BitSync: Bitcoin Sidechains with Integrated Security via Real-Time Synchronization

Kobe Koike
kobek@bitsync.dev
www.bitsync.dev

October 15, 2024

Abstract: A purely peer-to-peer sidechain system is proposed to enable direct integration with Bitcoin's security model through real-time synchronization and proof-of-work commitments. By referencing Bitcoin's canonical chain, we present an imaginary sidechain which secures transactions against private forks, ensuring valid withdrawals as well as allowing for modular scalability and efficient transaction processing. This paper outlines mechanisms for secure peg-ins and peg-outs, discusses data availability considerations, and introduces methods to maintain network integrity without compromising decentralization.

1 Introduction

The scalability and programmability limitations of Bitcoin[1] have led to the exploration of sidechains as a means to extend its capabilities without altering the main protocol. Existing solutions often struggle with securely linking sidechain operations to the Bitcoin blockchain, particularly during asset redemption back to the main chain.

We propose an imaginary sidechain architecture, termed **BitSync**, that leverages real-time synchronization with Bitcoin, utilizing proof-of-work commitments to intrinsically link sidechain transactions to Bitcoin's canonical chain. This design prevents private forks from manipulating withdrawals and maintains the integrity of both chains. Focusing on simplicity and unconditional security tied directly to Bitcoin, it offers optional scalability and flexibility features. Given recent advancements such as BitVM[2] and the possibilities to remove private forks through block introspection, we revisit the concept of sidechaining and combine aspects of merged-mining[4](AuxPOW) with it to create a complete modular settlement layer solution BitSync that extends the use case of Bitcoin with full Bitcoin PoW longest-chain security.

2 Unifying Sidechains and AuxPOW

The quest to extend Bitcoin's capabilities has led to various proposals, notably AuxPOW and sidechains. AuxPOW enables miners to use their proof-of-work for multiple chains, but it introduces security vulnerabilities where miners can attack auxiliary chains at minimal cost. Sidechains, on

the other hand, tightly couple their consensus to Bitcoin's, inheriting its security but often facing challenges with slow confirmation times and potential invalid block inclusion.

Our design presents a novel integration of sidechain and AuxPOW concepts, forming a harmonious system that leverages the strengths of both while mitigating their weaknesses. By synchronizing the sidechain with Bitcoin in real-time and utilizing mutual proof-of-work commitments, we create a sidechain that is intrinsically linked to Bitcoin's security without being susceptible to the typical attacks associated with AuxPOW.

2.1 Solving Security Challenges

The Bitcoin-BitSync integration addresses key security concerns. The first is to prevent miner attacks where auxiliary chains can be selfishly mined without giving up their Bitcoin rewards. This design prevents that due to the mutual commitments-Bitcoin blocks referencing sidechain blocks and sidechain blocks referencing Bitcoin blocks.

The second is eliminating private forks which is part of the critical attack surface for interoperability with Bitcoin itself via a trust-minimized bridge. Eliminating the possibility of private forks would enable a fully decentralized bridge with Bitcoin with native Bitcoin longest-chain security.

2.2 Challenges with Traditional Sidechain Designs

Sidechains have long been proposed as a method to extend Bitcoin's functionality without altering the main protocol. However, current implementations also fail to inherit bitcoin's security and/or fail to successfully present decentralization mechanisms. Key concerns include fork handling and invalid blocks. Consider If Bitcoin block B_i references sidechain block SC_i, and the subsequent Bitcoin block B_{i+1} references a conflicting sidechain block SC_i' (a fork), it raises questions about which sidechain block is valid. This issue is exacerbated if SC_i is invalid according to sidechain consensus rules, yet referenced by a valid Bitcoin block.

Security of node consensus as well as light clients is also a concern as the sidechain may end up accepting invalid sidechain blocks without some sort of sybil-attack prevention such as AuxPOW or full validation of state.

Given that we use mutual Proof-of-Work commitments through AuxPOW, the sidechain blocks are validated by ensuring they reference the canonical Bitcoin chain. This eliminates forks and low-cost attacks against various types of network nodes.

We may also leverage the use of zero-knowledge proofs[6] to enhance light-client security and validate in efficient ways. This is especially useful for a trust-minimized bridge between Bitcoin and BitSync.

2.3 Challenges with Traditional AuxPOW

Traditional AuxPOW allows miners to use the same proof-of-work (PoW) to mine on multiple blockchains simultaneously. While efficient, this approach introduces potential vulnerabilities such as low-cost attacks due to the fact that mining rewards of Bitcoin remain unaffected.

This creates a scenario where rational miners might have incentive to destabilize an auxiliary chain for financial gain.

However the use of AuxPOW requires that the sidechain block's proof-of-work satisfies the sidechain's own difficulty level, even when mined alongside Bitcoin's block. This has several benefits in our design:

- **Efficient Proofs for Syncing:** Nodes can sync to the longest chain with the most cumulative work without tracking the full set of Bitcoin headers, expediting the syncing process.
- **Prevention of Attacks:** By ensuring sidechain blocks meet their own difficulty levels, AuxPOW helps prevent attacks such as the injection of fake sidechain blocks or manipulation of consensus. Due to the fact that there can only be one coinbase transaction in Bitcoin and that it contains only unique commitments per chainID, we can secure against private forks. Other designs such as meta-transactions/anchoring or layering on Bitcoin to create off-chain consensus do not protect against the ability to create private forks [7][8][9][10][11].
- **Robustness for Light Nodes and Full Nodes:** Both light and full nodes benefit from AuxPOW's mechanisms, contributing to the overall security and integrity of the sidechain.

The combination of AuxPOW and the sidechain design elements creates a robust design enables incentive for rational miner behavior to mine the sidechain for rewards and build on the longest chain. As committing to any other strategy would not be profitable. Since the sidechain blocks are connected with the Bitcoin main chain the attack costs and security becomes equivalent. See Appendix A for a formal analysis on AuxPOW.

2.4 Eliminating Private Forks with `OP_BLOCKHASH`

To prevent attacks involving private forks and conflicting sidechain blocks we may leverage the use of a theoretical op code such as `OP_BLOCKHASH` as described by the BitVM paper. This would allow us to enforce that when settling on Bitcoin to peg-out from BitSync that neither Bitcoin nor the sidechain are residing on a private fork due to the mutual-chain connection between the chains. Combining the AuxPOW safety checks against duplicate chainID commitments in the Bitcoin coinbase merkle root as well as the enforcement of settlement on the canonical Bitcoin chain result in a reciprocal affect of ensuring the sidechain nor Bitcoin can be privately mined and presented to a BitVM bridge misrepresenting state.

To secure peg-outs, `OP_BLOCKHASH` allows a script to verify the hash of a specific block in the canonical Bitcoin chain. When a user burns coins on BitSync to initiate a withdrawal, they create a Zero-Knowledge Proof (ZKP) that the burn occurred in a valid BitSync block linked to a known Bitcoin block hash as a public witness. BitVM enables the verification of such ZKPs within Bitcoin's existing scripting capabilities.

The peg-out transaction on Bitcoin includes this ZKP and uses `OP_BLOCKHASH` to ensure the referenced block hash (from the public witness) aligns with the canonical Bitcoin chain:

```
<block_height> OP_BLOCKHASH
```

By verifying the block hash within the script, we ensure that the withdrawal is only valid if it references a block in the main Bitcoin chain, thus eliminating the risk of private forks. This method tightly binds the sidechain state to Bitcoin's canonical chain without requiring changes to the consensus rules.

2.5 Demonstrating Enhanced Security

By addressing the standard criticisms, we demonstrate that our design not only meets but exceeds the security expectations for sidechains:

1. **Resolution of Fork Conflicts:** In the event that Bitcoin block B_{i+1} references a conflicting sidechain block SC'_i, sidechain nodes validate both SC_i and SC'_i according to consensus rules. The valid block is accepted, and the invalid one is discarded, regardless of Bitcoin's referencing.

2. **Protection Against Invalid Blocks:** If a sequence of Bitcoin blocks $B_{i+1}, B_{i+2}, \ldots$ reference invalid sidechain blocks, the sidechain nodes will not accept these blocks. The sidechain's consensus remains secure, and the invalid references in Bitcoin do not undermine it.

3. **Robustness of SPV Clients:** With the integration of zero-knowledge proofs, SPV clients on the sidechain can securely verify the validity of sidechain blocks, maintaining security even in the presence of invalid block references in Bitcoin.

4. **Equivalence in Consensus Strength:** Through these mechanisms, we can confidently assert that the sidechain's consensus is as strong as Bitcoin's. Any attack on the sidechain would require resources and efforts equivalent to attacking Bitcoin itself.

3 Bitcoin Integration

3.1 Auxiliary Proof-of-Work (AuxPOW) Integration

Auxiliary Proof-of-Work (AuxPOW) allows BitSync to leverage the proof-of-work done on the Bitcoin blockchain, enabling miners to mine both chains simultaneously without additional computational effort. This integration is achieved through AuxPOW, where the work done on the parent blockchain (Bitcoin) is accepted by the auxiliary blockchain (BitSync) as valid proof-of-work.

3.1.1 Mechanism of AuxPOW

In our design, the following steps are undertaken to implement AuxPOW:

1. **Mining on Bitcoin (Parent Blockchain):**
   - Miners construct a Bitcoin block following all standard consensus rules.
   - In the coinbase transaction's *scriptSig*, miners include:
     - An **AuxPOW Tag**, a predefined magic value (e.g., `0xfa, 0xbe, 'm', 'm'`) indicating the presence of AuxPOW data.
     - A **Merkle Root of Auxiliary Block Headers**, which includes the hash of the BitSync block header $H(SC_i)$.

2. **Constructing the Merkle Tree of Aux Block Headers:**
   - When mining multiple auxiliary chains, miners construct a Merkle tree of all auxiliary block headers.
   - The position of each auxiliary chain in the Merkle tree is determined using a deterministic algorithm based on the chain's identifier and a nonce (merkle_nonce).
   - For BitSync, miners ensure that $H(SC_i)$ is included in this Merkle root.

3. **Including Parent Block Header in BitSync Block:**

- The BitSync block includes the Bitcoin block header B_i as part of its own block header.
- Additionally, the BitSync block contains the AuxPOW proof, which includes:
  - The coinbase transaction from the Bitcoin block.
  - The Merkle branch linking the coinbase transaction to the Bitcoin block's Merkle root.
  - The Merkle branch linking $H(SC_i)$ to the Merkle root of auxiliary block headers in the coinbase *scriptSig*.
  - The Bitcoin block header B_i.

3.1.2 Validation by BitSync Nodes

BitSync nodes perform the following steps to validate an AuxPOW block:

1. **Verify the Bitcoin Block Header:**
   - Confirm that the Bitcoin block header B_i is valid and meets Bitcoin's proof-of-work requirements.
   - Ensure that hash of the Bitcoin block header $H(B_i)$ is part of the Bitcoin canonical chain by checking against known block headers.
2. **Validate the Coinbase Transaction:**
   - Extract the coinbase transaction from the AuxPOW proof.
   - Verify that the coinbase *scriptSig* contains the AuxPOW Tag and the Merkle root of auxiliary block headers.
3. **Verify Merkle Branches:**
   - Validate the Merkle branch linking the coinbase transaction to the Bitcoin block's Merkle root.
   - Validate the Merkle branch linking $H(SC_i)$ to the Merkle root in the coinbase *scriptSig*.
4. **Check Proof-of-Work:**
   - Ensure that the Bitcoin block's proof-of-work meets BitSync's difficulty requirements.
   - Since Bitcoin's difficulty is higher, this condition is typically satisfied.
5. **Confirm Mutual Commitment:**
   - Verify that the BitSync block header includes $H(B_i)$ correctly.
   - This establishes the reciprocal link between the chains.

3.2 Handling Bitcoin Reorganizations in BitSync

Bitcoin reorganizations (reorgs) occur when a longer chain replaces a previously accepted chain, typically due to differences in cumulative proof-of-work. In the context of BitSync, reorgs are naturally handled due to the tight coupling between BitSync and Bitcoin through AuxPOW and block references.

3.2.1 Reorg Process in BitSync

In BitSync, every block is merge-mined with Bitcoin, meaning each BitSync block references a specific Bitcoin block and vice versa. This 1:1 relationship ensures that any reorganization in Bitcoin will directly impact BitSync.

When Bitcoin undergoes a reorg, where some blocks are removed and replaced by a new chain, BitSync will follow Bitcoin's chain and roll back to the last valid Bitcoin block that was merge-mined with BitSync. This ensures that BitSync always follows Bitcoin's canonical chain, and any blocks that are part of a Bitcoin reorg will also be rolled back in BitSync.

3.2.2 Depth of Reorgs and Their Impact

The depth of a Bitcoin reorg will determine how far back BitSync will need to roll back. Since not all Bitcoin blocks will include an AuxPOW commitment to BitSync, only the Bitcoin blocks that do will trigger a rollback in BitSync. For example, if a Bitcoin reorg removes several blocks, BitSync will roll back to the last merge-mined Bitcoin block that referenced a valid BitSync block.

In this way, BitSync guarantees that it follows the canonical Bitcoin chain, even in the event of deep reorgs, ensuring that all BitSync blocks remain valid and part of Bitcoin's longest chain.

3.2.3 Reorg Process in BitSync

In BitSync, each block is merge-mined with Bitcoin, meaning that a valid BitSync block must be tied to a specific Bitcoin block. However, not all Bitcoin blocks will be merge-mined with BitSync.

When a Bitcoin block that is not merge-mined is reorganized out of the chain, it doesn't affect BitSync. In this scenario, BitSync will follow Bitcoin's reorganization but only roll back the blocks that are linked via merge-mining.

Below, we show an example where Bitcoin blocks B_3 and B_4 are reorganized out. B_3 was not merge-mined with BitSync, but B_4, which was merge-mined with BitSync's SC_3, is also removed during the reorg. Bitcoin continues from B_2 (merge-mined with SC_2), and the new canonical Bitcoin tip B_5 (merge-mined with SC_4) is built on top of B_2. BitSync, in turn, reorganizes out SC_3, and SC_4 builds off of SC_2.

In this diagram:

Bitcoin undergoes a reorg where blocks B_3 and B_4 are removed. B_3 was not merge-mined, so it doesn't directly affect BitSync, but B_4, which was merge-mined with SC_3, forces a rollback in BitSync. As a result, SC_3 is reorganized out, and the new SC_4 block is built off of SC_2, just as B_5 builds off of B_2 in Bitcoin.

This process guarantees that BitSync follows Bitcoin's canonical chain and reflects any changes that occur due to reorganization.

3.2.4 Finality in BitSync

While Bitcoin itself does not provide strict finality, in practice, after a sufficient number of blocks have been mined on top of any given block, the probability of a reorganization diminishes significantly. For BitSync, finality is directly linked to Bitcoin's chain.

Since every BitSync block is merge-mined with Bitcoin via AuxPOW, the "finality" of a BitSync block is inherently tied to Bitcoin's reorg depth. Once a certain number of Bitcoin blocks are added on top of a BitSync-linked Bitcoin block, both Bitcoin and BitSync can probabilistically consider the chain to be finalized.

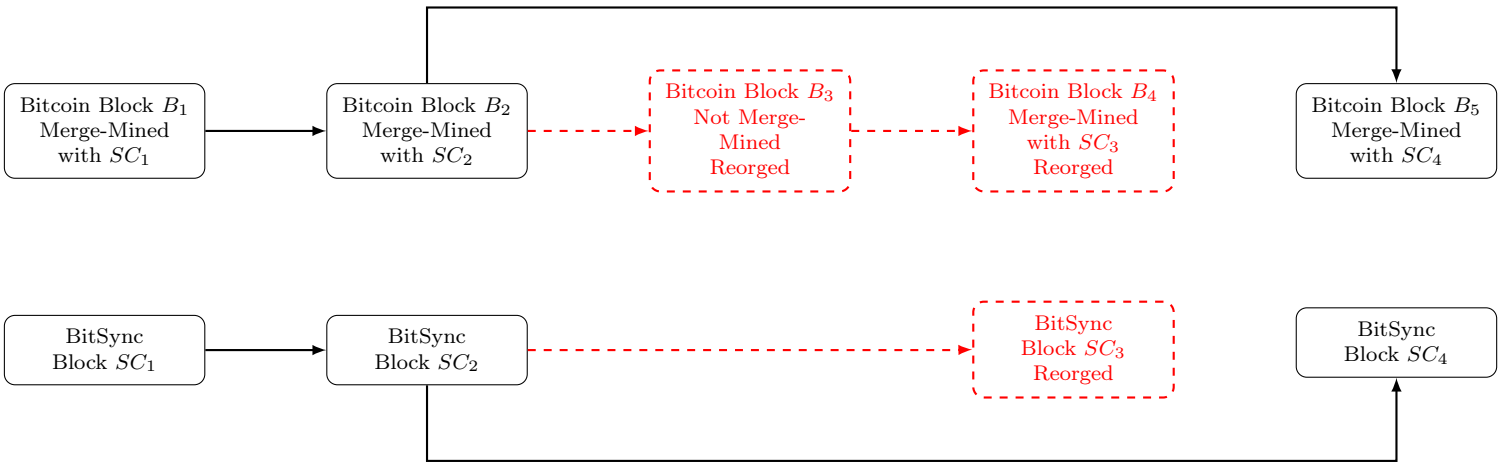

Figure 1: BitSync reorganization following Bitcoin reorg.

In the event of a deep Bitcoin reorg, BitSync will roll back to the last valid merge-mined block and follow the longest Bitcoin chain. Therefore, BitSync's security and finality assumptions are aligned with Bitcoin's.

3.2.5 Implications for Peg-ins and Peg-outs

Since peg-ins and peg-outs in BitSync are linked to Bitcoin blocks, reorgs in Bitcoin will naturally cause BitSync to adjust accordingly. Any pending transactions, such as peg-outs, will be re-evaluated following a reorg, ensuring that only transactions based on the canonical Bitcoin chain are processed. This guarantees that the integrity of the sidechain is maintained even in the face of significant Bitcoin network reorganizations.

4 Secure Redemption Process

To securely withdraw bitcoins from BitSync back to the Bitcoin network, we utilize a non-interactive zero-knowledge proof within an optimistic verification framework such as BitVM. This approach ensures execution safety in a trust-minimized environment, allowing users to redeem assets without intermediaries or interactive protocols.

4.1 Real-Time Synchronization

By synchronizing the sidechain with Bitcoin in real-time, we establish a tight coupling between the two chains. Each BitSync block includes a reference to the latest Bitcoin block hash, and conversely, Bitcoin miners include a commitment to the latest BitSync block hash in their coinbase transactions. This mutual commitment ensures that both chains are aware of each other's state at the time of block creation, preventing manipulation through private forks.

Diagram 1: Reciprocal Commitments between Bitcoin and BitSync

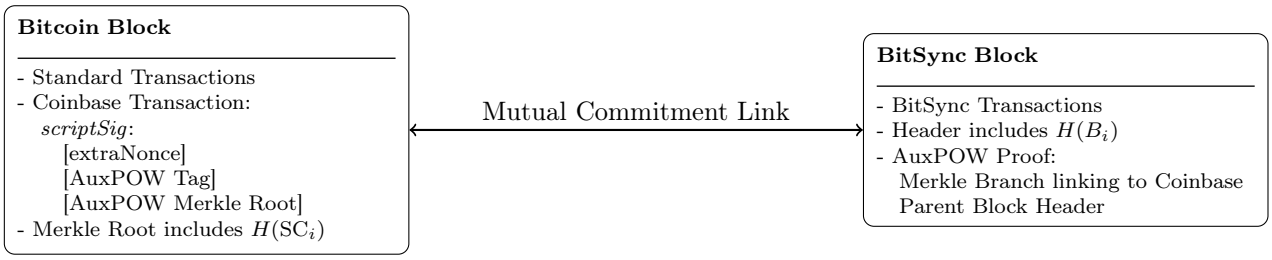

Figure 2: Reciprocal Commitments between Bitcoin and BitSync using AuxPOW

4.2 Withdrawal Mechanism

A user initiates a withdrawal by performing a proof-of-burn on BitSync, calling a burn or locking contract function, effectively removing the tokens from circulation and signaling the intent to withdraw. The burn transaction includes the user's Bitcoin address and is included in a BitSync block that references a specific Bitcoin block.

A non-interactive zero-knowledge proof is generated, demonstrating that the burn transaction occurred in a valid BitSync block linked to the canonical Bitcoin chain. This proof is designed to be verifiable on the Bitcoin network through a system like BitVM.

5 Trust-Minimized Withdrawals Leveraging BitVM

To facilitate secure and efficient withdrawals from BitSync back to the Bitcoin network, we employ an optimistic verification framework inspired by BitVM. This approach enables arbitrary program execution on Bitcoin without altering its consensus rules, allowing us to achieve trust-minimized withdrawals that connect directly to Bitcoin's security model without introducing new security assumptions.

When a user wishes to withdraw bitcoins, they initiate a burn transaction on the BitSync sidechain, effectively removing the tokens from circulation. This transaction includes the user's Bitcoin address and is recorded in a BitSync block that references a specific Bitcoin block hash, establishing a clear linkage between the sidechain and the main chain.

A non-interactive zero-knowledge proof may demonstrate that the burn transaction occurred in a valid BitSync block linked to Bitcoin's canonical chain. This proof is constructed within the BitVM framework, which allows for the verification of arbitrary computations, such as confirming the validity of the burn transaction and its inclusion in the sidechain's state.

By submitting this proof as part of a special transaction on the Bitcoin network, the user invokes a script that verifies the proof's validity and ensures that the referenced block hash corresponds to a block in the canonical Bitcoin chain. This process leverages Bitcoin's existing scripting capabilities, requiring no changes to the consensus rules or additional trust assumptions.

The verification is permissionless; any participant in the network can validate the proof and, if necessary, challenge invalid withdrawals. This mechanism aligns with Bitcoin's security principles,

relying on collective verification rather than trusted intermediaries. It ensures that fraudulent attempts are detected and prevented by the network itself.

By integrating BitVM's capabilities, we reduce on-chain complexity and resource requirements. Under normal circumstances, withdrawals are processed efficiently with minimal on-chain interaction. Only in cases where a dispute arises does additional verification occur, which is essential for maintaining security and integrity.

This approach connects the sidechain directly to Bitcoin's proof-of-work consensus, anchoring the sidechain's state to the main chain without introducing new security assumptions. By leveraging BitVM, we enhance Bitcoin's functionality, enabling trust-minimized withdrawals while preserving the foundational principles of decentralization and trustlessness inherent in the Bitcoin network.

5.1 Verification on Bitcoin Network

Submitting the proof as part of a special transaction on the Bitcoin network, the user includes the zero-knowledge proof and the Bitcoin block hash. The transaction script uses an operation like `OP_BLOCKHASH` to verify that the provided block hash corresponds to a block in the canonical Bitcoin chain.

5.2 Preventing Private Fork Attacks

This method eliminates the possibility of private forks being used to manipulate withdrawals. The proof must reference a block hash recognized by Bitcoin's consensus; any attempt to use a block from a private fork or a less-work chain results in verification failure.

5.3 Outcome of the Withdrawal Process

The withdrawal process results in a binary outcome:

- **Valid Proof:** If the proof is valid and unchallenged, the transaction is processed, and the bitcoins are released to the user's address.
- **Invalid Proof or Challenge:** If the proof is invalid or successfully challenged, the transaction is rejected.

Withdrawal Process Diagram

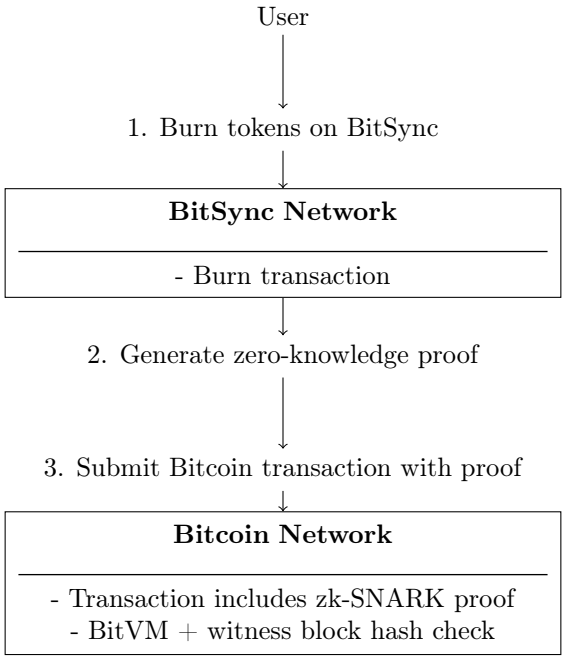

Figure 3: Diagram of the Withdrawal Process

6 Depositing Bitcoins into BitSync

To securely transfer bitcoins into BitSync, we utilize the sidechain nodes' awareness of the Bitcoin blockchain. Users wishing to deposit bitcoins begin by locking their bitcoins on the Bitcoin network, sending them to a special BitVM output script. The funds cannot be spent on the Bitcoin network except under withdrawal conditions described above.

6.1 Creating a Deposit Transaction

After locking the bitcoins, the user creates a deposit transaction on BitSync, referencing the Bitcoin transaction containing the locked funds. This deposit transaction includes a Simplified Payment Verification (SPV) proof of the Bitcoin transaction, allowing BitSync nodes to verify the deposit without downloading the entire Bitcoin blockchain.

6.2 Verification by BitSync Nodes

BitSync nodes validate the deposit transaction through the following steps:

1. **Verify the SPV Proof:** Confirm that the Simplified Payment Verification (SPV) proof provided by the user is valid and that the Bitcoin transaction locking the bitcoins is included in a block on the Bitcoin blockchain.

2. **Check Inclusion in Canonical Chain:** Ensure that the Bitcoin block containing the locking transaction is part of the canonical (longest) proof-of-work chain recognized by the Bitcoin network.

3. **Validate the Merkle Path:** Verify the Merkle branch linking the locking transaction to the block header, ensuring the transaction's integrity and its inclusion in the block.

4. **Confirm Block Header Authenticity:** Check that the block header is valid, adheres to Bitcoin's consensus rules, and is accepted by the majority of the network.

5. **Ensure Funds Are Unspent:** Confirm that the bitcoins locked in the transaction have not been spent elsewhere on the Bitcoin network since the locking transaction.

6.3 Preventing Double-Spending

BitSync effectively prevents double-spending by maintaining synchronization with the Bitcoin blockchain and utilizing SPV proofs. Since BitSync witnesses the canonical chain of Bitcoin and reorganizes with it the reflection of Bitcoin state guarantees the correctness and security of the SPV proof which would recognize and allow for the bitcoin deposit into BitSync.

Deposit Process Diagram

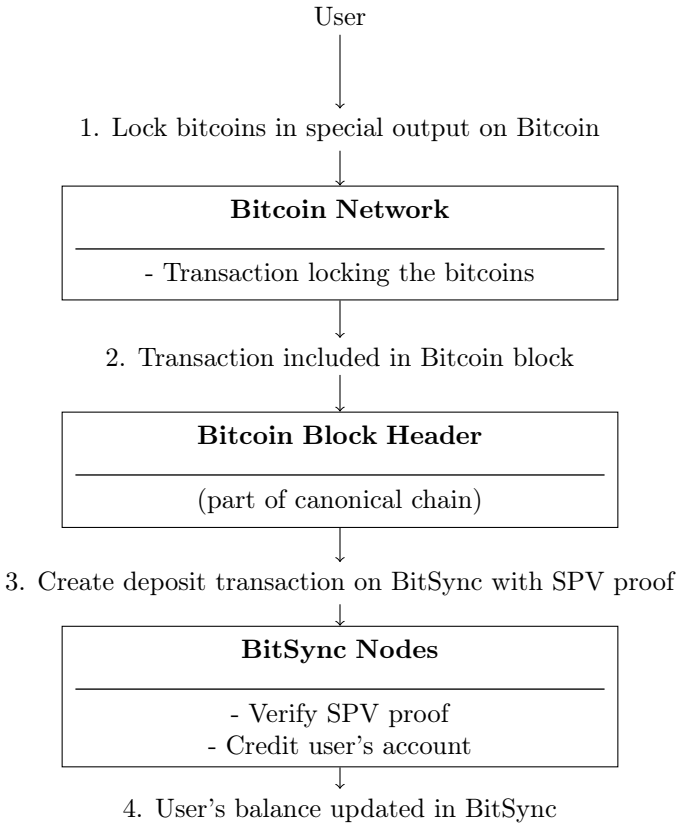

Figure 4: Diagram of the Deposit Process

7 Data Availability and Network Integrity

BitSync can employ a hash-based data availability mechanism, ensuring all necessary data for transaction verification is accessible without relying on slashing mechanisms common in proof-of-stake systems. This approach aligns with Bitcoin's proof-of-work context and avoids introducing additional trust assumptions.

Transactions are processed off-chain and settled on-chain through state commitments and data availability guarantees, improving throughput without compromising security.

8 Maintaining Decentralization and Network Health

8.1 Role of Special nodes

Special nodes in the network can be tasked with increasing the decentralization of the mesh-network and would not influence consensus or the protocol's security. Their incentive may come from network rewards and transaction fees for staying online and contributing to the system's health.

8.2 Decentralized Sequencing and MEV Protection

These Special nodes act as decentralized sequencers for rollups, facilitating the ordering of transactions to enhance liveness and alignment. This setup protects against centralized manipulation but also can mitigate risks associated with Maximal Extractable Value (MEV) by distributing sequencing responsibilities via auction mechanisms and race mechanics to solve-and-collect revenue.

9 Alternative Approaches and Considerations

9.1 Superblocks and NIPoPoWs

In scenarios where direct block hash verification is not possible, such as the absence of an operation like `OP_BLOCKHASH`, superblocks and Non-Interactive Proofs of Proof-of-Work (NIPoPoWs[3]) can be employed. However, although the security using superblocks will be great and likely better than existing designs, it is not unconditional unlike using `OP_BLOCKHASH`.

9.2 Potential Opcode Enhancements

The inclusion of an operation like `OP_CAT`[5] in Bitcoin would facilitate the creation of more complex scripts, potentially enabling direct block hash verification through script introspection. This capability would enhance the security and functionality of BitSync interactions but requires consensus changes to the Bitcoin protocol through a soft fork.

Historically, `OP_CAT` was disabled due to potential risks like denial-of-service (DoS) attacks, where an attacker could create scripts that consume excessive computational resources by concatenating large amounts of data. However, recent discoveries and proposals have mitigated these risks, making it safer to consider re-enabling `OP_CAT`. The `OP_CAT` op-code may make it easier to emulate `OP_BLOCKHASH` like capabilities and make it more efficient. There is also work in this direction to emulate these op-codes using ZKPs without any changes necessary to Bitcoin[12]

10 Conclusion

We have presented a sidechain design that integrates tightly with Bitcoin's security model through real-time synchronization and proof-of-work commitments. By requiring BitSync transactions to reference canonical Bitcoin block hashes and utilizing ZKPs for efficient verification, withdrawals are secured and resistant to manipulation through private forks.

BitSync enables flexibility and scalability through modular designs like rollups. The network maintains decentralization and integrity without compromising security by leveraging special nodes for governance and network support.

This architecture offers a robust foundation for extending Bitcoin's capabilities while preserving its core principles of security and decentralization.

A Formal Analysis of the Withdrawal Mechanism

In this appendix, we provide a detailed mathematical analysis of the peg-out (withdrawal) process in BitSync. We demonstrate how the design prevents private fork attacks and ensures secure Bitcoin redemption by integrating formal proofs, pseudocode from the AuxPOW protocol specification, and security analysis. The analysis covers the prevention of private forks in both the BitSync sidechain and the Bitcoin blockchain.

A.1 Notation and Definitions

We define the following:

- $\mathbb{B}$: The Bitcoin blockchain.
- $\mathbb{S}$: The BitSync sidechain.
- $B_i \in \mathbb{B}$: The i-th block in the Bitcoin blockchain.
- $SC_j \in \mathbb{S}$: The j-th block in the BitSync sidechain.
- $H_{B_i} = \text{Hash}(B_i)$: The hash of Bitcoin block B_i.
- $H_{SC_j} = \text{Hash}(SC_j)$: The hash of BitSync block SC_j.
- $\text{PoW}_{\mathbb{B}}(B_i)$: The proof-of-work for Bitcoin block B_i.
- $\text{PoW}_{\mathbb{S}}(SC_j)$: The proof-of-work for BitSync block SC_j.
- $\text{CanonicalChain}_{\mathbb{B}}$: The canonical (longest valid) Bitcoin chain.
- $\text{CanonicalChain}_{\mathbb{S}}$: The canonical (longest valid) BitSync chain.
- OP_BLOCKHASH(n): An opcode that returns H_{B_n}, the hash of the n-th Bitcoin block in the canonical chain.
- ZK_Proof: A zero-knowledge proof.
- $\mathcal{U}$: A user initiating a withdrawal.
- n: The nonce in the Bitcoin block header.
- c: The unique chain ID assigned to BitSync.
- h: The height of the chain Merkle tree in the coinbase transaction.
- getExpectedIndex(n, c, h): A function computing the expected index of BitSync's commitment in the chain Merkle tree.
- AuxPOW: The Auxiliary Proof-of-Work protocol used for merged mining.

A.2 AuxPOW Protocol Overview

The AuxPOW protocol enables merged mining, allowing miners to mine on both Bitcoin ($\mathbb{B}$) and BitSync ($\mathbb{S}$) simultaneously. Key aspects of AuxPOW relevant to our analysis include:

1. **Chain ID (c)**: A unique identifier for $\mathbb{S}$, ensuring commitments to different chains are distinguishable.
2. **Chain Merkle Tree**: A Merkle tree included in the Bitcoin coinbase transaction's scriptSig, containing commitments to auxiliary chain block hashes.
3. `getExpectedIndex` **Function**: Determines the expected index i in the chain Merkle tree for a given chain ID c, nonce n, and tree height h.

A.2.1 `getExpectedIndex` Function Pseudocode

The `getExpectedIndex` function is defined as:

Algorithm 1 `getExpectedIndex` Function

function GETEXPECTEDINDEX(n, c, h)
    $mod \leftarrow 2^h$
    $rand \leftarrow n$
    $rand \leftarrow (rand \times 1103515245 + 12345) \mod mod$
    $rand \leftarrow (rand + c) \times 1103515245 + 12345$
    $rand \leftarrow rand \mod mod$
    return $rand$
end function

This function ensures that for each combination of n, c, and h, the expected index i is unique and deterministic.

A.2.2 AuxPOW Validation Checks Pseudocode

The validation of an AuxPOW block includes the following steps:

Algorithm 2 AuxPOW Validation Checks

procedure ValidateAuxPOW(AuxPOW, H_{SC_j}, c, n, h)
    Ensure that the chain Merkle branch size is valid.
    if Number of chain Merkle branches > 30 **then**
        Reject block
    end if
    Compute the chain Merkle root M using H_{SC_j} and the chain Merkle branch.
    Verify that M is included in the coinbase transaction's scriptSig.
    Verify that only one AuxPOW header exists in the scriptSig.
    Verify that the expected index i matches the index in the chain Merkle branch:
        $i \leftarrow \text{getExpectedIndex}(n, c, h)$
    if Chain Merkle branch index $\neq i$ **then**
        Reject block
    end if
    Accept block if all checks pass.
end procedure

A.3 Withdrawal Process Description

The withdrawal process involves the following steps:

A.3.1 1. Inclusion of Bitcoin Block Hash in BitSync Block

Each BitSync block SC_j includes a field H_{B_i} representing the hash of the latest Bitcoin block B_i. This establishes a reference point to the Bitcoin blockchain.

A.3.2 2. Commitment in Bitcoin Coinbase Transaction

Bitcoin miners, when producing block B_{i+1}, include a commitment to the latest BitSync block hash H_{SC_j} in the coinbase transaction. This mutual reference reinforces the link between the chains.

A.3.3 3. User Initiates Withdrawal on BitSync

The user $\mathcal{U}$ performs a burn transaction tx_{burn} on BitSync, effectively removing the tokens from circulation. This transaction is included in BitSync block SC_j.

A.3.4 4. Zero-Knowledge Proof Construction

The user $\mathcal{U}$ constructs a zero-knowledge proof ZK_Proof that demonstrates:

1. The burn transaction tx_{burn} is included in SC_j.
2. SC_j includes H_{B_i}.
3. The proof-of-work up to SC_j is valid, i.e., $\text{PoW}_{\mathbb{S}}(SC_j)$ is valid.
4. Bitcoin block B_{i+1} includes a commitment to H_{SC_j} in its coinbase transaction.

A.3.5 5. Submission of Proof to Bitcoin Network

The user $\mathcal{U}$ submits a special transaction tx_{withdraw} to the Bitcoin network, including:

- The zero-knowledge proof ZK_Proof.
- The reference to H_{B_i}.

A.3.6 6. Verification on Bitcoin Network

Bitcoin nodes verify the transaction tx_{withdraw} by checking:

1. H_{B_i} corresponds to a block in the canonical Bitcoin chain via OP_BLOCKHASH$(i) = H_{B_i}$.
2. The zero-knowledge proof ZK_Proof is valid and references the canonical chains.

A.3.7 7. Releasing Funds

If verification succeeds, the Bitcoin network processes tx_{withdraw}, releasing the corresponding bitcoins to $\mathcal{U}$.

A.4 Formal Proof of Security Against Private Fork Attacks

We present a formal proof demonstrating that the withdrawal mechanism prevents private fork attacks by leveraging the AuxPOW protocol and the pseudocode provided.

A.4.1 Theorem

Theorem 1. *Within the AuxPOW protocol, it is impossible for an attacker to create a private fork of the BitSync sidechain or the Bitcoin blockchain that produces a valid withdrawal proof, due to the enforcement of unique chain ID commitments, the* `getExpectedIndex` *function, and the use of* `OP_BLOCKHASH` *for canonicalness.*

Proof.

Case 1: Private Fork of BitSync Chain Assume an attacker $\mathcal{A}$ attempts to create a private BitSync chain $\mathbb{S}'$ and produce a fraudulent burn transaction tx'_{burn} included in a sidechain block SC'_k. The attacker aims to use SC'_k to construct a zero-knowledge proof ZK_Proof$'$ for withdrawal.

1. **Uniqueness of Chain ID Commitment**

   According to the AuxPOW protocol and Algorithm 2, only one commitment per chain ID c can be included in the Bitcoin block's coinbase transaction. The protocol enforces:

   - Only one AuxPOW header exists in the coinbase transaction.
   - The commitment for chain ID c must be at the expected index i, computed by `getExpectedIndex` (Algorithm 1).

2. **Impossibility of Including Multiple Commitments**

   The deterministic nature of i ensures that miners cannot include multiple commitments for the same chain ID c without violating the protocol rules. Any attempt to include both H_{SC_j} and $H_{SC'_k}$ for c in the same Bitcoin block B_{i+1} would result in a conflict at index i.

3. **Verification Failure**

   The zero-knowledge proof ZK_Proof$'$ must satisfy the conditions:

$$\begin{cases} tx'_{\text{burn}} \in SC'_k \\ SC'_k \in \text{CanonicalChain}_{\mathbb{S}} \\ H_{B_i} \text{ is included in } SC'_k \\ H_{SC'_k} \text{ is included in } B_{i+1} \\ H_{B_i} = \text{OP\_BLOCKHASH}(i) \end{cases}$$

   Since SC'_k is not part of CanonicalChain$_{\mathbb{S}}$, and $H_{SC'_k}$ is not committed in B_{i+1} without violating the AuxPOW rules, the proof fails. Note that OP_BLOCKHASH would be likely validated against the witness input of the Bitcoin block hash into the verification circuit, and checked out of the circuit in script.

4. **Conclusion**

   The withdrawal verification on Bitcoin will fail because the proof cannot be constructed to satisfy all conditions without violating protocol rules. Thus, the attacker cannot perform a fraudulent withdrawal using a private fork of the BitSync chain.

Case 2: Private Fork of Bitcoin Chain Assume $\mathcal{A}$ attempts to create a private Bitcoin chain $\mathbb{B}'$ to include commitments to $H_{SC'_k}$.

1. **Canonicalness via `OP_BLOCKHASH`**

   The `OP_BLOCKHASH` opcode only returns block hashes from the canonical Bitcoin chain CanonicalChain$_{\mathbb{B}}$. Any blocks from $\mathbb{B}'$ are not recognized.

2. **Infeasibility of Sustaining a Private Bitcoin Fork**

   Without controlling the majority of Bitcoin's hash rate, $\mathcal{A}$ cannot produce a longer chain than CanonicalChain$_{\mathbb{B}}$. The network will reject $\mathbb{B}'$ in favor of the canonical chain.

3. **Verification Failure**

   The zero-knowledge proof ZK_Proof$'$ referencing $\mathbb{B}'$ will fail, as $H_{B_i} = \text{OP\_BLOCKHASH}(i)$ will not match any block in $\mathbb{B}'$.

4. **Conclusion**

   The withdrawal mechanism prevents $\mathcal{A}$ from using a private Bitcoin fork to facilitate fraudulent withdrawals, as the verification relies on data from the canonical Bitcoin chain.

Integration of Pseudocode in Proof The validation steps in Algorithm 2 are critical in enforcing the protocol rules that prevent the attack scenarios:

- The check for a single AuxPOW header and chain Merkle root ensures uniqueness of commitments.
- The `getExpectedIndex` function (Algorithm 1) provides a deterministic index, preventing multiple commitments for the same chain ID c.

- The verification of the chain Merkle branch and expected index in AuxPOW validation prevents inclusion of unauthorized sidechain blocks.

Final Conclusion By enforcing strict protocol rules through the AuxPOW validation checks and leveraging `OP_BLOCKHASH` for canonicalness, the withdrawal mechanism ensures that only valid withdrawals referencing the canonical Bitcoin and BitSync chains are accepted, effectively preventing private fork attacks.

■

A.5 State Representation and Transition Diagrams

A.5.1 State Variables

- $S_{\mathbb{S}}$: State of the BitSync blockchain.
- $S_{\mathbb{B}}$: State of the Bitcoin blockchain.
- $\mathcal{T}_{\text{burn}}$: Set of burn transactions on BitSync.
- $\mathcal{T}_{\text{withdraw}}$: Set of withdrawal transactions on Bitcoin.
- $\mathcal{U}$: User performing the withdrawal.

A.5.2 State Transition Functions

On BitSync When $\mathcal{U}$ performs tx_{burn}:

$$S_{\mathbb{S}} \xrightarrow{tx_{\text{burn}}} S'_{\mathbb{S}}$$
$$\mathcal{T}_{\text{burn}} \leftarrow \mathcal{T}_{\text{burn}} \cup \{tx_{\text{burn}}\}$$

On Bitcoin When $\mathcal{U}$ submits tx_{withdraw}:

$$S_{\mathbb{B}} \xrightarrow{tx_{\text{withdraw}}} S'_{\mathbb{B}}$$
$$\mathcal{T}_{\text{withdraw}} \leftarrow \mathcal{T}_{\text{withdraw}} \cup \{tx_{\text{withdraw}}\}$$

A.5.3 Process Flow Diagram

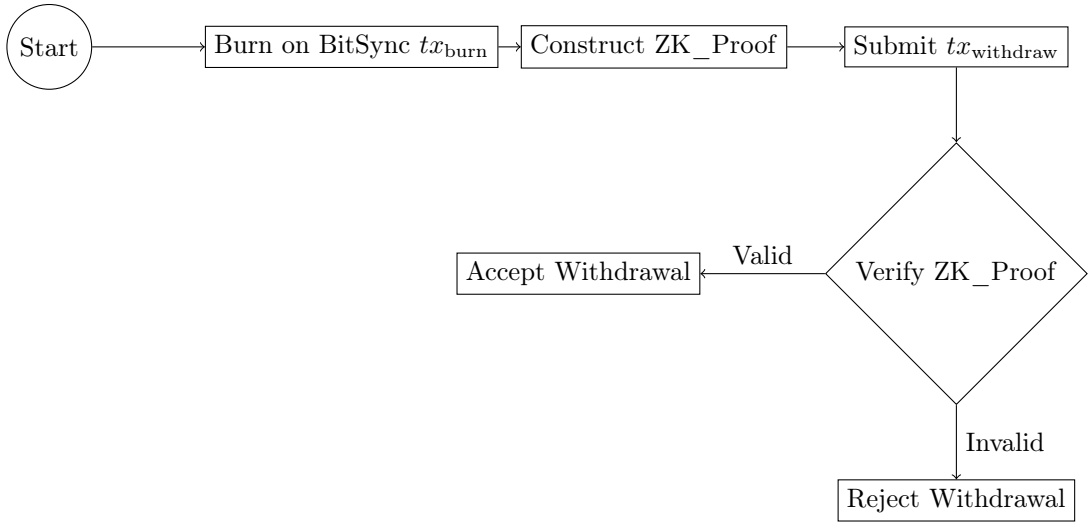

Figure 5: Process Flow of the Withdrawal Mechanism

A.6 Mathematical Representation of the Zero-Knowledge Proof

The zero-knowledge proof ZK_Proof must satisfy the following conditions:

1. Prove knowledge of tx_{burn} such that $tx_{\text{burn}} \in SC_j$ and $SC_j \in \text{CanonicalChain}_{\mathbb{S}}$.
2. SC_j includes H_{B_i}.
3. $\text{PoW}_{\mathbb{S}}(SC_j)$ is valid up to SC_j.
4. B_{i+1} includes a commitment to H_{SC_j} in its coinbase transaction.
5. $H_{B_i} = \text{OP\_BLOCKHASH}(i)$.

A.6.1 Formal Statement

Find $tx_{\text{burn}}, SC_j, B_i, B_{i+1}$ such that:

$$\begin{cases} tx_{\text{burn}} \in SC_j \\ SC_j \in \text{CanonicalChain}_{\mathbb{S}} \\ H_{B_i} \text{ is included in } SC_j \\ \text{PoW}_{\mathbb{S}}(SC_j) \text{ is valid} \\ B_{i+1} \in \text{CanonicalChain}_{\mathbb{B}} \\ H_{SC_j} \text{ is included in } B_{i+1} \\ H_{B_i} = \text{OP\_BLOCKHASH}(i) \end{cases}$$

Prove that these conditions hold without revealing tx_{burn} or other private data.

A.7 Security Analysis Against Attacks

A.7.1 1. Double-Spending Attacks

On BitSync Since tx_{burn} is included in SC_j and $SC_j \in \text{CanonicalChain}_{\mathbb{S}}$, the burn is irreversible on the sidechain.

On Bitcoin The zero-knowledge proof ensures that the funds are released only once, preventing double-spending on the Bitcoin network.

A.7.2 2. Replay Attacks

Including specific block hashes H_{B_i} and H_{SC_j} tied to the canonical chains prevents replay attacks, as each withdrawal is unique to the chain states at specific points in time.

A.7.3 3. Fraudulent Proofs

The use of zero-knowledge proofs ensures that only valid proofs constructed with correct data can pass verification. It is computationally infeasible to forge a valid proof without the required knowledge.

A.7.4 4. Private Fork Attacks

As demonstrated in the formal proof, the protocol prevents private fork attacks by:

- Enforcing unique chain ID commitments via the AuxPOW protocol.
- Utilizing the `getExpectedIndex` function to prevent multiple commitments for the same chain ID.
- Relying on `OP_BLOCKHASH` to ensure references to the canonical Bitcoin chain.

A.8 Conclusion

Through formal definitions, pseudocode integration, mathematical proofs, and security analysis, we have demonstrated that the withdrawal mechanism in BitSync securely allows users to redeem assets back to the Bitcoin network without the risk of private fork attacks or unauthorized redemption. The tight coupling between the chains, reliance on canonical chain data, and enforcement of protocol rules via validation checks ensure the integrity and security of the process.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008. Available at: `https://bitcoin.org/bitcoin.pdf`

[2] Robin Linus, *BitVM2: Bridging Bitcoin to Second Layers*, 2024. Available at: `https://bitvm.org/bitvm_bridge.pdf`

[3] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros, *Non-Interactive Proofs of Proof-of-Work*, Cryptology ePrint Archive, Paper 2017/963, 2017. Available at: `https://eprint.iacr.org/2017/963`

[4] Bitcoin Wiki, *Auxiliary Proof-of-Work (AuxPOW)*. Available at: `https://en.bitcoin.it/wiki/Merged_mining_specification`

[5] Ethan Heilman and Armin Sabouri, *BIP 420: OP_ CAT*, Bitcoin Improvement Proposals, Draft, 2023. Available at: `https://github.com/bip420/bip420`

[6] E. Ben Sasson et al., *Zerocash: Decentralized Anonymous Payments from Bitcoin*, 2014. Available at: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6956581`

[7] Paul Sztorc, *Drivechain: The Simple Two-Way Peg*, 2015. Available at: `https://www.truthcoin.info/blog/drivechain/`

[8] Sergio Demian Lerner, *RSK: Bitcoin-powered Smart Contracts*, 2016. Available at: `https://docs.rsk.co/RSK_White_Paper-Overview.pdf`

[9] Yoni Assia, Meni Rosenfeld, Vitalik Buterin, *Colored Coins: BitcoinX*, 2013. Available at: `https://bitcoil.co.il/BitcoinX.pdf`

[10] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille, *Enabling Blockchain Innovations with Pegged Sidechains*, Blockstream, 2014. Available at: `https://blockstream.com/sidechains.pdf`

[11] *The Stacks Blockchain: A Bitcoin Layer for Smart Contracts*, Stacks (Nakamoto), 2024. Available at: `https://stx.is/nakamoto`

[12] M. Komarov, *Bitcoin PIPEs: Covenants and ZKPs on Bitcoin Without Soft Fork*, [[alloc] init], May 1, 2024. Available at: `https://www.allocin.it/uploads/placeholder-bitcoin.pdf`