Special thanks to nico and Newki for the discussion with you guys.

This post is a proposed solution to release the restrictions that exist in Ethereum opcode by using zk-SNARKs. Furthermore, I hope to use this idea to solve the challenges of the optimistic bridges.

TLDR

- Ethereum has a blockhash opcode restriction commonly known as the “256 block problem”.
- This restriction means that the dispute period of the optimistic bridge is only about 51 minutes.
- To increase the dispute period, use a client that proves the blockhash hash-chain by using zk-SNARKs.
- Create a circuit of hashes leading up to the targeted blockhash by accessing backward from the latest opcode

Review of the optimistic bridge

If you already know about the optimistic bridge, please skip this part.

Optimistic bridges rather than requiring Ethereum to validate every bridge transaction, the protocol inspects only those that might be fraudulent. Suspicious activity can lead to a dispute, triggering a verification process settled by Layer 1.

Optimistic bridge consists of three participants - User, Relayer and Disputer. Relayer’s actions are constantly monitored by disupters and thrashed if fraud is discovered. See this post for more details.

Blockhash opcode

Ethereum has a blockhash opcode restriction commonly known as the “256 block problem”. The maximum number of blocks that can be referenced by Ethereum’s blockhash opcode is 256 blocks, and older blocks cannot be referenced.

This restriction means that at optimistic bridge, the time available for transaction verification by blockhash is only about 51 minutes (= 12 sec x 256 blocks). Consequently, the window for raising disputes is limited to approximately 51 minutes (12-second block time *256 blocks). In practical terms, this means that if disputers wish to challenge a relayer’s action, they have only about 51 minutes to do so, as the dispute process requires them to present the block hash of the contested transaction block as evidence. This tight timeframe could compromise system security and diminish the overall user experience.

Potential solution using zk-SNARKs

To address this restriction, I propose a solution leveraging zk-SNARKs.

Every block hash inherently embeds information from its predecessor. Given this property, it’s possible to create a cryptographic circuit that verifies the chain of block hashes, even beyond the most recent 256. This is done by anchoring the sequence with a block hash currently accessible via the blockhash opcode and tracing back to an older target hash.

In technical terms, if n1 represents the block number with the opcode-accessible block hash and n2 represents the block number of our target hash, the circuit would be designed to take three primary inputs:

- Block hash from block n1.
- All block headers between n1 and n2.
- Target block hash from block n2.

These operations will utilize RLP encoding and the Keccak256 hash function. This circuit proves that there is a hash connection up to the target block hash.

The following image provides a visual representation of this chain of hashes. However, in practice, the circuit computes this in reverse order. While this graphic specifically illustrates a block hash chain on Ethereum, the principle could also extend to Layer 2 networks.

[

circuit

1721×1080 137 KB

](https://ethresear.ch/uploads/default/original/2X/6/606609f2e659b178d985806544e99f5bfb4077bc.png)

Although this approach is in the research phase and awaits full implementation—primarily because of its intricate nature—it seems almost inevitable that such zk circuits will find integration within the bridge dispute mechanism. This is especially true considering some Layer 2 networks impose even tighter restrictions on block hash access than the 256-block restriction.