---

title: Danksharding description: Learn about Proto-Danksharding and Danksharding - two sequential upgrades for scaling Ethereum. lang: en summaryPoints: - Danksharding is a multi-phase upgrade to improve Ethereum's scalability and capacity. - The first stage, Proto-Danksharding, adds data blobs to blocks - Data blobs offer a cheaper way for rollups to post data to Ethereum and those costs can be passed on to users in the form of lower transaction fees. - Later, full Danksharding will spread responsibility for verifying data blobs across subsets of nodes, further scaling Ethereum to more than 100,000 transactions per second.

Danksharding {#danksharding}

Danksharding is how Ethereum becomes a truly scalable blockchain, but there are several protocol upgrades required to get there. **Proto-Danksharding** is an intermediate step along the way. Both aim to make transactions on Layer 2 as cheap as possible for users and should scale Ethereum to >100,000 transactions per second.

What is Proto-Danksharding? {#what-is-protodanksharding}

Proto-Danksharding, also known as EIP-4844, is a way for rollups to add cheaper data to blocks. The name comes from the two researchers who proposed the idea: Protolambda and Dankrad Feist. Right now, rollups are limited in how cheap they can make user transactions by the fact that they post their transactions in `CALLDATA`. This is expensive because it is processed by all Ethereum nodes and lives on chain forever, even though rollups only need the data for a short time. Proto-Danksharding introduces data blobs that can be sent and attached to blocks. The data in these blobs is not accessible to the EVM and is automatically deleted after a fixed time period (1-3 months). This means rollups can send their data much more cheaply and pass the savings on to end users in the form of cheaper transactions.

Rollups are a way to scale Ethereum by batching transactions off-chain and then posting the results to Ethereum. A rollup is essentially composed of two parts: data and execution check. The data is the full sequence of transactions that is being processed by a rollup to produce the state change being posted to Ethereum. The execution check is the re-execution of those transactions by some honest actor (a "prover") to ensure that the proposed state change is correct. In order for the execution check, the transaction data has to be available for long enough for anyone to download and check. This means any dishonest behavior by the rollup sequencer can be identified and challenged by the prover. However, it does not need to be available forever.

Rollups post commitments to their transaction data on-chain and also make the actual data available in data blobs. This means provers can check the commitments are valid or challenge data they think is wrong. At the node-level, the blobs of data are held in the consensus client. The consensus clients attest that they have seen the data and that it has been propagated around the network. If the data was kept forever, these clients would bloat and lead to large hardware requirements for running nodes. Instead, the data is automatically pruned from the node every 1-3 months. The consensus client attestations demonstrate that there was a sufficient opportunity for provers to verify the data. The actual data can be stored off-chain by rollup operators, users or others.

How is blob data verified? {#how-are-blobs-verified}

Rollups post the transactions they execute in data blobs. They also post a "commitment" to the data. They do this by fitting a polynomial function to the data. This function can then be evaluated at various points. For example, if we define an extremely simple function `f(x) = 2x-1` then we can evaluate this function for `x = 1`, `x = 2`, `x = 3` giving the results `1, 3, 5`. A prover applies the same function to the data and evaluates it at the same points. If the original data is changed, the function will not be identical, and therefore neither are the values evaluated at each point. In reality, the commitment and proof are more complicated because they are wrapped in cryptographic functions.

What is KZG? {#what-is-kzg}

KZG stands for Kate-Zaverucha-Goldberg - the names of the threeoriginal authors of a scheme that reduces a blob of data down to a small cryptographic "commitment". The blob of data submitted by a rollup has to be verified to ensure the rollup is not misbehaving. This involves a prover re-executing the transactions in the blob to check that the commitment was valid.

This is conceptually the same as the way execution clients check the validity of Ethereum transactions on layer 1 using Merkle proofs. KZG is an alternative proof that fits a polynomial equation to the data. The commitment evaluates the polynomial at some secret data points. A prover would fit the same polynomial over the data and evaluate it at the same values, checking that the result is the same. This is a way to verify the data that is compatible with zero-knowledge techniques used by some rollups and eventually other parts of the Ethereum protocol.

What is the KZG Ceremony? {#what-is-a-kzg-ceremony}

A KZG ceremony is a way for many people from across the Ethereum community to generate a secret random string of numbers together that can be used to verify some data. It is very important that this string of numbers is not known and cannot be recreated by anyone. To ensure this, each person that participates in the ceremony receives a string from the previous participant. They then create some new random values (e.g. by allowing their browser to measure the movement of their mouse) and mix it in with the previous value. They then send the value on to the next participant and destroy it from their local machine. As long as one person in the ceremony does this honestly, the final value will be unknowable to an attacker. The EIP-4844 KZG ceremony was open to the public and tens of thousands of people participated to add their own entropy. For the ceremony to be undermined, 100% of those participants would have to be actively dishonest. From the perspective of the participants, if they know they were honest, there is no need to trust anyone else because they know that they secured the ceremony (they individually satisfied the 1-out-of-N honest participant requirement).

When a rollup posts data in a blob, they provide a "commitment" that they post on chain. This commitment is the result of evaluating a polynomial fit to the data at certain points. These points are defined by the random numbers generated in the KZG ceremony. Provers can then evaluate the polynomial at the same points in order to verify the data - if they arrive at the same values then the data is correct.

If someone knows the random locations used for the commitment, it is easy for them to generate a new polynomial that fits at those specific points (i.e. a "collision"). This means they could add or remove data from the blob and still provide a valid proof. To prevent this, instead of giving provers the actual secret locations, they actually receive the locations wrapped in a cryptographic "black box" using elliptic curves. These effectively scramble the values in such a way that the original values cannot be reverse-engineered, but with some clever algebra provers and verifiers can still evaluate polynomials at the points they represent.

Neither Danksharding nor Proto-Danksharding follow the traditional "sharding" model that aimed to split the blockchain into multiple parts. Shard chains are no longer part of the roadmap. Instead, Danksharding uses distributed data sampling across blobs to scale Ethereum. This is much simpler to implement. This model has sometimes been referred to as "data-sharding".

What is Danksharding? {#what-is-danksharding}

Danksharding is the full realization of the rollup scaling that began with Proto-Danksharding. Danksharding will bring massive amounts of space on Ethereum for rollups to dump their compressed transaction data. This means Ethereum will be able to support hundreds of individual rollups with ease and make millions of transactions per second a reality.

The way this works is by expanding the blobs attached to blocks from 1 in Proto-Danksharding to 64 in full Danksharding. The rest of the changes required are all updates to the way consensus clients operate to enable them to handle the new large blobs. Several of these changes are already on the roadmap for other purposes independent of Danksharding. For example, Danksharding requires proposer-builder separation to have been implemented. This is an upgrade that separates the tasks of building blocks and proposing blocks across different validators. Similarly, data availability sampling is required for Danksharding, but it is also required for the development of very lightweight clients that do not store much historical data ("stateless clients").

Proposer-builder separation is required to prevent individual validators from having to generate expensive commitments and proofs for 32MB of blob data. This would put too much strain on home stakers and require them to invest in more powerful hardware, which hurts decentralization. Instead, specialized block builders take responsibility for this expensive computational work. Then, they make their blocks available to block proposers to broadcast. The block proposer simply chooses the block that is most profitable. Anyone can verify the blobs cheaply and quickly, meaning any normal validator can check that the block builders are behaving honestly. This allows the large blobs to be processed without sacrificing

decentralization. Misbehaving block builders could simply be ejected from the network and slashed - others will step into their place because block building is a profitable activity.

Data availability sampling is required for validators to quickly and efficiently verify blob data. Using data availability sampling, the validators can be very certain that the blob data was available and correctly committed. Every validator can randomly sample just a few data points and create a proof, meaning no validator has to check the entire blob. If any data is missing, it will be identified quickly and the blob rejected.

Current progress {#current-progress}

Full Danksharding is several years away. However, Proto-Danksharding should arrive relatively soon. At the time of writing (Feb 2023) the KZG ceremony is still open and has so far attracted over 50,000 contributors. The EIP for Proto-Danksharding is mature, the specification is agreed and the clients have implemented prototypes that are currently being tested and made production-ready. The next step is to implement the changes on a public testnet. You can keep up to date using the EIP 4844 readiness checklist.

Further reading {#further-reading}

- Proto-Danksharding notes - *Vitalik Buterin*
- Dankrad's notes on Danksharding
- Dankrad, Proto and Vitalik discuss Danksharding
- The KZG ceremony
- Carl Beekhuizen's Devcon talk on trusted setups
- More on data availability sampling for blobs
- Dankrad Feist on KZG commitments and proofs
- KZG polynomial commitments