Proving Ethereum's state on Starknet - Herodotus

# What are storage proofs ?

Storage proofs are a cryptographic way to track and share blockchain information across chains. Similar to oracles, storage proofs are a way to verify that information is true. However, unlike oracles, they do not require trust in a third party. With storage proofs, trust is built in mathematics.

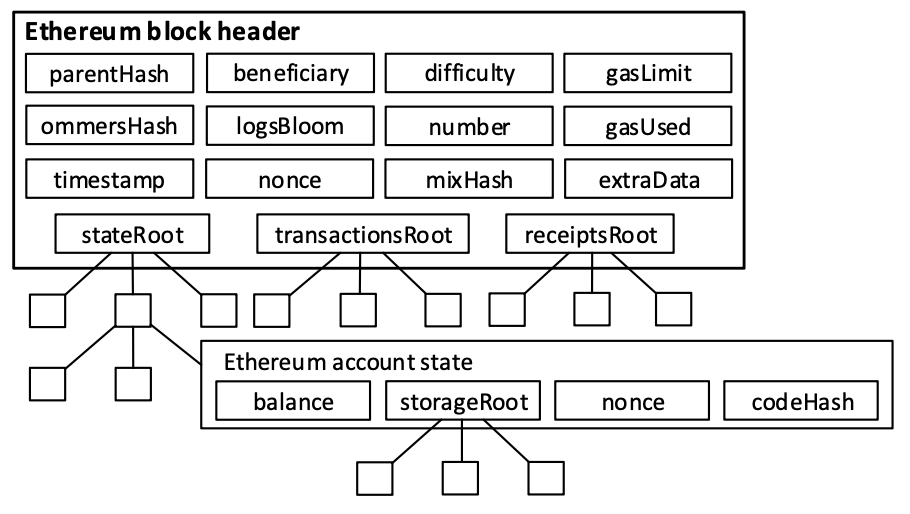
Storage proofs have a lot of potential use cases ranging from cross-chain information transfer to access to unlimited historical data access. The go-to solution for cross-chain information transfer is using oracles or multi-sig wallets. However, we often need to remember that the whole idea behind moving to a blockchain world was to remove human-based trust. Oracles and multi-signature wallets are human-based because humans have the power to cheat. In the case of oracles, if the majority of voters agree on an untruthful fact, it would be accepted without any penalty for the voters. Security restrictions are even looser in the case of multi-signature wallets. Storage proofs allow us to prove that a certain state existed at some point in the past without having to trust any third party.

In [one of our articles](https://www.starknet.io/en/posts/developers/what-are-storage-proofs-and-how-can-they-improve-oracles#) on storage proofs, we introduced [Herodotus](https://www.herodotus.dev/), the team that is leading in research and innovation on storage proofs. With the recent announcement of [Herodotus Turbo](https://twitter.com/HerodotusDev/status/1682400907969822722), dozens of Starknet projects are closely keeping an eye on the developments with the storage-proof ecosystem. The most recent exciting update from the Herodotus team is trustless proving of the Ethereum genesis block. Let's explore the ingenious method that is benign used by Herodotus and the vast range of use cases this feature could power on-chain.

# Proving Ethereum’s Genesis block on-chain

## What is Herodotus trying to achieve ?

Every block on Ethereum and EVM chains has an associated string called the "block hash" resulting from hashing all block details with a hashing function such as Keccak 256. The historical block hash accumulators that Herodotus is building will provide a trustless way to prove the block hash for any past block on Ethereum.



Ethereum block header and state merkle tree ([source](https://www.researchgate.net/publication/330752524_A_Platform_Architecture_for_Multi-Tenant_Blockchain-Based_Systems))

A wide range of information is stored inside a block header: hash of the parent block, block timestamp, the state root, etc. The state root is a particularly interesting structure that allows anyone to prove the state of the whole Ethereum network at any particular point in time using a particular structure - [Merke Patricia Trie](https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie/) - used by Ethereum.

Every Ethereum account has an associated storage space where all the account variables are stored (essentially, the smart contract's state). A cryptographic commitment of the account's storage is hashed and stored as a *storage root* along with the account balance, nonce, and code hash. This is a basic summary of the Ethereum account state (labeled in the diagram above).

Furthermore, a Merke Patricia State with all the Ethereum account states as nodes is constructed, and its hash is stored in the block header as the state root (labeled *stateRoot* in the diagram above).

Block hash (a simple hash of the block header structure above) hence acts as a cryptographic commitment of the entire Ethereum state at a particular time. Suppose there is a trustless way to retrieve the block hash of any historical block. In that case, it opens up opportunities to utilize this data in various ways (to be discussed later below).

## Can we get historical block hashes currently on EVM ?

In Solidity, to retrieve the block hash of the block that was mined two blocks ago (the backward count starts from the block in which a transaction is included), the developer has to use the following syntax:

| bytes32 lastBlockHash = blockhash(block.number - 2); |
| --- |

Pretty simple, isn't it? But there's a catch. The blockhash method mentioned above can only retrieve the blockhash for the last 256 blocks. With an average of 12-second blocks on Ethereum, that is 51.2 minutes of history on-chain. For someone who wishes to use historical blockhashes as a source of entropy (randomness), the 256 limit is generally enough. However, for any purposes that aim to use block hashes for fetching the historical state of the chain at a specific block, a 51-minute history is not nearly enough (compared to an eight-year history of the Ethereum blockchain).

The 256 limit of on-chain block hash retrieval is primarily a design choice made for state storage efficiency and to mitigate potential state growth problems. A simple full node generally only stores the complete state of the chain at some blocks in time. It has access to the latest state and the list of all on-chain transactions. Since Ethereum strives to be as lightweight as possible regarding the mandatory state that each node must maintain, the 256-block limit compromises functionality and efficiency.

## Methodology of proving the Genesis block hash ?

So, how does Herodotus solve this limitation of 256 blocks in a completely trustless manner? The power of storage proofs is the key. Let's delve deeper by dividing the Herodotus procedure into steps:

**Step 1. Registering a recent block hash.**

On the Ethereum mainnet, a recent block hash is registered in a simple smart contract named the ProvingManager smart contract. The block hash can be retrieved using the blockhash opcode (opcode value 0x40) and saved in the mentioned smart contract as a simple string variable. The corresponding block number can also be registered for access later on. Let us assume that the registered block number is block [18,000,000](https://etherscan.io/block/18000000). From EtherScan, we can see that the block hash of this block is 0x95b1…4baf3.

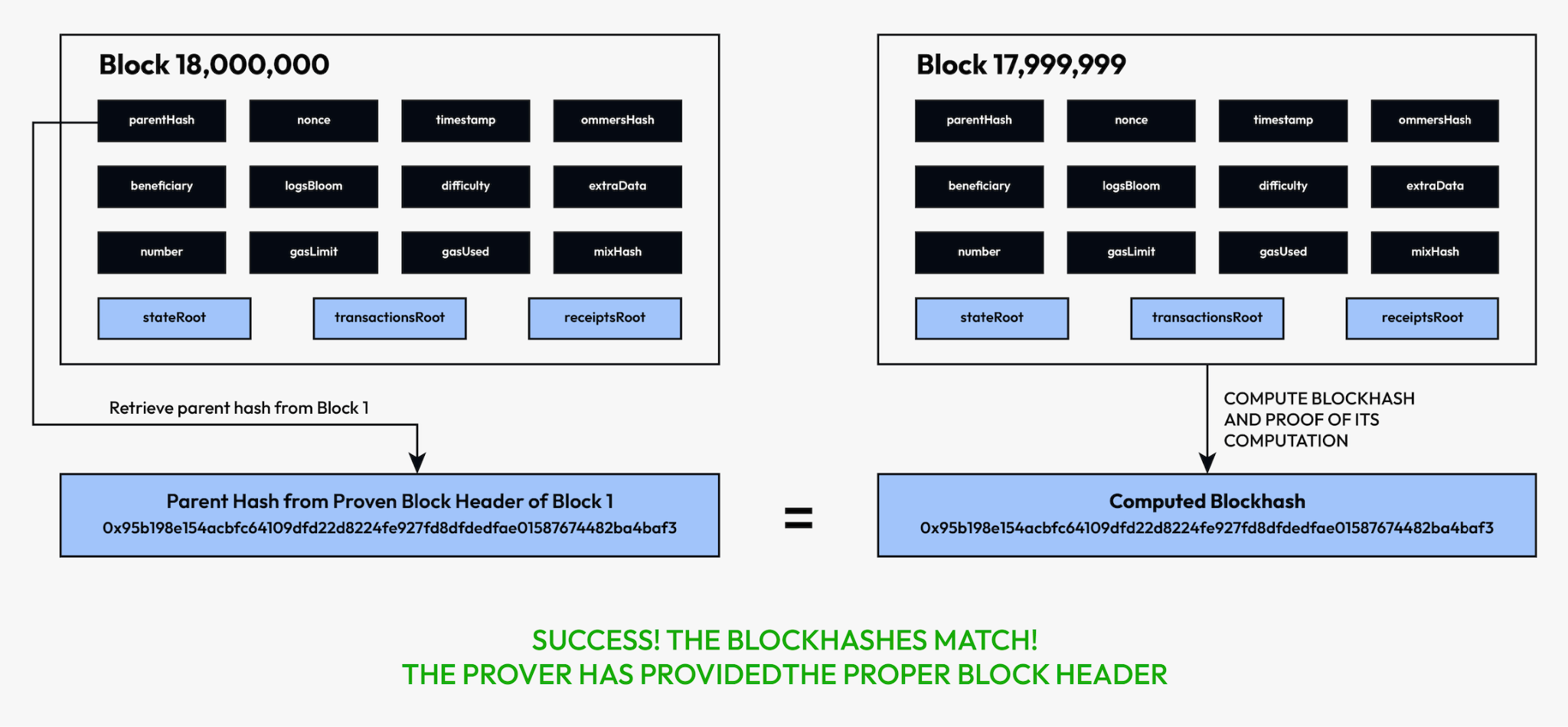
**Step 2. Proving the block hash of the latest block.**

The next step is to retrieve the block header (from an archival node) for block [18,000,000](https://etherscan.io/block/18000000), compute its blockhash in an off-chain computation, and compare it to the registered blockhash 0x95b1…4baf3. This calculation is also run through a prover to create a proof of this computation.

The block hashes for these blocks are added to another data structure - the Merke Mountain Range (MMR) - a variation of a Merkle tree such that appending new elements to the tree does not require significant computation.

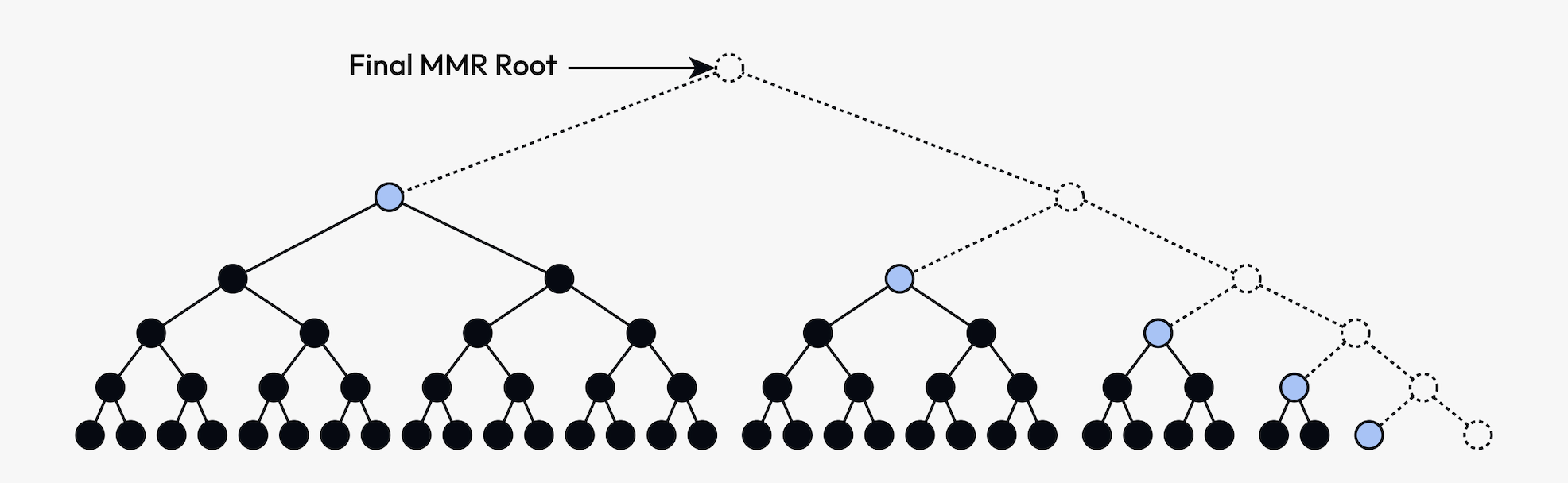
**Step 3. Proving the block hash of block X-1**

Once we have proven that the block header retrieved from the archival node is valid, we retrieve the block header for block X-1, compute its block hash and compare it with the parentBlockHash value for block X (it is available in the X's block header that we retrieved earlier). If the hashes match, we can be certain that the block header for X-1 is also valid. Since this whole computation can be modeled as a function, a STARK proof of this computation can be created simultaneously. Hence, a proof of validity for the block header of block X-1 is created (see diagram below).



**Step 4. Recursively proving the block hashes for previous blocks**

The block hashes computed similarly for all the previous blocks until we reach the Genesis block - the first block on the Ethereum mainnet - are appended to the MMR tree, thus creating a final MMR root.



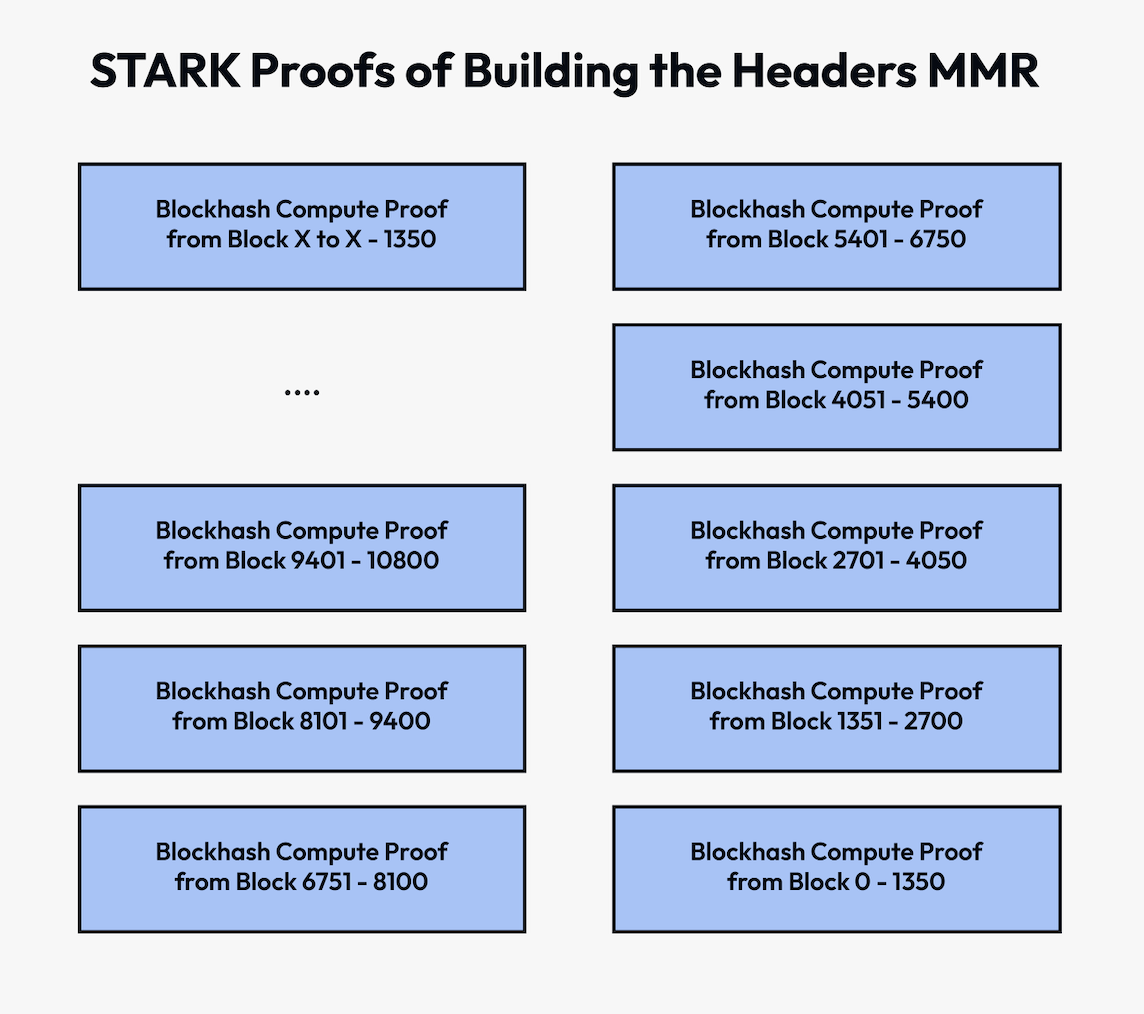
It should be noted that all of these computations are done off-chain with a simultaneous generation of proof of computation by Herodotus.

**Step 5. Publishing of proof on-chain and subsequent use**

Once the final MMR root is generated, this root can be published on-chain (on a Proofs Aggregation smart contract) along with the proof of computation for the millions of blocks. Since the generated STARK proofs are exponentially cheaper to verify, the cost of verifying these proofs on-chain adds up to a feasible amount.

Now that we have an idea of the complete process, some key points are important to note:

* Herodotus processes the blocks in batches of 1350 blocks, and the proof is published on-chain for every single batch. Once the whole process is completed for the first 18,000,000 blocks on Ethereum, the MMR root for the history of the block can be updated periodically as new blocks are added to the history of the chain.



* Once the MMR root is available on-chain, it becomes possible to prove the inclusion of every single block hash in the MMR (that's a basic property of Merkle trees). With this, any information contained in the block hash can be proven.
* The prover the Herodotus team uses before sending the proof on the Ethereum mainnet is the SHARP prover created by the StarkWare team. The primary benefit of the SHARP system lies in its ability to decrease costs and enhance the efficiency of the generated proofs.
* The complete process described above is being duplicated for two independent MMRs - one that uses the Keccak256 hashing function (which is native to Ethereum) and one that uses the Poseidon. Poseidon is a hash function specifically designed for STARK proofs.
* Herodotus is creating two different MMR trees for adding cross-chain interoperability of the block hash accumulators between Ethereum and Starknet. The MMR root for the Ethereum mainnet is also sent to Starknet using the native Starknet L1-to-L2 messaging protocol for use on Starknet.
* The block hash accumulators that Herodotus will publish would achieve a similar outcome as the [EIP-2935](https://eips.ethereum.org/EIPS/eip-2935) proposed by Vitalik Buterin in 2020 - but the EIP was never implemented. This is enough proof to highlight the requirement of a solution to access historical block hashes older than 256 blocks in a chain-native way - and the problem had remained unsolved for over three years!

# New Opportunities that open up with the Herodotus block hash accumulators

These block hash accumulators are being brought to Ethereum by Herodotus and StarkWare as a public good. Once the final MMR roots are published on Ethereum mainnet and Starknet along with proofs of computation, any developer could utilize it for accessing a provable state of the chain for any time since inception. The possibilities are limitless for the use of such information.