PegaSys Litepaper

Earn native yield on your idle Bitcoin

Ankur Dubey Shivaansh Kapoor Aman Raj

September 17, 2024

Contents

1 Abstract

Fifteen years ago, Bitcoin[1] showed the world how to expend a physical commodity to create native value on the internet. Since then, Bitcoin has established itself as the flagship store of value for the digital asset landscape. Currently, Bitcoin has a market capitalization of over a trillion dollars, accounting for more than half the total value of the entire cryptocurrency market. Despite the immense amount of monetary value locked on the Bitcoin blockchain, Bitcoin holders lack a reliable means to generate passive yield on their idle coins. We present PegaSys, a non-custodial protocol for earning native interest on your Bitcoin. The PegaSys network aggregates liquidity from depositors to operate a liquidity hub on the Lightning Network[2], accruing fees on every transaction routed through it and providing inbound liquidity at a price. The user can withdraw their share of the protocol's value on-demand. In this paper, we delineate the motivations behind PegaSys along with a detailed system architecture and economic model.

2 Lack of Native Yield

Today, there are three methods to earn Bitcoin - excluding custodial services. The first involves consuming energy to mint new coins, while the second and third involve deploying on-chain capital to earn yield:

1. **Block Mining**
   The original method for earning bitcoin was block mining. Mining is the act of creating a block of transactions, producing a corresponding proof of work, and broadcasting it to the network in hopes that other nodes continue mining blocks on top of yours. This process results in the miner profiting from a predetermined block subsidy along with transaction fees. Unlike the other two methods, this requires a sizable investment in hardware and energy and doesn't depend on on-chain capital. Additionally, if you are not a "large" miner it is extremely likely that you will experience a high degree of variance in your earnings. To circumvent this, you can join a mining pool, responsible for aggregating energy and hardware to make it more likely to mine a block.

2. **Lightning Nodes**
   Bitcoin has a couple of inherent limitations which make it infeasible to use as a payment layer. First, its limited block size results in high transaction fees, making it illogical to perform low-value or high-frequency transactions. Second, its reliance on probabilistic finality requires approximately an hour (6 block confirmations) to be confident that the transaction won't be reverted. The Lightning Network solves this by allowing two users to transact almost instantaneously through payment channels. Payment channels are open and closed on-chain, but "transacted through" by exchanging off-chain commitments to their most recent state. If a user doesn't have a direct channel with the intended recipient, they can route the

payment through a series of channel "hops". These "hops" are facilitated by routing nodes who charge a fee proportional to the amount of Bitcoin being transferred through them. Although Lightning provides an elegant solution to the problems posed by Bitcoin, it also comes with its own limitations. On the operations side, node operators need to be able to run lightning nodes and make sure they are available for payment routing. On the liquidity management end, node operators must ensure capital efficiency for payment channels, periodic rebalancement of channels, and sufficient procurement of inbound liquidity to enable outbound payments. We will discuss these in more depth in this section.

3. **Babylon**
   More recently, Babylon[3] has emerged as a means of earning yield by staking one's Bitcoin to secure other blockchains. Babylon presents a graceful solution which overcomes Bitcoin's limited programmability by using EOTS (extractable one-time signatures) to enforce slashing conditions. Yield generation on Babylon differs from the methods discussed above, as Babylon stakers earn yield in the native currency of the chain they are validating instead of Bitcoin. Stakers also have the option of delegating their stake to a validator on their behalf for a share of their earnings.

While there are multiple avenues for generating yield on Bitcoin, each comes with its own set of trade-offs, often requiring significant technical expertise, sizable capital investment, or acceptance of yield in non-Bitcoin assets. Also note that all methods besides delegated staking on Babylon require active participation by the user, making them unappealing to the masses.

3 PegaSys: A Symbiotic Protocol

3.1 Overview

PegaSys's principle vision is to allow Bitcoin holders to earn native yield on their capital in a passive, non-custodial manner. We aim to achieve this by aggregating liquidity from depositors, provisioning it to the Lightning Network in numerous ways, and distributing the accrued fees back to our depositors. By using idle Bitcoin to enable cheaper (less "hops") and higher value (more channel capacity) transactions on Lightning, in addition to providing native yield to depositors, PegaSys is mutually beneficial to Bitcoin and Lightning users.

3.2 Deposits

The only thing a prospective depositor needs to do to start earning yield is to create a Deposit UTXO using a spending script that can be used in one of two ways:

1. **Spending by PegaSys**: Once included in a block, the UTXO can be spent by the PegaSys network using an aggregated Schnorr signature. This mechanism provides flexibility for the PegaSys network to manage funds as needed.

2. **Time-locked Spending by the Depositor**: If 144 blocks (approximately 1 day) have passed since the UTXO was created, the depositor can spend the UTXO using their key. This spending path acts as an escape hatch for depositors if PegaSys hasn't allocated their funds within a day.

Once the deposit UTXO has been added to a block and finalized (received 6 confirmations), the PegaSys network will add the UTXO to its "free liquidity pool" and make it provisionable to the lightning network for earning yield.

3.3 The PegaSys Hub

The PegaSys Hub is the term used to describe the singleton lightning node that is controlled by the network's Schnorr[4] key. Each PegaSys node holds a shard of the network key, and all (Bitcoin and Lightning) transactions done by the system must go through a threshold signature scheme[5] (TSS). The signature threshold is set to $2f + 1$, where f is the number of failures the system can tolerate, to ensure a majority honest approval for every transaction. Once enough nodes have verified and signed the transaction, a single node broadcasts the transaction to the intended recipient. Once signed, each transaction also needs to be added to PegaSys's internal ledger for the bookkeeping of user funds.

3.4 The PegaSys Ledger

The PegaSys ledger is a core component of the network, responsible for recording transactions, maintaining account balances, and ensuring the integrity of the system. Every time the PegaSys Hub generates a valid aggregated signature for a transaction, the transaction will be added to a block's operation log through an additional consensus round. Each block in the ledger consists of the following elements:

1. **Block Number**: The block number provides a logical ordering for all blocks

2. **Account Tree**: A MS-SMT (Merkle Sum Sparse Merkle Tree) [6] that stores every user's current balance (deposit + fees accrued). The root of this tree therefore commits to the total User Deposits D_T as of block time T.

3. **Lightning Channel Commitment Tree**: A Merkle Sum Tree built over an exhaustive list of the Lightning Channel Commitment Transactions describing the latest state of each Lightning Channel the System is a participant of.

   The leaves commit to a lightning channel commitment, while the internal nodes commit to their children as well as the sum of the System's share in each of the Lightning Channels described by each of their children at the block number T. The root of this tree therefore commits to the total amount R_T deployed by the network into all Lightning Channels at T.

4. **Operation Log**: The operation log is a regular Merkle Tree that commits to the list of all actions signed by the network since the last block. This will include

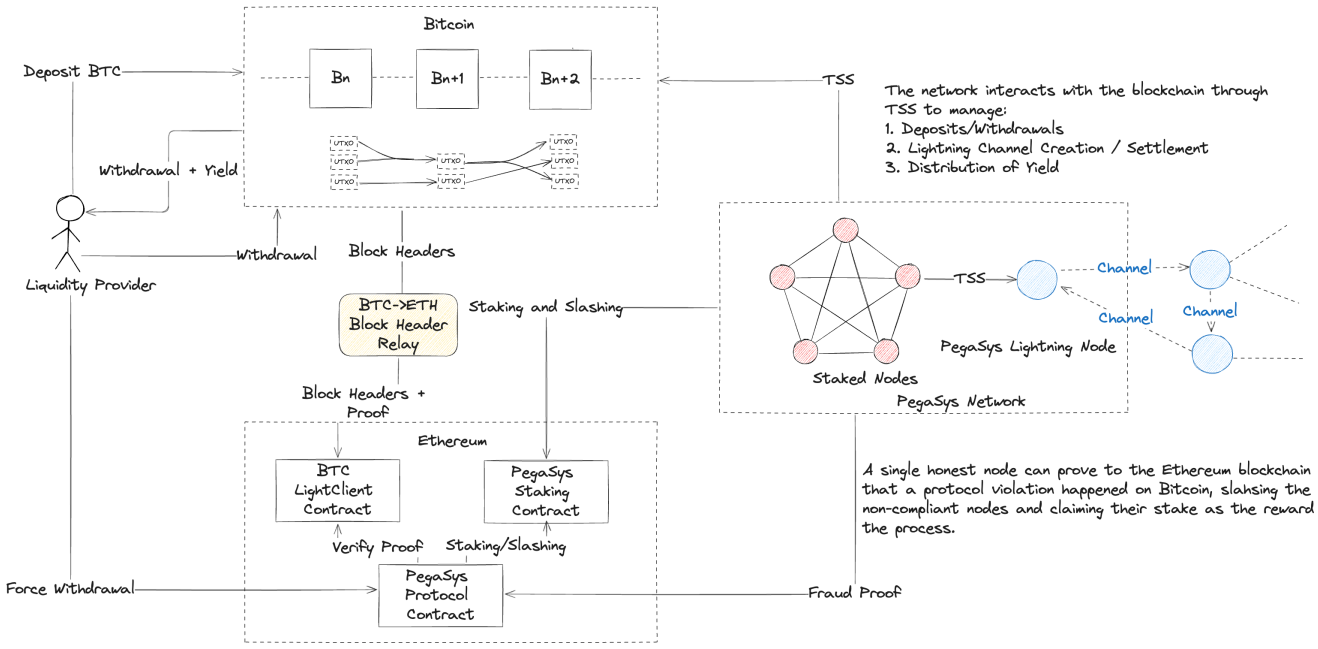

Figure 1: PegaSys Technical Architecture

the actions for channel bootstrap, channel settlement, intermediate routing HTLC commitments and post-routing commitments.

The Merkle Sum Tree structure for the Account Tree and the Commitment Tree is chosen for efficient verification of the solvency of the network, and a fraud-proof construction based on this data is described in this section.

3.5 Withdrawals

PegaSys employs an optimistic withdrawal scheme. Initially, the user sends a withdrawal request to at least $f+1$ PegaSys nodes - guaranteeing that at least 1 honest node receives the request and broadcasts it to the rest of the network. Then, the system reaches consensus on how to compose a withdrawal payment for the user's current balance in the Account Tree. Once these actions are completed, the system creates a transaction with a UTXO spendable by the user's public key. In the case that the network does not comply to the user's request, the user can initiate a forced withdrawal by appealing to a "higher" authority. Read more about this in this section.

4 Economic Security

All operations performed by the network such as processing deposits, withdrawals and deployment of user funds into lightning channels require consensus. While a system of

$3f + 1$ nodes operating under partial synchrony can tolerate f Byzantine failures, this guarantee is insufficient for a network of nodes managing all of the user's deposits.

To achieve full collusion resistance we introduce a fraud-proof mechanism for a single honest party to objectively prove misconduct by a dishonest majority and slash their stake. As discussed in section 4.1.1, the party submitting a valid fraud-proof receives all of the slashed staked, enabling quadratic scaling of safely manageable TVL with the total network stake.

It immediately follows that the participants in the protocol must be slashable by a "higher authority" such as a Layer 1 blockchain with significantly more economic security. While it would be ideal to build such a system on the Bitcoin Chain, it is non-trivial due to the limited execution environment and stateless nature of the Bitcoin Scripting System.

The next best candidate for a chain hosting the proof verification environment would be Ethereum[7]. Assuming the existence of an independent and trustless relay like BTC-Relay[8] submitting block headers from the Bitcoin Blockchain to Ethereum, arbitrary state transitions on the Bitcoin Blockchain can be proven on the Ethereum blockchain.

This paper assumes the existence of such a Trustless Relay and an accompanying on-chain BTC Light Client Contract for its fraud-proof constructions.

4.1 Staking

The security of the PegaSys network is derived from Ethereum. To participate in the PegaSys network, each node must stake wBTC in an Ethereum staking contract. The staked wBTC tokens are subject to slashing in the case that the node exhibits provably malicious behaviour. As demonstrated in the following section, the TVL safely manageable by the network is proportional to the square of the total stake. Therefore, in order to prevent market fluctuations from affecting this ratio, the asset staked is wBTC.

The network is designed to withstand up to f faulty nodes out of a total of $3f + 1$ nodes, meaning it can function correctly as long as fewer than one-third (approximately 33%) of the nodes are faulty. However, if more than one-third of the nodes collude, the network's integrity may be compromised, and our users' funds will be at risk. Since this outcome is unacceptable, the system incorporates a game theory-based mechanism which can tolerate collusion of any size in the presence of at least one honest node.

4.1.1 Quadratic Staking Impact

This mechanism incentivizes nodes to detect and report collusion by offering them all the stake in the system as a reward for reporting an offense. The goal is to make it more profitable for at least one node to act honestly than for anyone to collude, thereby making collusion an irrational choice for any node.

Consider a system with N nodes, where each node must stake a fixed amount S of wBTC to participate. The total staked amount across the network would then be $N \cdot S$ wBTC. If the protocol allows a single honest party to provably report misconduct by a

malicious majority and claim their stake, the cost of bribing a single node is $(N-1)S$. It follows that the cost of bribing the whole system is $N(N-1)S$, ensuring that the system can safely control $\mathcal{O}(SN^2)$ TVL. This is an **imperative** property since the network should control more in TVL than its total stake, otherwise is no incentive to be a part of the protocol.

4.2 Slashing

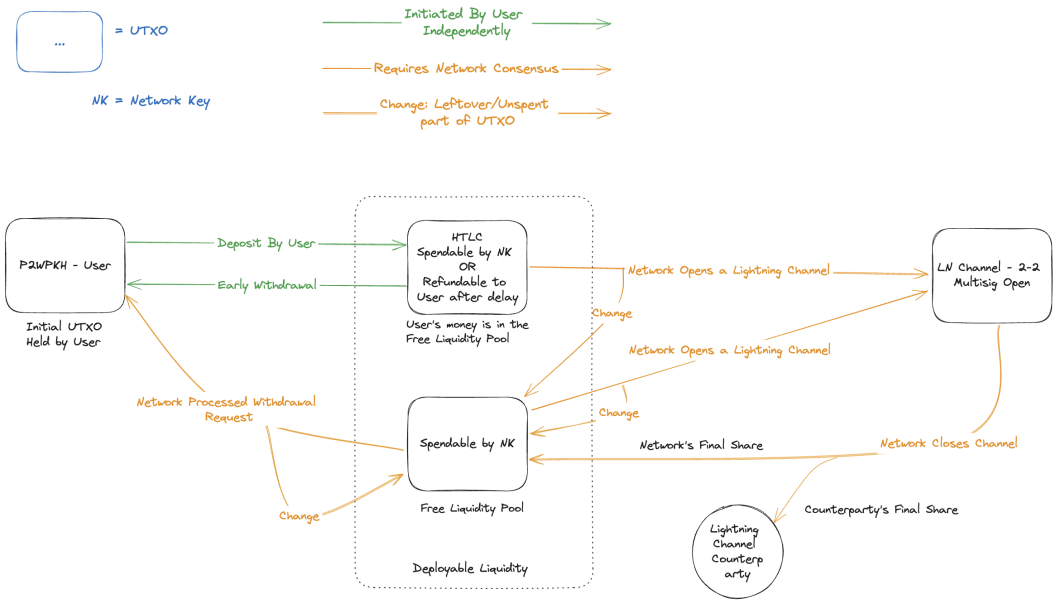

Figure 2: All Legal Fund Movement Paths by the System on Bitcoin

The approach we propose for this paper is to have the slashing condition verified and executed on the Ethereum blockchain. This introduces the following constraints into the system:

- Active participants in the protocol such as node operators must stake on the Ethereum blockchain. This allows the fraud-proof validation logic to slash offending nodes without dependence on an external system. Note that BTC liquidity providers will not need to interact with the Ethereum Blockchain for deposits and claiming yield unless the majority is censoring their requests.
- The Ethereum contract must have a source of truth for the transactions performed on the Bitcoin blockchain in a trust-minimised fashion. We assume the existence of a service posting Bitcoin block headers to an on-chain light-client contract on Ethereum and design the fraud-proof mechanism such that it includes the Merkle proof for offending transactions verifiable against corresponding block headers.

- The fraud proofs must be submitted to the Ethereum blockchain by the prover.

4.2.1 Proof of Solvency

Broadly speaking, the network is responsible for the following operations:

- Deployment of funds from the Free Liquidity pool (henceforth referred to as FL pool) into a lightning channel.
- Settling and Closing an existing Lightning Channel on the Bitcoin chain as determined by the global algorithm. The network's share of the closed Lightning Channel must return to the FL pool.
- Processing deposits and withdrawals of User funds from the FL pool.

As long as an honest majority exists and the protocol is being followed, the lightning network routing protocol will guarantee the **solvency** of the network. Whenever a payment of amount A is routed through a network-owned lightning channel, it receives an amount $A + f$ from an incoming channel and forwards A through an outbound channel, generating f as the revenue. It therefore follows that if the network at any point in time becomes insolvent, the majority has acted dishonestly and must be slashed.

In the following sections, we describe an interactive protocol for a single honest party to prove such a protocol violation to the Ethereum blockchain and have a dishonest majority be slashed.

4.2.2 Observing the Current State of the Lightning Channels

For a given lightning channel in which the protocol is one of the participants, its state at any point in time can be observed by looking at its latest **Lightning Channel Commitment Transaction**, a Transaction as defined by the Lightning Network Protocol describing the fund distribution of each party in the channel at that point of time and executable instructions for realizing the mentioned distribution by settling the channel on-chain.

Let the number of Lightning Channels in which a network is a participant at time t be $N(t)$. For the ith channel, let $A_N(i,t)$ and $A_C(i,t)$ represent the balances owned by the network and the counterparty respectively. Then, $LC(i,t)$, the latest signed Lightning Channel Commitment Transaction for the $i$the channel owned by the network at the time t be defined as:

$$LC_i^t = \big(A_N(i,t), A_C(i,t), Sig_N(i,t), Sig_C(i,t)\big)$$

where Sig_k represent the signature by party k.

A snapshot S_t of all lightning network channels owned by the network at a given time can be defined as:

$$S_t = \big\{LC_i^t | i \in N(t)\big\}$$

The total share of the liquidity owned by the protocol across all channels the network is a counterparty in can be defined as:

$$R_t = \sum_{LC(i,t) \in S_t} A_N(i,t)$$

The solvency criterion for the protocol can then be defined as

$$\sum_{d \in D_t} d \leq R_t$$

where D_t is the set of all user deposits that have not been withdrawn at the time t.

4.2.3 Preventing a Dishonest Majority from Censoring Fraud Proofs

If we assume an honest majority, the rules of the consensus will prevent the network from being dishonest since all operations including opening and settling lightning channels must go through network consensus.

However, this does not prevent the network itself from colluding and stealing the user's deposits by using the user's deposits to create a lightning channel and send the funds to an arbitrary address. If this channel commitment were to be reported to the protocol and be recorded in the ledger, the protocol would become provably insolvent and be liable to be slashed. Instead, it would be in the interest of the colluding majority to censor this commitment and prevent it from being included in the internal ledger.

To prevent this situation, we empower a single honest node to prove the non-inclusion of a commitment transaction on Ethereum and have the colluding majority slashed in the process.

This is achieved as follows:

1. The network is required to publish a state commitment ("state root") to the state of the internal ledger on the Ethereum blockchain, periodically. This acts as the "source of truth" to the ledger's state on Ethereum and can be used to build and verify proof that asserts the existence or absence of certain keys in it. A liveness failure for posting the state root is also a slashable offence.

2. Given that all actions in the network must go through consensus, it can be reasonably assumed that a majority of the honest nodes must have at some point access to every message signed by the network.

3. If $LC(i, t_i)$ is signed by the network *but not included in the ledger*, a single honest node can submit proof to the Ethereum blockchain asserting the non-inclusion of $LC(i, t_i)$ in all prior state-commitments published to Ethereum after $t \geq t_i + delay$. Such a proof can be constructed as a Zero Knowledge Proof to reduce the cost of verification, and for shielding the commitment for reasons described in a later section.

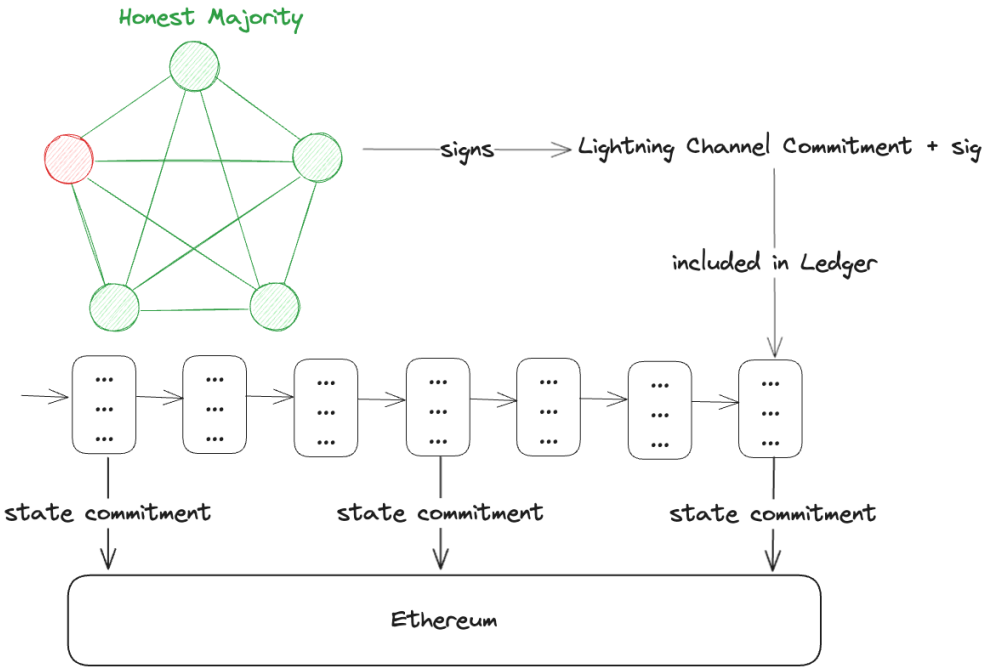

Figure 3: Normal Operation of the Protocol

With this construction, it becomes possible for a single honest party to provably slash a dishonest majority, a key condition for quadratic scaling of safely manageable TVL with the total protocol stake.

4.2.4 The Need for Shielding Lightning Commitments

A Lightning Channel commitment is a signed Bitcoin Transaction with the current state of the channel, that can be broadcast to settle the channel on-chain and claim the funds. Lightning channels are not settled frequently transactions are typically just a message exchange between the transacting parties. Due to this, there is a possibility of a malicious counterparty settling the channel with an older commitment.

The Lightning Channel protocol has a mechanism to deal with this scenario. I'll not go into detail about the actual mechanism, but what is relevant to the protocol is that if either party attempts to settle the balances with a penultimate or older commitment, the other party can claim all the funds in the channel including the offending party's initial balance.

This is relevant to the network slashing protocol because submitting a commitment to the Ethereum blockchain broadcasts it to the whole world, giving the ability to a malicious un-staked third party to broadcast the commitment to the Bitcoin blockchain. If this is not the latest commitment of the channel, the counterparty will initiate the recovery protocol and claim all the funds in the channel, causing the protocol to lose capital and become insolvent.

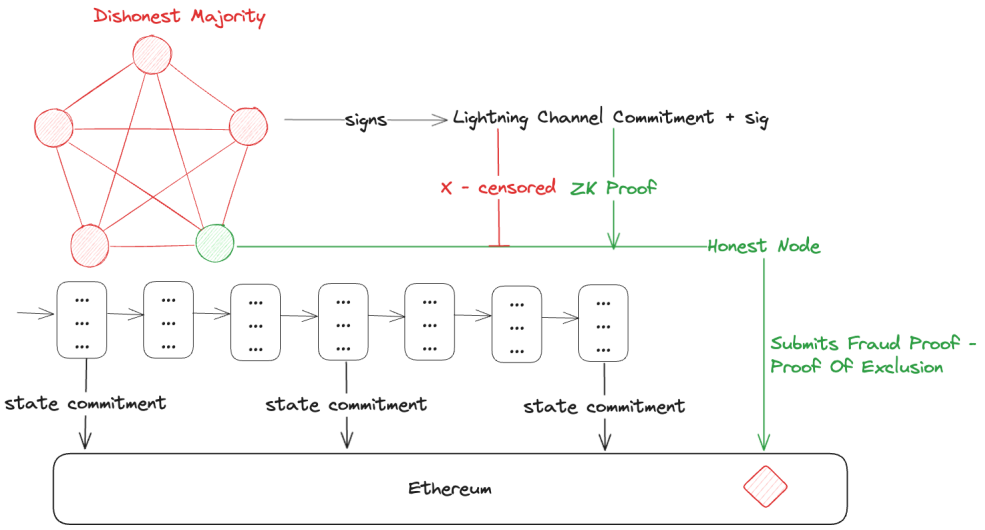

Figure 4: Submission of Fraud Proof for Censorship of Commitment

A solution to this problem is to instead wrap the commitment in a zero-knowledge proof so that the prover can prove to Ethereum that it has the knowledge of a majority signed commitment (without revealing the commitment) that was not included in the ledger.

4.2.5 Guaranteed Withdrawals

Another challenge is to ensure that the network does not censor valid withdrawal requests from liquidity providers. When a user raises a withdrawal request, if their withdrawal is being censored they must submit the request directly to the Ethereum Blockchain, along with proof of the deposit being performed on the Bitcoin chain. The contract on Ethereum verifies the deposit against the block headers received from the Bitcoin block header relay to the on-chain BTC Light-Client Contract.

Once the withdrawal request has been verified on Ethereum, the Network must process the withdrawal and produce proof of the withdrawal being processed within some pre-defined time T_{wd} of submitting the withdrawal request, otherwise, the network is liable to be slashed. This proof is again verified against the Bitcoin block headers received from the relay.

5 Yield Generation

5.1 The Market

Just like in any market, the market for liquidity on the Lightning Network has both a supply and demand side. The supply side consists of Lightning Network users who have excess liquidity and are willing to open channels with other nodes to facilitate payment

routing and, in return, earn fees. These users, often referred to as "liquidity providers", play a crucial role in maintaining the efficiency and functionality of the network by ensuring that sufficient liquidity is available where needed.

Lightning channels are classified as either inbound or outbound:

- **Inbound Channels** represent the capacity to receive payments from other nodes. Inbound liquidity is crucial for users or services that expect to receive payments, as it ensures that other nodes can send funds to them.
- **Outbound Channels**, on the other hand, represent the ability of a node to send payments to other nodes. When a node has outbound liquidity, it means it can initiate transactions and transfer funds out to others on the network.

On the demand side, some users require liquidity in their channels to be able to send payments. These users need sufficient outbound liquidity in their channels to complete transactions. However, if a node expects to receive payments frequently, it must ensure adequate inbound liquidity. These users might operate businesses, run services, or engage in other activities that necessitate frequent inbound or outbound transactions on the Lightning Network. To ensure their channels have enough capacity, these users are willing to pay fees to liquidity providers in exchange for access to the required liquidity, be it inbound or outbound.

The interplay between these two sides creates a dynamic market where liquidity is continuously reallocated across the network in response to shifting demand patterns. The equilibrium price of liquidity—determined by the fees set by liquidity providers—reflects the underlying supply and demand conditions. The PegaSys protocol positions itself within this market by acting as a large-scale liquidity provider and payment router, aggregating funds from multiple depositors to maximize fee earnings and, consequently, yield generation for its users. By carefully managing both inbound and outbound channels, PegaSys ensures optimal allocation of liquidity to areas of highest demand, thereby enhancing the yield potential for depositors.

5.2 Lightning Channels

The most basic way of earning fees on the Lightning Network is by participating as an intermediary routing node. The fee consists of a base fee for every transaction routed and a fee per Satoshi routed. In October of 2023, River Financial published a comprehensive report[9] highlighting the immense growth exhibited by the Lightning Network in the last couple of years. Lightning saw a 1,212% increase in transactions from August 2021 to 2023. In August 2023 alone, the Lightning Network was estimated to have processed 6.6 million monthly transactions, with a total value of $78.2 million. The report also mentions that River Financial was earning approximately 1% annual yield on their Lightning capacity around that time. This amount may seem modest compared to other financial instruments, but it is important to consider the low-risk nature of Lightning Network operations, and the potential for higher returns as the

network continues to grow and mature. Lightning channels also don't have any sort of lock time or maturity date, so they are perpetually open until closed by either party. This is very useful for PegaSys, as we can source liquidity to fulfill withdrawals on an as-needed basis by closing channels. In the context of PegaSys, this yield generation tactic will only be used for a fraction of the protocol's reserves in order to make sure that the system is poised to be able to handle a sudden influx in withdrawals. The bulk of the protocol's yield is going to come from more lucrative methods discussed below.

5.3 Lightning Pool

Lightning Pool[10] is a non-custodial channel lease marketplace (CLM) that leverages modern auction theory to efficiently allocate inbound channel liquidity within the Lightning Network. The marketplace allows participants to buy and sell channel capacity, specifically through a financial instrument known as a Lightning Channel Lease (LCL).

An LCL operates similarly to a traditional bond: one party commits capital to another for productive use within the Lightning Network, and in return, the lender is compensated for their opportunity cost. However, unlike traditional bonds, the capital within an LCL is locked into the Lightning Network for a set duration, enabling specific routing and transaction functionalities during that period. The commitment to a particular duration is enforced on-chain through Bitcoin Script, ensuring that the leased channels remain open for the agreed-upon time frame, thus providing stability and predictability in liquidity allocation.

For PegaSys, Lightning Pool presents a strategic opportunity to maximize yield generation on the Lightning. The lightning pool sends us the most lucrative demand signals, and the protocol decides whether or not to meet the demand with our free liquidity. The fixed-term nature of LCLs aligns well with PegaSys's liquidity management strategy, allowing the protocol to lock in predictably high returns through the pool while always having a liquid reserve in basic lightning channels to pay back our depositors on demand.

Unfortunately, Lightning Pool doesn't have any publicly available figures for historical orders, APYs, etc. However, we can extrapolate these numbers from another similar liquidity marketplace on Lightning called Amboss[11]. Amboss claims that the average APY for providing liquidity is north of 5%. Keep in mind that this is the average, and having aggregated liquidity will allow PegaSys to capitalize on the most lucrative opportunities in the market.

6 Future Directions

6.1 Taproot Assets

With the Taproot upgrade introduced as a part of BIP341[12], a new class of assets called Taproot Assets[13] has emerged to enable the native issuance of Assets on Bitcoin. Taproot assets enable high-volume, near-instant, and low-cost asset transfers through the lightning network. More importantly, Taproot Assets are compatible with

the Lightning network out of the box, allowing a future version to add support for these assets with minimal changes to the protocol.

Integration of Taproot Assets into the protocol would allow Liquidity Providers to natively earn yield in several assets apart from Bitcoin, adding more revenue streams to the protocol and improving capital efficiency by utilizing multi-hop routing with heterogeneous channels.

6.2 Sharding

Various estimates suggest the lightning network can scale up to 1 million transactions per second across the entire network. To generate the maximum yield, our network must be centrally positioned in the lightning network to capture the maximum traffic and routing fee. With this, it is fair to assume that our network would need to scale up as demand for the network throughput increases.

In the current network design, all transactions (on-chain and off-chain) must go through network consensus, which significantly limits the network throughput. One way to address this issue is for the network to operate in shards, where each shard is a sub-network of $n \geq 4$ (single fault tolerant) Pegasys nodes controlling a lightning node. Each shard can independently reach a consensus and route transactions at the cost of decreased capital efficiency, increased fragmentation and overall linear decrease in the TVL safely controllable by the network stake. A suitable shard size can be found for an optimal balance between the different tradeoffs.

6.3 Governance

The PegaSys Network governance model is designed to ensure transparency, security, and decentralized decision-making across the network. The governance structure includes the following components:

- **Node Operators:** Participants who operate the nodes in the network, stake wBTC. They have voting rights proportional to their stake.
- **Community Members:** Users of the network who can propose and vote on protocol changes, submit challenges, and engage in governance discussions. Voting weight is determined by their activity and contribution to the network.
- **Governance Council:** A multi-signature wallet controlled by key stakeholders (node operators, developers, community representatives) responsible for making high-level decisions, including protocol upgrades and dispute resolutions.

6.3.1 Smart Contract Upgradability

The smart contracts acting as the "higher" authority for the PegaSys Network are designed to be upgradeable, ensuring the network can evolve while maintaining security and trust.

- **Proxy Contract:** The network uses a proxy contract that delegates calls to an implementation contract. The proxy can be upgraded to point to a new implementation contract, allowing for seamless updates without interrupting the network's operations.
- **Upgrade Proposals:** Proposals for contract upgrades must undergo the standard governance process. Once approved, the Governance Council initiates the upgrade by executing a transaction that points the proxy to the new implementation contract.

6.3.2 Yield Distribution

A portion of the yield generated by the network is redistributed to the node operators in order to compensate them. The exact portion of total fees to be allocated to operators can be left to governance.

- **Proposal Submission:** Any participant can submit a proposal for how yield should be distributed. This can include proposals for node operator rewards, community incentives, or funding for network development.
- **Voting Process:** The proposal is reviewed and voted on by Node Operators and Community Members. The voting process follows the same rules as for smart contract upgrades, ensuring that the distribution is aligned with the network's goals and the community's interests.

References

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf.

[2] Thaddeus Dryja Joseph Poon. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. URL: https://lightning.network/lightning-network-paper.pdf.

[3] The Babylon Team. *Bitcoin Staking: Unlocking 21M Bitcoins to Secure the Proof-of-Stake Economy*. URL: https://docs.babylonchain.io/papers/btc_staking_litepaper.pdf.

[4] Tim Ruffing Pieter Wuille Jonas Nick. *BIP-340: Schnorr Signatures for secp256k1*. URL: https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

[5] Yevgeny Zaytman Ahmet Ramazan Agirtas Jorge Arce-Garro and Jelilat Anofiu. *Threshold Signature Schemes*. URL: https://medium.com/nethermind-eth/threshold-signature-schemes-36f40bc42aca.

[6] Mario Oettler. *MS-SMT*. URL: https://blockchain-academy.hs-mittweida.de/courses/bitcoin-taproot/lessons/taproot-assets-2/topic/merkle-sum-sparse-merkle-tree-ms-smt/.

[7] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. URL: https://ethereum.org/en/whitepaper/.

[8] *BTC Relay*. URL: http://btcrelay.org/.

[9] River Financial. *The Lightning Network Grew by 1212 percent in 2 Years .Why It's Time to Pay Attention*. URL: https://river.com/learn/files/river-lightning-report-2023.pdf?ref=blog.river.com.

[10] Lightning Labs. *Lightning Pool*. URL: https://lightning.engineering/pool/.

[11] *Amboss*. URL: https://amboss.space/magma.

[12] Anthony Towns Pieter Wuille Jonas Nick. *Taproot: SegWit version 1 spending rules*. URL: https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[13] Lightning Labs. *Taproot Assets*. URL: https://docs.lightning.engineering/the-lightning-network/taproot-assets.