Overpass Channels:

Global Scale, Private, Censorship Proof, Payments on TON

Introduction

Overpass Channels offers a revolutionary approach to blockchain scalability, enhancing privacy and providing robust security for decentralized payments on **The Open Network (TON)**. Traditional blockchain solutions suffer from issues like high transaction fees, slow confirmation times, and limited scalability. Overpass Channels addresses these problems by introducing a novel Layer 2 architecture that emphasizes privacy, scalability, and efficiency.

The system does not rely on traditional **miners** or **validators** for consensus, making it both **cost-effective** and **energy-efficient**. Instead, it utilizes **unilateral payment channels** and **off-chain transaction processing** to achieve **high throughput** without compromising security. Overpass Channels also integrates **zk-SNARKs** (zero-knowledge proofs) to ensure transaction privacy, meaning that transaction details are validated without revealing sensitive information like sender identity, recipient, or transaction amount.

In summary, Overpass Channels is designed to:

1. Scale horizontally, meaning it can support an unlimited number of transactions as more channels are opened.
2. Protect user privacy through advanced cryptographic techniques.
3. Allow independent verification of transactions without needing the entire network's agreement.
4. Ensure fluid liquidity through dynamic rebalancing, allowing funds to flow smoothly between users.
5. Resist censorship, ensuring no one can block valid transactions.

This paper explains the technical foundations, security features, scalability potential, and real-world applications of Overpass Channels, demonstrating how it could redefine decentralized payments.

Key Innovations

Overpass Channels incorporates several novel ideas that distinguish it from other blockchain or Layer 2 systems. Let’s break down these innovations in a way that’s easy to follow:

Horizontal Scalability without Validators

A core innovation is **horizontal scalability**—the ability to support more transactions as more channels are created—without relying on validators or miners to confirm transactions. Instead, the system uses **unilateral payment channels**, which allow two users to transact off-chain *(meaning they don’t need the blockchain for every transaction)*.

Unilateral Payment Channel: A unilateral payment channel is a secure connection between two users where they can conduct transactions off-chain. Only the opening and closing of the channel need to be registered on the blockchain, making the system much faster.

For example, if Alice and Bob want to trade tokens, they open a channel once on-chain, conduct all their transactions privately and quickly off-chain, and only update the blockchain when they close the channel.

Theorem: Horizontal Scalability: The number of transactions that Overpass Channels can handle grows with the number of active channels. This means that as more users create channels, the network can support an increasing number of transactions without overloading the underlying blockchain.

```
T = n × t
```

Where:

- `T` is the total transaction throughput.
- `n` is the number of active channels.
- `t` is the number of transactions each channel can handle per second.

Privacy-Enhanced Transactions with zk-SNARKs

Privacy is a key feature of Overpass Channels. The system uses **zk-SNARKs** (zero-knowledge Succinct Non-interactive Arguments of Knowledge) to prove that a transaction is valid without revealing any details about the transaction itself. This means the network can confirm that Alice sent Bob a payment, but it doesn't know how much was sent, or even that it was Alice and Bob involved in the transaction.

zk-SNARK Definition: zk-SNARKs allow one person (the prover) to prove to another (the verifier) that they know a piece of information without revealing the information itself.

Algorithm: zk-SNARK Transaction Proof Generation:

```
1. Alice generates a transaction proof.
2. Alice sends the proof to Bob.
3. Bob verifies the proof without learning anything else about the transaction.
```

In this system, every transaction is verified using zk-SNARKs, so while all transactions are guaranteed to be valid, no one can see who's involved or what amounts are being transferred.

Independent Verification and Instant Finality

Another key innovation is **instant finality**. In most blockchains, you need to wait for several confirmations before a transaction is considered final. With Overpass Channels, the transaction is finalized as soon as the recipient (Bob) verifies it. There's no need for the entire network to agree, which dramatically reduces waiting times.

Theorem: Instant Finality: As soon as Bob verifies the transaction using the zk-SNARK proof, the transaction is final. There's no need to wait for additional confirmations.

Fluid Liquidity through Dynamic Rebalancing

Liquidity, or the availability of funds to complete transactions, is managed dynamically in Overpass Channels. This system ensures that payment channels never run dry by constantly adjusting the distribution of funds between different channels.

Dynamic Rebalancing: The system checks the balance of each channel. If one channel is low on funds, it will rebalance by moving funds from other, less active channels.

```
Algorithm: Dynamic Channel Rebalancing
1. Check the balance of a channel.
2. If the balance is too low, pull funds from another channel.
3. Rebalance the system.
```

Robust Censorship Resistance

Censorship resistance is crucial for decentralized systems. Overpass Channels ensures that no single person or group can block a valid transaction from being processed. This is achieved through the use of zk-SNARKs and off-chain processing, where transactions can be verified independently by the parties involved without relying on the network.

Theorem: Censorship Resistance: No entity can block a valid transaction in Overpass Channels, as transactions are handled off-chain and verified with zk-SNARKs.

Transaction Validity and zk-SNARK Integration

The security and privacy of Overpass Channels are heavily reliant on **zk-SNARKs**. These cryptographic proofs ensure that transactions are both valid and private.

1. zk-SNARK Overview

zk-SNARKs are a type of cryptographic proof that allows one party to prove to another that a statement is true without revealing why it's true. In Overpass Channels, zk-SNARKs are used to validate transactions and ensure that balances remain accurate without exposing any transaction details.

2. zk-SNARK Circuit for Transaction Validation

A special type of circuit is used to validate transactions in Overpass Channels. This circuit checks the following:

- The transaction is properly signed by the sender.
- The sender has enough balance to complete the transaction.
- The balances are updated correctly after the transaction.

```
Algorithm: Transaction Validation Circuit
1. Check if the signature is valid.
2. Confirm the sender's balance is sufficient.
3. Update the balance accordingly.
```

The use of this circuit ensures that all transactions are valid while keeping them private.

Proof Generation and Verification

When users transact, they sign the new state of their channel (which includes the updated balances) to prevent disputes. After each transaction, the new state is agreed upon by both parties, preventing any disagreements about the transaction history.

1. Privacy Preservation

The zk-SNARKs used in Overpass Channels also ensure that no private information is leaked during a transaction. The system proves that a transaction is valid without exposing the transaction details or the identity of the participants.

Proof Generation and Verification

In Overpass Channels, when users engage in a transaction, they are required to generate a **zk-SNARK proof**. This proof ensures that the transaction complies with the established rules without revealing the sensitive details of the transaction. Each transaction has to be signed based on the current state of the channel to preserve consistency between participants. This signed proof serves to prevent disputes about the transaction order or validity.

How It Works:

1. Alice initiates a transaction and signs it based on the current channel state.
2. A zk-SNARK proof is generated that validates the transaction.
3. The transaction is broadcast off-chain, and the state of the channel is updated based on Alice's signed proof.
4. Bob, the receiver, can verify the transaction at any time, but Alice's signature guarantees the transaction is valid and compliant with all rules.

```
Algorithm: Transaction Proof Generation and Verification


1. Alice signs the transaction based on the current channel state.
2. Generate zk-SNARK proof based on transaction and updated channel state.
3. Verify the proof.
```

This mechanism enforces trust and ensures consistency within the channel, preventing any fraudulent transactions or attempts to revert to an outdated state.

Unilateral Channels: How They Work

Overpass Channels introduces **unilateral payment channels**, which allow one party to execute transactions without needing the other party to be immediately available online.

Key Characteristics of Unilateral Channels:

1. **No Immediate Receiver Involvement**: Alice can send a payment to Bob without Bob needing to verify the transaction in real-time. Bob can review and verify the transaction later.

2. **Pre-agreed Channel Rules**: When the channel is created, both parties (Alice and Bob) agree on how transactions will be handled, including how balances are updated.
3. **Asynchronous Execution**: Alice can initiate transactions at any time, even if Bob is offline. When Bob comes online, he can verify the updated state of the channel.

```
Transaction Flow in Unilateral Channels:

1. Alice sends tokens to Bob.
2. Alice signs the updated state, which reflects the new balances.
3. Bob can verify the updated state whenever he returns online.
```

This allows for **instant and asynchronous transactions** that do not depend on both parties being online simultaneously.

Balance Consistency

Maintaining the correct balance across all channels is critical for ensuring the integrity of the Overpass Channels system. Balance consistency means that no participant's balance can become negative, and the sum of all balances across the system remains constant.

Formal Definition of Balance Consistency

A channel maintains **balance consistency** if:

1. The sum of all balances across channels stays constant, except for external deposits and withdrawals.
2. The sum of balances in each channel equals the channel's total capacity.
3. No participant's balance becomes negative.

This prevents double-spending and ensures that all transactions respect the system's rules.

Theorem of Balance Consistency:

Every valid transaction in Overpass Channels preserves balance consistency. This theorem is proven using induction, starting from the initial state of the network (when all channels are opened with a predefined capacity) and ensuring that every subsequent transaction adheres to these balance rules.

```
Proof:
1. At network initialization, all channels are consistent.
2. Every valid transaction maintains the balance consistency
   by adjusting balances according to the transaction details.
```

This guarantees that users cannot overspend their balance and prevents any inconsistencies from arising during transactions.

Fraud Prevention Mechanisms in Overpass Channels

Ensuring security is one of the key goals of Overpass Channels. The system implements several mechanisms to prevent fraud and abuse, especially in situations where one party might be offline or trying to cheat the system.

1. Pending Transaction Acceptance

To prevent one party (e.g., Alice) from taking advantage of the other party's (e.g., Bob's) absence, Overpass Channels uses a mechanism where **pending transactions** must be accepted by the receiver before new transactions can occur. For example:

- Alice sends tokens to Bob.
- The transaction remains **pending** until Bob verifies it.
- Alice cannot send any new transactions until Bob accepts the previous one.

2. Extended Absence Mitigation

If Bob is offline for an extended period, Alice's channels won't freeze. Instead, Alice can continue using other channels. This is ensured through **dynamic rebalancing**, which automatically manages liquidity between Alice's other channels while Bob is offline. At the same time, Bob is protected from any fraudulent transactions while offline, as no new transactions can proceed until he returns.

3. Deterministic Conflict Resolution

When a conflict arises between two parties about the state of the channel, the system uses **deterministic conflict resolution**. The latest valid state (based on nonces and zk-SNARK proofs) is always chosen, ensuring that disputes are resolved quickly and fairly.

Deterministic Conflict Resolution

In Overpass Channels, conflict resolution is deterministic and guarantees that disputes over the state of a payment channel are resolved in a fair and consistent manner.

How Conflict Resolution Works

1. **Conflicting States**: In the event of a conflict, such as two parties disagreeing on the current state of a channel, each state must be accompanied by a valid zk-SNARK proof. For instance, Alice might submit state \(S1\) while Bob submits state \(S2\).
2. **Proof Verification**: The on-chain smart contract will validate both zk-SNARK proofs (P1 and P2) associated with the respective states \((S1 and S2)\).
3. **Nonce Comparison**: After verifying the proofs, the contract checks the **nonces** attached to the states. The state with the higher nonce is selected as the valid one because it represents the more recent transaction.
4. **Tie-breaking Rule**: In the rare case that both states have the same nonce, the smart contract uses a deterministic tie-breaking rule, such as selecting the state with the lexicographically smaller transaction hash.

```
Theorem: Deterministic Conflict Resolution
- Given two conflicting states (S1 and S2), the state with the higher
  nonce is selected.
- If the nonces are identical, a tie-breaking rule is applied.
```

This ensures that conflicts are always resolved the same way, preventing any ambiguity or misuse of the system.

50% Spending Rule for Off-Chain Transactions

To protect against attacks where one party tries to drain their balance in a single transaction, Overpass Channels introduces the **50% spending rule**.

What the Rule Ensures

1. **Transaction Limit**: Each party can spend only up to 50% of their current balance in a single off-chain transaction. This rule limits the potential damage in case of malicious behavior.
2. **Griefing Prevention**: If Alice tries to act maliciously by draining her channel balance, she can only spend up to half of her balance. This ensures that the other party (Bob) still has a recourse to close the channel and claim a portion of the remaining funds.

Proof of the 50% Spending Rule:

- If Alice's balance is \(BA\) and Bob's balance is \(BB\), Alice can spend at most \(BA/2\) in one transaction.
- After this, her balance will be at least \(BA/2\), ensuring that Bob can recover his funds even in the worst-case scenario by closing the channel on-chain.

This rule makes it computationally infeasible for Alice to commit fraud without consequences, as Bob can always recover funds through on-chain resolution.

```
Theorem: Griefing Prevention
- The 50% spending rule prevents malicious parties from depleting the entire
  channel balance in a single transaction.
```

---

zk-SNARK Proofs and State Updates

At the heart of Overpass Channels is the use of **zk-SNARKs** for validating state updates. This ensures both privacy and correctness of off-chain transactions while minimizing the load on the blockchain.

How zk-SNARKs Work in State Updates:

1. **State Update Generation**: Each state transition in a channel must be accompanied by a zk-SNARK proof. This proof validates that the update adheres to the channel rules, such as balance consistency, signature verification, and proper nonce incrementing.

2. **Proof Verification**: Both parties (Alice and Bob) can independently verify the zk-SNARK proof. If the proof is valid, the new state is accepted by the system.
3. **Off-chain Processing**: The zk-SNARK proof is verified off-chain, ensuring that the transaction is processed quickly and without requiring every step to be recorded on the blockchain.

```
Algorithm: State Update Validation
1. Generate zk-SNARK proof for the state update.
2. Verify proof against the rules of the channel (balance, nonce, signatures).
3. If valid, accept the new state and update the channel.
```

The use of zk-SNARKs ensures that the state transition is correct without revealing any private details about the transaction, such as how much was transferred or who was involved.

Cross-Shard Transaction Security

Overpass Channels is designed to operate efficiently in a **sharded blockchain environment**, such as **TON**. Sharding allows the network to process many transactions in parallel by dividing the blockchain into smaller pieces (shards). Cross-shard transactions (transactions that involve participants on different shards) require additional security considerations to ensure that they are as secure as intra-shard transactions.

Ensuring Cross-Shard Security:

1. **Transaction Proofs**: Cross-shard transactions are accompanied by zk-SNARK proofs that verify the validity of the transaction across different shards. Each shard verifies the state transitions independently using zk-SNARK proofs.
2. **Merkle Root Verification**: Each shard maintains a **Merkle tree** of its current state. When a cross-shard transaction occurs, both the source and destination shards verify that their respective Merkle roots are consistent before finalizing the transaction.
3. **Final Settlement**: Once the transaction is verified by both shards, the final balances are updated and the transaction is considered complete.

```
Theorem: Cross-Shard Security
- Cross-shard transactions in Overpass Channels are as secure as intra-shard
- transactions due to zk-SNARK proof verification and Merkle root
  consistency checks.
```

This mechanism ensures that transactions across different shards maintain the same security guarantees as those within a single shard, ensuring the integrity of the entire system.

Prevention of Old State Submission

One potential attack vector in payment channel systems is the submission of an **old state** in an attempt to revert the channel balance to a previous state, essentially stealing funds. Overpass Channels prevents this by utilizing a combination of **nonces**, zk-SNARK proofs, and the TON smart contract system's **seqno** (sequence number) capabilities.

How It Works:

1. **Nonce Mechanism**: Each channel state is assigned a **nonce**, a number that increments with each valid state update. This ensures that each new state is easily distinguishable from older ones.
2. **SEQNO in Channel Contracts**: Each channel contract on TON is assigned a **seqno**, which increments with each new valid state. This seqno is stored on-chain, ensuring that old states cannot be submitted.
3. **Invalidating Old States**: If an adversary attempts to submit an old state, the system's zk-SNARK proof verification process will check whether the nonce and seqno of the old state are greater than the current valid state. Since they aren't, the old state will be rejected.

Theorem: Old State Invalidation

The system guarantees that it is computationally infeasible for an attacker to submit an old state as the current valid state.

Proof:

1. Suppose an attacker tries to submit an old state, S_{old}, with a **nonce** n_{old} and **seqno** seq_{old}.
2. The current valid state is S_{current}, with a nonce n_{current} and seqno $\text{seq}_{\text{current}}$.
3. The attacker would need to prove that $n_{\text{old}} > n_{\text{current}}$ and $\text{seq}_{\text{old}} = \text{seq}_{\text{current}}$, which contradicts the system's rules.
4. By the nature of zk-SNARKs and TON's seqno, it is computationally infeasible to generate such a proof, making this type of fraud impossible.

On-Chain Verification of Channel Closure

While most transactions in Overpass Channels happen off-chain, the system incorporates **on-chain verification** to ensure a final layer of fraud prevention. This comes into play during the **closing of payment channels**, where the final state is submitted on-chain for verification.

How On-Chain Verification Works:

1. **Submission of Final State**: When a participant wishes to close a channel, they submit the **final state** and a corresponding **zk-SNARK proof** to the on-chain contract.
2. **State Validation**: The on-chain contract verifies the zk-SNARK proof to ensure that the final state is consistent with the channel's recorded history.
3. **Distribution of Funds**: Once the final state is validated, the on-chain contract distributes the funds according to the channel's balance.

Algorithm: On-Chain Channel Closure Verification

```
1. Retrieve stored channel state.
2. Verify zk-SNARK proof against final state and stored state.
3. If valid, update channel state and distribute funds.
4. If invalid, reject the closure attempt.
```

This on-chain verification acts as a safeguard, ensuring that even if off-chain processes fail or are manipulated, the on-chain settlement remains secure and correct (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Elimination of the Need for Watchtowers

In traditional payment channel networks, external entities called **watchtowers** are used to monitor the blockchain for fraudulent transactions, particularly attempts to close channels with old states. However, Overpass Channels eliminates the need for such watchtowers by using zk-SNARK proofs and on-chain verification.

Why Watchtowers Are Redundant in Overpass Channels:

1. **Fraud Detection through zk-SNARKs**: Since every transaction is accompanied by a zk-SNARK proof, any attempt to close a channel with an old state will automatically fail the proof verification process. Therefore, there is no need for an external entity to watch for fraudulent behavior.
2. **Nonce and Seqno Checks**: The system's **nonce** and **seqno** mechanisms ensure that only the most recent state can be submitted for closure.
3. **On-Chain Verification**: Even in the event of an off-chain dispute, the on-chain contract will always reject fraudulent attempts to close a channel with outdated information.

Theorem: Watchtower Redundancy

The design of Overpass Channels makes watchtowers unnecessary by ensuring that the on-chain verification process is robust enough to detect and prevent any fraudulent channel closures.

Proof:

1. For any attempt to close a channel, the participant must submit the final state along with a valid zk-SNARK proof.
2. The on-chain contract verifies both the proof and the state.
3. If an old state is submitted, the zk-SNARK proof will fail to verify, and the closure will be rejected.
4. Therefore, Bob (or any other participant) can close the channel securely without relying on a watchtower.

This significantly reduces operational overhead and complexity compared to systems that require watchtowers (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Cryptographic Proofs and Tamper-Evident Records

One of the key guarantees provided by Overpass Channels is that the entire history of wallet and channel states remains tamper-evident and verifiable. This is crucial to maintaining the integrity and transparency of the system.

1. **Wallet State Representation**: Each wallet contract is associated with a **Sparse Merkle Tree (SMT)**. In this structure:
    - Each **leaf node** represents a channel's state, such as its balance and transaction history.
    - The leaf is defined by the hash of the state $H(S_i)$, where S_i represents the i-th state.
    - The position of the leaf in the tree is determined by the **SEQNO** (sequence number) of the channel.

2. **Merkle Roots and zk-SNARK Integration**: The **root** of the Sparse Merkle Tree is included in each zk-SNARK proof. Any change to the channel state automatically updates the tree, meaning that altering any part of the history would modify the Merkle root. This ensures data integrity.
    - When state S_i transitions to state S_{i+1}, the zk-SNARK proof P_i is generated. This proof confirms that S_{i+1} is a valid successor of S_i.
3. **Collision Resistance**: Because the underlying cryptographic hash function H is collision-resistant, it is computationally infeasible to alter past states without being detected. Any tampering with historical states will break the chain of zk-SNARK proofs.

Hierarchical Sparse Merkle Trees in Overpass Channels

Overpass Channels leverages **Sparse Merkle Trees (SMTs)** to manage and verify the states of wallets and payment channels efficiently. These trees play a crucial role in achieving scalability while minimizing storage overhead.

Benefits of Sparse Merkle Trees:

1. **Compact Proofs**: Sparse Merkle Trees enable constant-size proofs, regardless of the number of leaves in the tree. This ensures that verification remains efficient even as the network scales.
2. **Efficient Updates**: When a state changes (e.g., when a channel balance is updated), only the path from the updated leaf to the root needs to be recalculated. This keeps the system responsive with a complexity of $O(\log n)$, where (n) is the number of leaves (channels).
3. **Proof of Integrity**: Every change to a channel or wallet state is reflected in the Merkle root, ensuring data integrity at each level of the hierarchy. If a single leaf is modified, the corresponding proofs and root are automatically updated, making tampering evident.

Hierarchical Structure of Overpass Channels

The architecture of Overpass Channels is organized in a **multi-tiered hierarchy**, where states are aggregated at each level, enabling scalability and system efficiency. Let’s look at how this works across different levels.

Hierarchical Levels:

1. **Payment Channel Contracts**:
    - At the lowest level, individual payment channels between users are managed. These are **unilateral channels** where transactions happen off-chain.
    - A channel contract keeps track of each channel’s balance and updates through the use of zk-SNARK proofs and Merkle roots.
2. **Wallet Contracts**:
    - Each wallet contract manages a collection of payment channels. The state of each wallet is represented as a Sparse Merkle Tree, where each leaf represents a payment channel.
    - Wallet contracts periodically submit their aggregated state (Merkle root) to an **intermediate contract**.

3. **Intermediate Contracts**:
    - Intermediate contracts aggregate the states from multiple wallet contracts. This significantly reduces the computational load on the **root contract**.
    - For example, if an intermediate contract manages several wallets, it creates an SMT of their states, which is periodically submitted to the root contract.
4. **Root Contract**:
    - The root contract is responsible for maintaining the global state of the system by aggregating Merkle roots from all intermediate contracts. This root is then submitted on-chain for verification .

System-Level Efficiency through Sparse Merkle Trees

The hierarchical structure, underpinned by Sparse Merkle Trees, enables **system-level efficiency** in Overpass Channels. Key efficiency gains come from how these trees are employed at each layer of the hierarchy.

1. **Global State Verification**: The root contract aggregates Merkle proofs from intermediate contracts and submits a global Merkle root on-chain at regular intervals. This provides a tamper-evident, decentralized verification of the entire network's state.
2. **Redundant and Decentralized Storage**: Each node in the network stores only a subset of the global data but can verify the entire system using Merkle proofs. This provides scalability without the need for each node to store all data .

Tokenomics in Overpass Channels

The **tokenomics** of Overpass Channels are designed to ensure both stability and sustainability for the network, incentivizing various participants and maintaining a balanced ecosystem. Key aspects of this system involve the initial distribution of tokens, governance and treasury management, and a structured fee system.

1. Fixed Supply and Initial Distribution

Overpass Channels operates on a **fixed supply** model, where all tokens are minted during the network's genesis. This allows for precise tracking of token movement across the ecosystem. The total token supply is set at **100 billion tokens**.

Distribution Breakdown:

- **70% (70 billion tokens)** are allocated to a community **airdrop**. This strategy aims to distribute tokens widely, fostering community engagement from the outset.
- **20% (20 billion tokens)** are reserved for the **Treasury and Governance** system. This allocation will be used to fund future developments and ensure the long-term sustainability of the ecosystem.
- **10% (10 billion tokens)** are reserved for **team members, investors, and advisors**. This incentivizes early participants and helps build the network by rewarding those contributing to its success.

The goal of this distribution model is to ensure wide participation, with a significant portion allocated to community members, while maintaining sufficient resources for development and governance decisions.

2. Governance and Treasury

The 20 billion tokens allocated to **Treasury and Governance** ensure that the Overpass Channels ecosystem remains sustainable and adaptable. These tokens serve several purposes:

- **Governance**: Token holders will be able to participate in decentralized governance, making key decisions about the future of the network.
- **Funding**: The treasury will fund ecosystem development, marketing efforts, and infrastructure improvements, ensuring the network can grow and adapt over time.

The governance mechanism plays a central role in ensuring that the treasury is allocated efficiently for long-term growth.

3. Token Utility and Fee Structure

The Overpass Channels token has a vital role in the network's operations. **Transaction fees** collected from users fuel the system by compensating key participants:

1. **Storage Nodes**: A portion of the transaction fees compensates off-chain storage nodes that maintain the channel state data efficiently.
2. **TON Nodes**: Since Overpass Channels is built on **TON**, part of the fees is also used to compensate TON validators that store and verify global root state data.

Additionally, the system recycles some tokens back into the reserve to provide liquidity for future developments.

```
Key Utility Points:
1. Off-chain storage node payments.
2. On-chain TON node compensation.
3. Fee recycling into reserves.
```

4. Fee Distribution Model

The fee distribution model ensures the sustainability of the network while compensating critical infrastructure providers. Here's how fees are distributed:

- **Recycling to the Reserve**: A portion of the fees returns to the reserve fund to support future developments and ensure the ecosystem's longevity.
- **Off-Chain Node Payments**: These payments go to off-chain storage nodes for maintaining and updating channel states.
- **On-Chain Node Payments**: A portion of fees compensates TON nodes for their role in processing and verifying the system's global state.

This ensures that the core participants are incentivized while maintaining a dynamic and self-sustaining economy.

Dynamic Rebalancing and Liquidity Management

Overpass Channels implements a **dynamic rebalancing** system that ensures smooth liquidity management across all channels. Liquidity management in Overpass Channels is key to enabling the **fluid operation of payment channels** and ensuring that users can transact freely, even when some channels become inactive or less liquid.

How Dynamic Rebalancing Works:

1. **Channel Liquidity**: Liquidity refers to the funds available in a payment channel to make transfers. Each user may have multiple payment channels open, and sometimes certain channels can run low on liquidity.
2. **Rebalancing Between Channels**: If Alice wants to send funds to Bob, but her channel with Bob is low on liquidity, the system dynamically pulls liquidity from less active channels (e.g., Alice's channel with Charlie) to ensure the payment goes through without requiring on-chain intervention.
3. **Proof-Based Rebalancing**: All rebalancing actions are verified using cryptographic proofs, ensuring that liquidity is adjusted correctly without violating the channel rules.

This system allows the network to continue operating smoothly without forcing users to close channels or initiate on-chain transactions, providing scalability and efficiency.

```
Algorithm: Dynamic Rebalancing


1. Detect low liquidity in a channel.
2. Pull liquidity from less active channels.
3. Generate cryptographic proof to verify the updated balance.
4. Rebalance the channel.
```

Dynamic rebalancing ensures that the Overpass Channels system can handle varying levels of channel activity while maintaining efficiency and minimizing the need for on-chain interaction.

Censorship Resistance in Overpass Channels

Censorship resistance is a key feature of Overpass Channels, designed to ensure that no entity can block or alter the transmission of transactions. This property is vital for maintaining the **decentralized** nature of the network.

Theorem: Censorship Impossibility

In Overpass Channels, it is **computationally infeasible** for any entity or group to censor transactions. This is due to the system's design, where:

1. **Off-chain Transactions**: All individual transactions are processed off-chain, meaning that network nodes do not directly handle or see transaction details. Instead, they work with **Merkle tree roots** and **zk-SNARK proofs**.
2. **Epoch-Based Processing**: Transactions are batched into **epochs**. Each epoch is represented by a Merkle tree root, which is submitted to the network along with a zk-SNARK proof validating the epoch.
3. **Infeasibility of Rejection**:
    - Rejecting a valid Merkle root would require rejecting all the transactions within that epoch, which would result in the loss of valid transactions. This is against the economic interests of network nodes.

- zk-SNARK proofs themselves reveal no information about individual transactions, making it impossible to censor specific transactions without rejecting entire epochs of valid ones.

Proof of Censorship Impossibility:

1. A transaction, (T), can be censored only if:
   - It is excluded from the Merkle tree root.
   - A valid Merkle root containing (T) is rejected.
   - A valid zk-SNARK proof for an epoch including (T) is rejected.
2. Since Merkle tree construction is performed locally by channel participants, exclusion of transactions is infeasible.
3. Nodes cannot distinguish which transactions are in a Merkle root due to the zk-SNARK structure, making selective rejection impossible.
4. Rejecting an entire epoch of valid transactions is computationally and economically infeasible for nodes (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Security Analysis of zk-SNARK Integration

The **security** of Overpass Channels is based on zk-SNARKs, which allow for the validation of transactions and state transitions without exposing sensitive information.

1. **zk-SNARK Circuit for Transaction Validation**

The circuit used for zk-SNARK transaction validation is designed to:

1. **Check Signatures**: Ensure that the transaction is signed by the rightful party.
2. **Validate Balances**: Confirm that the sender has enough balance to complete the transaction.
3. **State Transition Validation**: Ensure that the state transition (i.e., balance updates) follows the system's rules.
4. **Nonce Management**: Ensure that the nonce for each transaction is incremented properly, preventing replay attacks.

```
Algorithm: zk-SNARK Circuit for Transaction Validation
1. Validate signature.
2. Check if sender has sufficient balance.
3. Validate state transition from old to new state.
4. Ensure nonce is incremented correctly.
```

These steps guarantee that all transactions in the network are valid, secure, and cannot be manipulated by an adversary (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

2. **zk-SNARK Proof Security**

The zk-SNARK proofs used in Overpass Channels are designed to be **succinct** and **non-interactive**, meaning they can be verified without the need for back-and-forth communication between the prover and verifier. This allows for:

- **Efficient Verification**: Proofs can be verified quickly, ensuring that network performance is not slowed down.

- **Privacy Preservation**: zk-SNARKs guarantee that no transaction details are revealed during verification. All that is disclosed is the validity of the transaction.

These properties ensure that Overpass Channels can scale while maintaining strong security guarantees (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Post-Quantum Security

As quantum computing advances, current cryptographic systems (including zk-SNARKs) could potentially be compromised. Overpass Channels acknowledges this risk and includes plans to develop **post-quantum secure** zk-SNARK systems.

Theorem: Post-Quantum Vulnerability of zk-SNARKs

Current zk-SNARK implementations rely on cryptographic assumptions that are vulnerable to **quantum attacks**. Specifically, they depend on the hardness of problems like the **discrete logarithm** or **elliptic curve discrete logarithm**, both of which can be solved in **polynomial time** using **Shor's algorithm** on a sufficiently powerful quantum computer.

Proposed Solutions for Post-Quantum Security:

1. **Lattice-Based zk-SNARKs**: Future research will focus on adapting zk-SNARKs to use **lattice-based cryptography**, which is currently believed to be resistant to quantum attacks.
2. **Other Post-Quantum Approaches**: Research into **hash-based**, **code-based**, and **multivariate quadratic** cryptosystems may offer alternatives that can be incorporated into zk-SNARK frameworks (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Privacy Analysis and Use Cases

The privacy guarantees of Overpass Channels are at the core of its design, focusing on ensuring that transaction data is kept private while maintaining security and scalability. The system's use of **zk-SNARKs** ensures that transaction data can be validated without revealing critical details like participant identities or transaction amounts.

Theorem: Overpass Channels Transaction Privacy

In Overpass Channels, zk-SNARK proofs are utilized to ensure that no transaction data is exposed beyond the minimum required for validation. The following components are key to this theorem:

1. **Off-Chain Processing**: Transactions are processed off-chain, and only aggregated, cryptographically protected data is submitted on-chain.
2. **Merkle Roots**: Merkle roots of transaction batches are shared with the network, ensuring tamper-evidence while protecting transaction details.
3. **zk-SNARK Proofs**: zk-SNARKs are used to verify state transitions, revealing no information beyond the fact that the transaction adheres to the required rules.

This ensures that even though the network processes the transaction, no sensitive information about the transaction itself is revealed, protecting both parties involved (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Future Directions and Challenges

Post-Quantum Security

As advancements in **quantum computing** emerge, ensuring the security of Overpass Channels in a post-quantum world becomes a crucial objective. Current cryptographic techniques, including those used in zk-SNARKs, rely on problems such as the **discrete logarithm problem** or the **elliptic curve discrete logarithm problem**. These problems, though secure against classical computers, can be efficiently solved by quantum computers using **Shor's algorithm**.

Theorem: Post-Quantum Vulnerability

Current zk-SNARK implementations in Overpass Channels are vulnerable to quantum attacks.

Proof:

1. zk-SNARKs depend on cryptographic assumptions like the discrete logarithm.
2. A sufficiently powerful quantum computer can use Shor's algorithm to break these cryptographic systems in polynomial time.

To mitigate this, research into **post-quantum secure** zk-SNARKs will focus on replacing existing cryptographic primitives with **lattice-based cryptography** or other quantum-resistant algorithms (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf))..

Privacy-Preserving Analytics

While Overpass Channels offers robust privacy guarantees, this can complicate network-wide analytics. Analyzing the system's performance or detecting potential issues becomes challenging when no detailed transaction data is revealed. However, **privacy-preserving analytics** techniques such as **secure multi-party computation (SMPC)** or **differential privacy** may provide ways to collect useful information without compromising the system's privacy promises.

Theorem: Privacy-Analytics Tradeoff

There is a fundamental tradeoff between maintaining transaction privacy and enabling network-wide analytics in Overpass Channels.

Potential Solutions:

1. **Secure Multi-Party Computation**: Enables different parties to jointly compute analytics without revealing sensitive data.
2. **Differential Privacy**: Adds noise to analytics to protect individual transaction details while providing aggregate data insights.

By employing these techniques, Overpass Channels can strike a balance between privacy and operational visibility (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf))..

Use Cases for Privacy in Overpass Channels

Overpass Channels presents multiple use cases, demonstrating its wide-reaching impact and the scalability of its privacy solutions. Here are a few key real-world applications that leverage the zk-SNARK-enabled privacy framework within the Overpass Channels system:

1. Confidential Voting Systems

Decentralized voting systems need to ensure the **confidentiality** of votes while maintaining transparency for public verification of results. Overpass Channels supports such systems by using zk-SNARKs to keep individual votes secret without compromising the integrity of the voting process.

Implementation:

1. **Voter Registration**: Smart contracts on TON manage voter registration, issuing unique anonymous identifiers to eligible voters.
2. **Vote Casting**: Voters cast their ballots through Overpass Channels, generating a zk-SNARK proof. This ensures:
    - The voter is eligible.
    - The voter hasn't voted before.
    - The vote is valid without revealing which candidate they voted for.
3. **Vote Tallying**: A final zk-SNARK proof is generated after tallying all votes, ensuring that the total vote count is accurate without revealing individual choices.
4. **Result Verification**: The election outcome can be verified by checking the zk-SNARK proof on the TON blockchain without revealing individual votes.

This approach ensures election transparency while protecting voter privacy (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

---

2. Private Asset Transfers

Another crucial use case for Overpass Channels is **confidential asset transfers**, allowing users to transfer digital assets while keeping transaction details private. The integrity and validity of each transaction are maintained through zk-SNARK proofs without exposing sensitive information.

Implementation:

1. **Tokenized Assets**: Digital assets are represented as tokens on the TON blockchain.
2. **Private Transfer Process**: Users transfer these tokens privately via Overpass Channels, using zk-SNARK proofs that confirm:
    - The sender owns the assets being transferred.
    - The transaction amount is valid.
    - The recipient is a legitimate receiver of the assets.

3. **On-Chain Settlement**: Periodically, the system updates the aggregate state of all transfers on-chain, ensuring final balances are correct without revealing individual transaction details.
4. **Audit Capabilities**: Users can generate zk-SNARK proofs of their transaction history for audit purposes without compromising privacy (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf))...

---

3. Secure Health Records Management

In the healthcare domain, managing sensitive health records securely is vital. Overpass Channels offers a solution that ensures that patient data remains private while providing verification capabilities for authorized parties.

Implementation:

1. **Off-Chain Record Storage**: Health records are stored off-chain, with their cryptographic hashes maintained through Overpass Channels.
2. **Access Management**: Smart contracts handle access permissions, allowing patients to control who has access to their data.
3. **Data Sharing**: zk-SNARK proofs are used to validate that:
    - The data belongs to the correct patient.
    - The party requesting the data has the correct authorization.
    - The health data has not been tampered with.
4. **Verification**: Authorized parties can verify the authenticity and integrity of the records using zk-SNARK proofs on the TON blockchain without viewing the actual content (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

---

4. Global Payment System

A scalable and privacy-focused global payment system is a significant use case for Overpass Channels. By leveraging zk-SNARKs, Overpass Channels ensures privacy while maintaining the necessary transparency and scalability for mass adoption.

Implementation:

1. **Multi-Currency Support**: Overpass Channels supports multiple currencies, with exchange rates managed by on-chain oracles.
2. **Cross-Border Transfers**: Users can send cross-border payments, and zk-SNARK proofs ensure:
    - The sender has sufficient funds.
    - Currency conversion rates are accurate.
    - Compliance with regulations, such as Anti-Money Laundering (AML) checks, is enforced without exposing transaction details.
3. **Liquidity Pools**: Smart contracts manage liquidity pools for seamless currency conversion.
4. **Regulatory Compliance**: zk-SNARKs generate confidential reports for regulators, ensuring compliance without compromising user privacy.
5. **Integration with Traditional Finance**: Bridge contracts interact with traditional financial systems, allowing users to deposit and withdraw funds while keeping transaction details private within Overpass Channels (Overpass

Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Implementation Considerations

Implementing Overpass Channels on a sharded blockchain like **TON** presents several unique technical challenges and optimizations that ensure the system's security, efficiency, and scalability. These considerations are vital for maintaining smooth transaction processing, minimal data sharing across shards, and seamless integration of zk-SNARKs.

1. Merkle Tree Structure

Overpass Channels leverages **Sparse Merkle Trees (SMTs)** to efficiently represent the state of all channels across the network. The key advantage of SMTs is their compactness, allowing for constant-size proofs regardless of the number of leaves. This ensures that verification costs remain low, even as the system scales to support millions of transactions and channels.

Implementation:

- **Merkle Root Updates**: As new transactions are processed, the state of a channel is hashed and updated in the Merkle tree.
- **Proof Generation**: A Merkle proof is generated for each channel update, allowing participants to verify the integrity of the update without needing to inspect the entire tree.
- **Root Submission**: Periodically, the global Merkle root (aggregated from all wallet contracts) is submitted on-chain for verification.

The efficiency of this structure ensures that Overpass Channels can scale to handle a massive number of transactions without imposing a heavy computational burden on the blockchain.

```
Algorithm: Merkle Tree Update

1. procedure UpdateMerkleTree(channelID, newState)
2. leaf ← Hash(channelID ‖ newState)
3. path ← GetMerklePath(channelID)
4. newRoot ← ComputeNewRoot(leaf, path)
5. UpdateRoot(newRoot)
6. end procedure
```

---

2. zk-SNARK Circuit Design

The design of the **zk-SNARK circuit** in Overpass Channels is fundamental to ensuring privacy and security across all transactions. The circuit encodes the logic for state transitions within channels, ensuring that no invalid state updates can be processed.

Key components of the zk-SNARK circuit include:

- **Signature Verification**: Ensures that the transaction has been authorized by the legitimate party.

- **Balance Transition**: Confirms that the sender's balance is sufficient and that the transition complies with the channel's rules.
- **Nonce Management**: Prevents replay attacks by ensuring that the nonce is incremented properly with each transaction.

```
Algorithm: zk-SNARK Circuit for State Transition

1. procedure TransitionCircuit(seqno, oldState, newState, tx, signature, nonce)
2. AssertValidSignature(tx, signature)
3. AssertValidBalanceTransition(oldState, newState, tx)
4. AssertValidNonceIncrement(nonce, oldState.nonce, newState.nonce)
5. end procedure
```

---

3. Cross-Shard Transaction Data Minimization

Cross-shard transactions, which involve communication between different shards, are optimized in Overpass Channels to minimize data transfer and reduce overhead. The system uses compact data representations for cross-shard communication, ensuring that the state transitions are communicated efficiently without revealing excessive information.

Key Techniques:

- **Shard Identification**: Cross-shard transactions are identified by their `shardID` and `workchainID`, ensuring that only the minimal required data is shared between shards.
- **Compact zk-SNARK Proofs**: The zk-SNARK proofs for cross-shard transactions are compressed to reduce the data footprint, further minimizing the communication overhead.

```
Algorithm: Cross-Shard Transaction Encoding

1. function EncodeCrossShardTx(tx, proof, workchainID, shardID)
2. encodedTx ← Hash(tx)
3. encodedProof ← CompressZKProof(proof)
4. return encodedTx ‖ encodedProof ‖ workchainID ‖ shardID
```

These optimizations allow Overpass Channels to scale across multiple shards while maintaining low latency and high throughput for cross-shard operations .

Liquidity Models and Incentive Mechanisms

In Overpass Channels, liquidity models and incentive mechanisms are designed to support the efficient operation of the network. These systems ensure that participants are compensated for providing liquidity and that the system remains balanced to prevent transaction bottlenecks.

1. Liquidity Provision

The liquidity pools in Overpass Channels play a crucial role in enabling smooth transactions across the network. Liquidity providers (LPs) deposit assets into **hub contracts**, enabling users to trade and transfer assets without experiencing significant slippage or delays.

How it works:

- **Liquidity Pools**: LPs deposit assets into on-chain liquidity pools managed by smart contracts. In return, they receive **LP tokens** representing their share in the pool.
- **Reward Distribution**: As transactions and trades are executed, fees are collected and distributed among the LPs based on their contribution to the liquidity pool.
- **Liquidity Tracking**: The amount of liquidity used in trades is tracked off-chain using **Sparse Merkle Trees** (SMTs), which are periodically updated to reflect usage.

```
Algorithm: Liquidity Distribution

1. LP deposits assets into the hub contract.
2. Hub contract updates the liquidity pool.
3. LP receives LP tokens as proof of deposit.
4. Rewards are calculated based on trading volume.
5. SMT updates reflect liquidity usage.
```

This approach ensures the system has enough liquidity to handle large trades while also compensating LPs fairly (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

2. Dynamic Rebalancing

To prevent liquidity imbalances where certain channels run low on liquidity, Overpass Channels implements a **dynamic rebalancing** mechanism. This feature allows the system to redistribute liquidity from less active channels to more active ones as needed, ensuring that all transactions are processed smoothly.

How Dynamic Rebalancing Works:

- The system constantly monitors the liquidity levels of all channels.
- When liquidity is running low in one channel, the system pulls liquidity from other, less active channels.
- This rebalancing is performed off-chain, ensuring minimal latency and transaction costs.

```
Algorithm: Dynamic Rebalancing

1. Monitor channel liquidity levels.
2. If a channel's liquidity falls below a threshold, find channels with excess liquidity.
3. Transfer liquidity from less active channels to the low-liquidity channel.
4. Generate a cryptographic proof to ensure balance consistency.
```

This mechanism reduces the need for frequent on-chain settlements, making the system more efficient and scalable (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)..

3. Incentive Structures

Overpass Channels incentivizes participants through a well-balanced reward system. Participants, including LPs, validators, and off-chain storage nodes, are compensated for their contributions to the network.

- **LP Rewards**: Liquidity providers earn fees based on the trading volume in the pools they support.
- **Validator Compensation**: Validators that process on-chain transactions are compensated with a portion of the transaction fees.
- **Storage Node Rewards**: Off-chain storage nodes are responsible for storing channel states. They are compensated for ensuring the security and availability of channel data.

```
Fee Distribution:

1. Liquidity Providers → Earn trading fees based on pool contribution.
2. Validators → Compensated for verifying on-chain transactions.
3. Off-Chain Storage Nodes → Paid for maintaining channel state security.
```

This incentive model ensures that key actors within the network are rewarded for maintaining the system's integrity and efficiency (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)..

Future Research and Challenges

1. Post-Quantum Security

As mentioned earlier, the current cryptographic methods used in zk-SNARKs could be vulnerable to quantum computing advances. The system relies on cryptographic assumptions such as the **discrete logarithm problem**, which may become solvable using quantum algorithms like **Shor's Algorithm**.

Future research aims to develop **post-quantum secure** zk-SNARKs, which will provide the same level of security even in a post-quantum computing world. Possible directions include **lattice-based cryptography**, which is considered resistant to quantum attacks(Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

2. Privacy-Preserving Analytics

Overpass Channels offers strong privacy guarantees by using zk-SNARKs. However, this can make it difficult to gather system-wide analytics for performance optimization or issue detection.

Research into **privacy-preserving data analysis techniques** such as **secure multi-party computation** and **differential privacy** could help balance the need for analytics with the privacy protections built into the system. These methods would allow the network to generate useful insights without compromising the privacy of individual users (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

Circuit Implementation in Overpass Channels

Overpass Channels leverages hierarchical circuits to manage transactions, state updates, and global consistency. These circuits include processes for channel verification, intermediate contract functions, and shard state updates, ensuring that the system operates efficiently and securely across all layers.

1. Channel State Verification for Both Channels

This circuit verifies the state of both channels involved in a transaction to ensure data consistency and integrity. Each channel maintains a **Merkle root** reflecting its current state.

Algorithm: Channel State Verification

1. **Inputs**:
    - Source Channel State S_{source}
    - Destination Channel State $S_{\text{destination}}$
    - Merkle Proofs for both channels $\pi_{\text{Merkle source}}, \pi_{\text{Merkle destination}}$
    - Current Shard Merkle Root R_{M_shard}
2. **Steps**:
    - Verify the Merkle proofs of both the source and destination channels using the shard Merkle root.
    - Ensure that the source channel has sufficient funds for the transaction.
    - Update the balances of both the source and destination channels.
    - Generate a zk-SNARK proof $\pi_{\text{cross channel}}$ to verify the transaction.
    - Compute the new shard Merkle root $R_{M\_shard\_new}$.

This ensures the state consistency of both channels during transactions, preventing any inconsistencies in cross-channel transfers.

2. Intermediate Contract Circuits

Intermediate contracts handle the aggregation and verification of multiple state transitions within a shard. These circuits ensure that the overall system remains efficient and scalable as transactions grow.

Algorithm: Rebalancing Circuit

1. **Inputs**:
    - Channel Identifier
    - Current and Desired Balance Distributions
    - Rebalancing Amount $\Delta_{\text{rebalance}}$
    - Signatures from both parties
    - Shard Merkle Root R_{M_shard}
2. **Steps**:
    - PoseidonHash the balances before and after rebalancing to ensure integrity.
    - Verify balance conservation.

- Update the shard's Merkle tree with the new state.
- Generate a zk-SNARK proof $\pi_{\text{rebalance}}$ to confirm rebalancing and agreement.

This circuit ensures that liquidity within channels can be adjusted dynamically, reducing the risk of bottlenecks without requiring frequent on-chain updates (Overpass Channels Technical Paper (https://cryptskii.github.io/Overpass_Channels_Technical_Paper.pdf)).

3. Global State Update Circuit

This circuit is responsible for updating the global Merkle root, aggregating changes across all shards, and verifying consistency across the system.

Algorithm: Global State Update Circuit

1. **Inputs**:
   - Current Global Merkle Root
   - Shard Merkle Roots and Proofs
   - Aggregated State Update Proofs
2. **Steps**:
   - Verify the Merkle roots and proofs for each shard.
   - Check the consistency of the balances across all shards.
   - Generate a recursive zk-SNARK proof to validate all updates.
   - Update the global Merkle root to reflect the latest state.

This ensures that global state transitions are consistent, preventing double-spending and ensuring the security of cross-shard operations.

Decentralized Exchange (DEX) on Overpass Channels

Overpass Channels can also facilitate decentralized exchanges (DEX) by managing liquidity pools, order matching, and secure trading. The DEX uses zk-SNARKs to ensure that all trades are processed privately and securely.

Overview of the System Components

1. **On-Chain Hub Contract**: Manages liquidity pools and order execution.
2. **Off-Chain Router**: Matches orders and updates the order book.
3. **Sparse Merkle Trees (SMTs)**: Used for tracking order states and ensuring that the order book remains tamper-proof.
4. **zk-SNARK Integration**: Verifies that all trades adhere to the correct rules without exposing sensitive trading information.

This allows traders to enjoy the speed, orderbook, and all other features of a centralized exchange, while benefiting from the privacy and decentralization offered by zk-SNARKs.

Conclusion

Overpass Channels represent a significant leap forward in blockchain scalability, privacy, and censorship resistance. By leveraging advanced cryptographic techniques such as **zk-SNARKs** and off-chain processing, it achieves high throughput and privacy while eliminating the need for traditional consensus mechanisms like validators or miners. The integration with the **TON blockchain** allows for seamless cross-shard communication and maximizes efficiency through its sharding architecture.

Key Takeaways:

1. **Scalability**: The system scales linearly with the number of channels, ensuring that the overall transaction throughput increases as more users and channels are added.
2. **Privacy**: zk-SNARKs provide robust privacy by ensuring that transaction data remains confidential while still allowing for valid proof verification.
3. **Censorship Resistance**: The decentralized architecture makes it computationally and economically infeasible for any single entity to censor or block transactions.
4. **Use Cases**: The platform supports diverse applications, including global payments, secure healthcare records management, and confidential voting systems, all while maintaining privacy and compliance with necessary regulations.
5. **Post-Quantum Security and Future Research**: As quantum computing advances, ongoing research will focus on developing post-quantum zk-SNARK systems and privacy-preserving analytics to ensure long-term security and usability.

By combining these technical innovations, Overpass Channels offer a scalable and privacy-enhanced solution for the future of decentralized finance and blockchain applications .

Bibliography

Here are the key references that the Overpass Channels system draws upon for its foundational cryptographic and blockchain technologies:

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)
2. **Buterin, V., & Poon, J. (2017)**. *Plasma: Scalable Autonomous Smart Contracts*. Plasma Whitepaper (https://plasma.io/plasma.pdf)
3. **Raiden Network Team (2017)**. *Raiden Network: Fast, Cheap, Scalable Token Transfers for Ethereum*.
4. **Celer Network (2019)**. *Celer Network: Bring Internet Scale to Every Blockchain*. Celer Network Whitepaper (https://www.celer.network/doc/CelerNetwork-Whitepaper.pdf)
5. **PLONK Documentation (n.d.)**. *ZK-SNARKs: PLONK*. PLONK Docs (https://docs.plonk.cafe/)
6. **Ben-Sasson, E., Chiesa, A., Tromer, E., & Virza, M. (2014)**. *Scalable Zero-Knowledge via Cycles of Elliptic Curves*. *International Cryptology Conference*.
7. **Groth, J. (2016)**. *On the Size of Pairing-Based Non-Interactive Arguments*. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*.
8. **The Open Network (2021)**. *The Open Network Whitepaper*. TON Whitepaper (https://ton.org/whitepaper.pdf).