Title

– Proposal: Allow access to L1 block hash on Arbitrum

Constitutional / Non-Constitutional

- Constitutional

Abstract

- Allow access to L1 block hash using precompile or opcode on Arbitrum. The L1 block hash can be included in the Arbitrum block's header similar to L1 block and timestamp.

Motivation

- L1 block hash can enable accessing data on L1 or other L2s. Ability to have L1 block hash on Arbitrum in a trustless way can enable novel cross-chain use cases with improved security.

Rationale

- Currently applications that require access to data from L1 or other L2 chains have to use oracles which usually derive security from multisig. We have historically seen events where signers can be compromised and it can pose a threat to such applications. Having L1 block hash available on Arbiturm can help with this.

Key Terms

(optional) - N/A

Specifications

-

The ArbSys contract should accept calls using getL1BlockHash()

and returns back a recent L1 block hash which is present in the block header of the arbitrum block.

function getL1BlockHash() external returns (bytes32);

Steps to Implement

-

1. Include a recent L1 block hash on arbitrum's block header
2. Add a function to ArbSys to retrieve L1 block hash

Timeline

- TBD

Overall Cost

- TBD

Additional Information

- I've recently worked on a PoC of reading L1 data on L2 at EthGlobal Paris. The problem statement is that gnosis safe has to be deployed and managed on all the chains independently. Hence, this project keeps the source of truths of gnosis safe signers on L1 and on L2 it uses the state of L1 without using a trusted bridge. Here is the GitHub link GitHub - zemse/sunflower: use L1 multisig owners to sign on L2 multisig . I have created a proposal here on Arbitrum forum so that trustless cross-chain applications would be able to build on Arbitrum too.