title: Data availability description: An overview of problems and solutions relating to data availability in Ethereum lang: en

"Don't trust, verify" is a common maxim in Ethereum. The idea is that your node can independently verify that the information it receives is correct by executing all the transactions in the blocks they receive from peers to ensure that the changes proposed precisely match those computed independently by the node. This means nodes do not have to trust that the senders of the block are honest. This is not possible if data is missing.

Data availability refers to the confidence a user can have that the data required to verify a block is really available to all network participants. For full nodes on Ethereum layer 1 this is relatively simple; the full node downloads a copy of all the data in each block - the data *has* to be available for the downloading to be possible. A block with missing data would be discarded rather than being added to the blockchain. This is "on chain data availability" and it is a feature of monolithic blockchains. Full nodes cannot be tricked into accepting invalid transactions because they download and execute every transaction for themselves. However, for modular blockchains, layer 2 rollups and light clients, the data availability landscape is more complex, requiring some more sophisticated verification procedures.

Prerequisites {#prerequisites}

You should have a good understanding of blockchain fundamentals, especially consensus mechanisms. This page also assumes the reader is familiar with blocks, transactions, nodes, scaling solutions, and other relevant topics.

The data availability problem {#the-data-availability-problem}

The data availability problem is the need to prove to the whole network that the summarized form of some transaction data that is being added to the blockchain really represents a set of valid transactions, but doing so without requiring all nodes to download all data. The full transaction data is necessary for independently verifying blocks, but requiring all nodes to download all transaction data is a barrier to scaling. Solutions to the data availability problem aim to provide sufficient assurances that the full transaction data was made available for verification to network participants that do not download and store the data for themselves.

Light nodes and Layer 2 rollups are important examples of network participants that require strong data availability assurances but cannot download and process transaction data for themselves. Avoiding downloading transaction data is what makes light nodes light and enables rollups to be effective scaling solutions.

Data availability is also a critical concern for future "stateless" Ethereum clients that do not need to download and store state data in order to verify blocks. The stateless clients still need to be certain that the data is available *somewhere* and that it has been processed correctly.

Data availability solutions {#data-availability-solutions}

Data availability sampling (DAS) {#data-availability-sampling}

Data Availability Sampling (DAS) is a way for the network to check that data is available without putting too much strain on any individual node. Each node (including non-staking nodes) downloads some small, randomly selected subset of the total data. Successfully downloading the samples confirms with high confidence that all of the data is available. This relies upon data erasure coding, which expands a given dataset with redundant information (the way this is done is to fit a function known as a *polynomial* over the data and evaluating that polynomial at additional points). This allows the original data to be recovered from the redundant data when necessary. A consequence of this data creation is that if *any* of the original data is unavailable, *half* of the expanded data will be missing! The amount of data samples downloaded by each node can be tuned so that it is *extremely* likely that at least one of the data fragments sampled by each client will be missing *if* less than half the data is really available.

DAS will be used to ensure rollup operators make their transaction data available after EIP-4844 has been implemented. Ethereum nodes will randomly sample the transaction data provided in blobs using the redundancy scheme explained above to ensure that all the data exists. The same technique could also be employed to ensure block producers are making all their data available to secure light clients. Similarly, under proposer-builder separation, only the block builder would be required to process an entire block - other validators would verify using data availability sampling.

Data availability committees {#data-availability-committees}

Data Availability Committees (DACs) are trusted parties that provide, or attest to, data availability. DACs can be used instead of, or in combination with DAS. The security guarantees that come with committees depends on the specific set up. Ethereum uses randomly sampled subsets of validators to attest to data availability for light nodes, for example.

DACs are also used by some validiums. The DAC is a trusted set of nodes that stores copies of data offline. The DAC is required to make the data available in the event of a dispute. Members of the DAC also publish on-chain attestations to prove that the said data is indeed available. Some validiums replace DACs with a proof-of-stake (PoS) validator system. Here, anyone can become a validator and store data off-chain. However, they must provide a “bond”, which is deposited in a smart contract. In the event of malicious behavior, such as the validator withholding data, the bond can be slashed. Proof-of-stake data availability committees are considerably more secure than regular DACs because they directly incentivize honest behavior.

Data availability and light nodes {#data-availability-and-light-nodes}

Light nodes need to validate the correctness of the block headers they receive without downloading the block data. The cost of this lightness is the inability to independently verify the block headers by re-executing transactions locally in the way full nodes do.

Ethereum light nodes trust random sets of 512 validators that have been assigned to a *sync committee*. The sync committee acts as a DAC that signals to light clients that the data in the header is correct using a cryptographic signature. Every day, the sync committee refreshes. Each block header alerts light nodes as to which validators to expect to sign off the *next* block, so they can't be tricked into trusting a malicious group pretending to be the real sync-committee.

However, what happens if an attacker somehow *does* manage to pass a malicious block header to light clients and convince them that it was signed off by an honest sync-committee? In that case, the attacker could include invalid transactions and the light client would blindly accept them, as they do not independently check all the state changes summarized in the block header. To protect against this, the light client could use fraud proofs.

The way these fraud proofs work is that a full node, seeing an invalid state transition being gossiped around the network, could quickly generate a small piece of data demonstrating that a proposed state transition could not possibly arise from a given set of transactions and broadcast that data to peers. Light nodes could pick up those fraud-proofs and use them to discard bad block headers, ensuring they stay on the same honest chain as the full nodes.

This relies on full nodes having access to full transaction data. An attacker who broadcasts a bad block header and also fails to make the transaction data available would be able to prevent full nodes from generating fraud proofs. The full nodes might be able to signal a warning about a bad block, but they couldn't back up their warning with proof, because the data wasn't made available to generate the proof from!

The solution to this data availability problem is DAS. Light nodes download very small random chunks of the full state data and use the samples to verify that the full data set is available. The actual likelihood of incorrectly assuming full data availability after downloading N random chunks can be calculated (for 100 chunks the chance is 10^-30, i.e. incredibly unlikely).

Even in this scenario, attacks that withhold just a few bytes could feasibly go unnoticed by clients making random data requests. Erasure coding fixes this by reconstructing small missing pieces of data that can be used to check proposed state changes. A fraud proof could then be constructed using the reconstructed data, preventing light nodes from accepting bad headers.

Note: DAS and fraud proofs have not yet been implemented for proof-of-stake Ethereum light clients, but they are on the roadmap, most likely taking the form of ZK-SNARK based proofs. Today's light clients rely on a form of DAC: they verify the identities of the sync-committee and then trust the signed block headers they receive.

Data availability and layer 2 rollups {#data-availability-and-layer-2-rollups}

Layer 2 scaling solutions, such as rollups, reduce transaction costs and increase Ethereum's throughput by processing transactions off-chain. Rollup transactions are compressed and posted on Ethereum in batches. Batches represent thousands of individual off-chain transactions in a single transaction on Ethereum. This reduces congestion on the base layer and reduces fees for users.

However, it is only possible to trust the 'summary' transactions posted to Ethereum if the state change proposed can be independently verified and confirmed to be the result of applying all the individual off-chain transactions. If rollup operators do not make the transaction data available for this verification, then they could send incorrect data to Ethereum.

Optimistic rollups post compressed transaction data to Ethereum and wait for some amount of time (typically 7 days) to allow independent verifiers to check the data. If anyone identifies a problem, they can generate a fraud-proof and use it to challenge the rollup. This would cause the chain to roll back and omit the invalid block. This is only possible if data is available. Currently, data is made permanently available as `CALLDATA` which lives permanently on-chain. However, EIP-4844 will soon allow rollups to post their transaction data to cheaper blob storage instead. This is not permanent storage. Independent verifiers will have to query the blobs and raise their challenges within ~1-3 months before the data is deleted from Ethereum layer-1. Data availability is only guaranteed by the Ethereum protocol for that short fixed window. After that, it becomes the responsibility of other entities in the Ethereum ecosystem. Any node can verify data availability using DAS, i.e. by downloading small, random samples of the blob data.

Zero-knowledge (ZK) rollups don't need to post transaction data since zero-knowledge validity proofs guarantee the correctness of state transitions. However, data availability is still an issue because we can't guarantee the functionality of the ZK-rollup (or interact with it) without access to its state data. For example, users cannot know their balances if an operator withholds details about the rollup’s state. Also, they cannot perform state updates using information contained in a newly added block.

Data availability vs. data retrievability {#data-availability-vs-data-retrievability}

Data availability is different from data retrievability. Data availability is the assurance that full nodes have been able to access and verify the full set of transactions associated with a specific block. It does not necessarily follow that the data is accessible forever.

Data retrievability is the ability of nodes to retrieve *historical information* from the blockchain. This historical data is not needed for verifying new blocks, it is only required for syncing full nodes from the genesis block or serving specific historical requests.

The core Ethereum protocol is primarily concerned with data availability, not data retrievability. Data retrievability can be provided by a small population of archive nodes run by third parties, or it can be distributed across the network using decentralized file storage such as the Portal Network.

Further reading {#further-reading}

- WTF is Data Availability?
- What Is Data Availability?
- The Ethereum Off-Chain Data Availability Landscape
- A primer on data availability checks
- An explanation of the sharding + DAS proposal
- A note on data availability and erasure coding
- Data availability committees.
- Proof-of-stake data availability committees.
- Solutions to the data retrievability problem