zk-SNARK proofs for each transaction, making it impractical for attackers to flood the network with a large number of transactions.
22. The decentralized architecture and the resilience of the underlying Ethereum blockchain provide additional protection against DoS attacks.

Scalability Analysis

The proposed decentralized payment network exhibits significant scalability potential due to its innovative use of zero-knowledge proofs (ZKPs), particularly zk-SNARKs, and the absence of traditional consensus mechanisms, which together enable instant finality. In this section, we will provide a detailed mathematical assessment and real-world benchmarks to validate the network's scalability potential.

Mathematical Formalism of TPS Scalability

Let us define the total time per transaction T_{tx}

as the sum of the time for proof generation T_{pg}

, network latency T_{nl}

, and the overhead for contract execution and state updates T_{oh}

. Given that we aim for high scalability, we will leverage parallel processing capabilities of nodes to handle multiple channels efficiently.

T_{tx} = T_{pg} + T_{nl} + T_{oh}

Assuming average values for these times, such as:

\begin{align*} T_{pg} & \approx 50 \text{ ms} \\ T_{nl} & \approx 50 \text{ ms} \\ T_{oh} & \approx 50 \text{ ms} \\ \end{align*}

The total time per transaction can be approximated as:

$$T_{tx} = 50 \text{ ms} + 50 \text{ ms} + 50 \text{ ms} = 150 \text{ ms}$$

Thus, the transactions per second (TPS) per node can be calculated as:

$$TPS_{\text{per node}} = \frac{1}{T_{tx}} = \frac{1}{150 \text{ ms} / 1000 \text{ ms/s}} \approx 6.67 \text{ TPS}$$

If we consider the network scaling linearly with the number of nodes, the total TPS for n nodes can be expressed as:

$$TPS_{\text{total}} = TPS_{\text{per node}} \times n = 6.67 \times n$$

For example, with 100 nodes, the network could achieve:

$$TPS_{\text{total}} = 6.67 \times 100 = 667 \text{ TPS}$$

TPS Within Channels

To further detail the TPS within individual payment channels, consider that each node can manage multiple channels. Let c denote the number of channels a node can handle, and let $T_{channel}$ represent the time to process a transaction within a channel.

$$T_{channel} = T_{pg} + T_{nl} + T_{oh} = 70 \text{ ms} \quad (\text{considering optimized conditions})$$

Thus, the TPS per channel:

$$TPS_{\text{per channel}} = \frac{1}{T_{channel}} = \frac{1}{70 \text{ ms} / 1000 \text{ ms/s}} \approx 14.29 \text{ TPS}$$

If each node can handle c channels, the total TPS per node considering channels would be:

$$TPS_{\text{node, channels}} = TPS_{\text{per channel}} \times c$$

Assuming a node can handle 10,000 channels:

$$TPS_{\text{node, channels}} = 14.29 \times 10{,}000 = 142{,}900 \text{ TPS}$$

For a network with n nodes, the total TPS could be:

$$TPS_{\text{total, channels}} = 142{,}900 \times n$$

Real-World Micro Benchmarks

To validate these theoretical calculations, we consider benchmarks from existing state channel implementations:

1. Celer Network

: Claims to handle up to 15,000 TPS per node.

1. Raiden Network

: Aims for several thousand TPS per node.

1. Lightning Network

: Estimates around 1,000 TPS per node in practical scenarios.

Given these benchmarks, our assumption of handling 10,000 channels per node with approximately 14.29 TPS per channel, resulting in 142,900 TPS per node, is ambitious but within a reasonable range for a highly optimized implementation leveraging zk-SNARKs and efficient contract management.

Potential Bottlenecks

Despite the promising scalability, several bottlenecks could impact performance:

1. Proof Generation and Verification

: While zk-SNARKs are efficient, the complexity of proofs can increase with advanced use cases.

1. Network Latency

: Global transactions can introduce delays that affect overall throughput.

1. Smart Contract Efficiency

: Inefficiencies in smart contracts can create processing delays.

1. Storage and Data Management

: Managing large numbers of channels and associated data could become challenging.

1. Node Reliability and Security

: Ensuring the reliability and security of each node is critical.

Addressing these bottlenecks through ongoing optimization and robust infrastructure will be crucial to achieving the theoretical TPS and ensuring the network's scalability and robustness.

Summary

The proposed decentralized payment network, leveraging zk-SNARKs and instant finality mechanisms, exhibits significant scalability potential. The mathematical formalism and real-world benchmarks indicate that the network can achieve high TPS by efficiently managing multiple channels per node. Continuous optimization and addressing potential bottlenecks will be essential to realizing this potential in practice.

Scalability Comparison with Existing Layer 2 Solutions

Key Features of the Proposed Network

1. Unilateral Payment Channels:

Enables high transaction throughput by facilitating off-chain transactions.

1. Zero-Knowledge Proofs (zk-SNARKs):

Ensures privacy and efficient transaction validity.

1. Instant Finality:

Transactions achieve instant finality without on-chain confirmations.

1. Partitioned Storage Nodes:

Manages off-chain data efficiently, reducing on-chain storage requirements.

Existing Layer 2 Solutions

State Channels (e.g., Lightning Network, Raiden Network):

- Scalability:

High throughput off-chain.

- Finality:

Near-instant off-chain finality.

- Challenges:

Requires channel monitoring and on-chain closures.

Plasma:

- Scalability:

High throughput with off-chain child chains.

- Finality:

Periodic on-chain commitments.

- Challenges:

Complex exit management and data availability.

Optimistic Rollups:

- Scalability:

Batches transactions off-chain.

- Finality:

Delayed due to fraud proof periods.

- Challenges:

Requires fraud proof monitoring.

ZK-Rollups:

- Scalability:

High throughput with off-chain transaction bundling.

- Finality:

Near-instant with zk-SNARKs.

- Challenges:

Complex proof generation.

Comparative Analysis

Throughput and Finality:

- The proposed network achieves high throughput and instant finality, comparable to state channels and ZK-Rollups and superior to Optimistic Rollups.

Efficiency and Cost:

- More cost-efficient by reducing on-chain transactions and eliminating mining, outperforming state channels and Plasma.

Data Management:

- Efficient off-chain data management through partitioned storage nodes, similar to Plasma and rollups.

Security and Privacy:

- Robust security and privacy with zk-SNARKs, comparable to ZK-Rollups and superior to solutions relying on fraud proofs.

Implementation Details

The proposed decentralized payment network is implemented using a combination of Rust, TypeScript, and Solidity. The core components, such as the zk-SNARK proof generation and verification, are written in Rust for performance and security reasons. The smart contracts are developed using Solidity, while the frontend and client-side interactions are built with TypeScript.

Specific zk-SNARK Construction

The system employs the PLONK zk-SNARK construction, which offers universality, updatability, and efficient proof generation and verification. PLONK allows for the creation of a universal and updateable structured reference string (SRS) that can be reused across multiple circuits or applications, reducing the complexity and coordination overhead associated with repeated trusted setups.

The PLONK circuits are designed using the arkworks library in Rust, which provides a set of tools and primitives for building zk-SNARK circuits compatible with the PLONK proving system. The library supports efficient constraint generation, witness computation, and proof generation, making it well-suited for the development of the decentralized payment network.

Challenges and Optimizations

One of the main challenges in implementing PLONK is the complexity of designing and optimizing the circuits to take advantage of the universal SRS. This requires a deep understanding of the PLONK framework and the techniques for constructing efficient and secure circuits.

To address this challenge, the implementation leverages various optimization techniques, such as:

1. Constraint system optimization: Minimizing the number of constraints in the circuit by using efficient gate design and layout techniques, such as gate aggregation and constant folding.
2. Witness compression: Reducing the size of the witness by using compact data representations and eliminating redundant information.
3. Proof aggregation: Batching multiple proofs together to reduce the overall proof size and verification cost.

These optimizations help to improve the performance and scalability of the PLONK-based zk-SNARK circuits, ensuring that the decentralized payment network can handle a high volume of transactions efficiently.

Integration with Ethereum

The smart contracts for the payment network are implemented using Solidity and deployed on the Ethereum blockchain. The contracts interact with the PLONK proofs generated by the Rust components through a verification contract that is optimized for the PLONK proving system.

The verification contract is designed to be gas-efficient and supports batch verification of PLONK proofs, allowing multiple proofs to be verified in a single transaction. This helps to reduce the overall cost and improve the throughput of the system.

Trusted Setup Ceremony

As PLONK requires a trusted setup for the universal SRS, a multi-party computation (MPC) ceremony is conducted to generate the SRS. The ceremony involves multiple participants from different organizations and backgrounds, ensuring that no single party has control over the entire setup process.

The MPC ceremony is organized and facilitated using secure computation frameworks, such as the ZEXE library, which provides a set of tools and protocols for conducting distributed key generation and parameter setup.

Concise Example: Private Asset Transfer

In a private asset transfer, Alice can transfer assets to Bob without revealing the transaction details to the public. Using PLONK, Alice generates a proof π that verifies the validity of the transfer and her sufficient balance without disclosing the transaction amount Δ.

Transfer T = (A, B, \pi)

where \pi

is the PLONK proof

1. \pi \leftarrow \text{generateProof}(A, B, \Delta)
2. Submit (A, B, \pi)

to the transfer contract

1. Contract verifies \pi
2. If \pi

is valid, execute transfer from A

to B

- Else, reject transfer
- If \pi

is valid, execute transfer from A

to B

1. Else, reject transfer

The proof \pi

ensures the following conditions:

- B_A \geq \Delta

(Alice’s balance is sufficient)

- B_A' = B_A - \Delta

(Alice’s updated balance)

- B_B' = B_B + \Delta

(Bob’s updated balance)

The smart contract executes the transfer only if the proof is valid, ensuring the transfer’s legitimacy without revealing the transaction details.

By leveraging PLONK, the proposed decentralized payment network achieves a balance between privacy, scalability, and ease of implementation. The universal and updateable nature of PLONK, combined with the optimization techniques and secure trusted setup ceremony, provides a solid foundation for building a privacy-focused and efficient payment system.

Use Cases for Privacy

Confidential Voting Systems

Confidential voting systems are a critical use case for enhanced privacy in decentralized networks. Voting systems must ensure that each vote is anonymous and secure while maintaining the integrity and transparency of the election process. By leveraging zk-SNARKs, our network can provide a solution that guarantees the confidentiality of votes while allowing for public verification of election results.

In a confidential voting system built on the proposed network, voters would cast their votes through private transactions, with zk-SNARKs proving that each vote is valid and belongs to an eligible voter without revealing the voter’s identity. The votes would be tallied through a series of confidential transactions, with the final result verifiable through the aggregated Merkle roots stored on-chain. This approach ensures that the voting process is transparent and auditable while preserving the privacy of individual voters.

Private Asset Transfers

Private asset transfers are another significant use case for enhanced privacy in a decentralized network. These transfers require confidentiality to protect the financial privacy of users, ensuring that transaction details remain private while the integrity of the transfer is verifiable.

With the proposed network, users can transfer assets through confidential payment channels, with zk-SNARKs proving the validity of the transactions without revealing the amounts or the identities of the parties involved. This feature is particularly valuable for businesses and individuals who wish to keep their financial transactions private, such as in the case of sensitive business deals, wealth management, or personal remittances.

Secure Health Records Management

Secure health records management is an essential use case for enhanced privacy, where sensitive health information must be kept confidential while ensuring that authorized parties can verify the records. Using zk-SNARKs, the proposed network can enable the secure storage and sharing of health records while maintaining patient privacy.

In this use case, health records would be stored off-chain, with zk-SNARKs proving the authenticity and integrity of the records without revealing their contents. Patients can grant access to their records to authorized parties, such as healthcare providers or insurance companies, through private transactions. The authorized parties can then verify the records’ authenticity using the zk-SNARK proofs, ensuring that the records have not been tampered with while preserving patient confidentiality.

Global Payment System

A global payment system is perhaps the most scalable and impactful use case for a decentralized network with enhanced privacy. Such a system must provide sufficient privacy to protect user transactions while ensuring transparency and scalability to facilitate mass adoption. By leveraging zk-SNARKs, the proposed network can achieve a balanced privacy level that ensures transaction confidentiality without hindering scalability or regulatory compliance.

In a global payment system built on the proposed network, users can transact through confidential payment channels, with zk-SNARKs proving the validity of transactions without revealing the amounts or the identities of the parties involved. This privacy level can be customized based on the specific requirements of different jurisdictions, ensuring compliance with local regulations while still preserving user privacy.

To facilitate cross-border transactions and enable seamless interoperability with existing payment systems, the network can integrate with traditional financial institutions and payment processors through secure off-chain communication channels. These channels can leverage zk-SNARKs to prove the authenticity of transactions and balances without revealing sensitive information, enabling a hybrid approach that combines the benefits of decentralized privacy with the reach and stability of established financial networks.

By leveraging zk-SNARKs in these use cases, the proposed decentralized payment network can provide enhanced privacy and scalability, making it suitable for a wide range of applications. These examples illustrate how the network can achieve a balance between privacy and transparency, facilitating mass adoption while maintaining the necessary confidentiality.

Conclusion

The proposed decentralized payment network offers:

1. Higher Throughput:

Comparable to or exceeding state channels and rollups.

1. Instant Finality:

Superior to Optimistic Rollups.

1. Cost Efficiency:

Reduces on-chain interactions and eliminates mining.

1. Enhanced Privacy:

Matches or surpasses ZK-Rollups.

The unique combination of features in the proposed network makes it a potentially more scalable and private solution compared to existing Layer 2 systems.

By leveraging zk-SNARKs in these use cases, we can provide enhanced privacy and scalability, making our decentralized network suitable for a wide range of applications. These examples illustrate how the network can achieve a balance between privacy and transparency, facilitating mass adoption while maintaining the necessary confidentiality.

References

1. Poon, J., & Dryja, T. (2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Lightning Network Whitepaper

. https://lightning.network/lightning-network-paper.pdf

1. Buterin, V., & Poon, J. (2017). Plasma: Scalable Autonomous Smart Contracts. Plasma Whitepaper

. https://plasma.io/plasma.pdf

1. Raiden Network Team. (2017). Raiden Network: Fast, Cheap, Scalable Token Transfers for Ethereum. Raiden Network

2. Celer Network. (2019). Celer Network: Bring Internet Scale to Every Blockchain. Celer Network Whitepaper

. https://www.celer.network/doc/CelerNetwork-Whitepaper.pdf

1. PLONK Documentation. (n.d.). ZK-SNARKs: PLONK. Retrieved from https://docs.plonk.cafe/

2. Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014). Scalable Zero-Knowledge via Cycles of Elliptic Curves. In International Cryptology Conference

(pp. 276-294). Springer, Berlin, Heidelberg.

1. Groth, J. (2016). On the Size of Pairing-based Non-interactive Arguments. In Annual International Conference on the Theory and Applications of Cryptographic Techniques

(pp. 305-326). Springer, Berlin, Heidelberg.

1. Zhang, F., Cecchetti, E., Croman, K., Juels, A., & Shi, E. (2016). Town Crier: An Authenticated Data Feed for Smart Contracts. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security

(pp. 270-282).

1. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., & Virza, M. (2015). SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In Annual Cryptology Conference

(pp. 90-108). Springer, Berlin, Heidelberg.

1. Hioki, L., Dompeldorius, A., & Hashimoto, Y. (2024). Plasma Next: Plasma without Online Requirements. Ethereum Research

. Retrieved from Plasma Next: Plasma without Online Requirements