Documentation Index

Fetch the complete documentation index at: /llms.txt

Use this file to discover all available pages before exploring further.

Skip to main content

Starknet Documentation home page STARKNET

Welcome

Build

Secure

Learn

- Ask AI

Search...

Navigation

Protocol

Introduction to Starknet's protocol

Search... ⌘K

Protocol

- Introduction
- Concepts
- Components

S-two book

Archived copy · cryp2nova.com

- Introduction
- Why Use a Proof System?
- Why S-two?
- AIR Development
- The Cairo AIR
- S-two: Under the Hood
- Benchmarks Report
- Awesome S-two

Cheatsheets

- Specifications
- Ecosystem

Starknet Privacy

- Getting started
- Protocol
- Compliance and DeFi
- Security and operations
- Reference

On this page

- User interactions
- Block production and validation
- Tokenomics and consensus
- Summary

Protocol

Introduction to Starknet's protocol

Copy pageCopy page

Copy pageCopy page

Starknet is a decentralized, permissionless Layer 2 (L2) validity rollup that scales Ethereum by utilizing STARK-based zero-knowledge proofs to bundle thousands of transactions off-chain for verified settlement on the mainnet. As of 2026, the network evolved into a privacy-centric hub with the launch of the STRK20 framework, which allows assets like stablecoins to maintain confidential balances and transfers while remaining regulatory-compliant.

User interactions

As with other blockchains, users interact with Starknet by sending transactions from their account. On Starknet, accounts are smart contracts, a model known as native account abstraction. This allows for flexible authorization logic such as multisig, session keys, or passkey-based authentication — all without changes to the protocol itself. The transactions users send are usually invoke transactions, which essentially contain a list of [(contract, selector, parameters)] for execution. Notice that multicall is thus natively supported on Starknet. For more details on invoke transactions as well as rarer types such

Archived copy · cryp2nova.com

as declare, see transactions Some transactions may involve communication between Ethereum and Starknet (or even be initiated by an Ethereum transaction), and are handled through L1↔L2 messaging. An example is deposits to Starknet from Ethereum through secure bridges such as StarkGate.

Block production and validation

Transactions are arriving in the Starknet sequencer's mempool, and from there are collected and ordered into blocks. To ensure that state transitions are valid, Starknet uses SHARP to generate and aggregate proofs that running the SNOS program on the set of the block's transactions indeed yields the Starknet state at the end of the block. These proofs (with further proof aggregation, happening inside SHARP), compress many blocks' execution into a succinct artifact, which is submitted to Ethereum to be verified, so that Starknet's execution can be trusted without re-running it. Starknet also ensures access to the data used in the computation via data availability, publishing compressed state diffs to Ethereum so the full state can be reconstructed and verified.

Tokenomics and consensus

STRK, Starknet's native token, is used to pay fees for transaction execution. Minimal fee level is set to cover the cost of using network resources. STRK is also used to power Starknet's staking protocol, where STRK stakers help secure the sequencing layer and validate block production. This mechanism is designed to support decentralization and provide economic guarantees around block inclusion and ordering.

Summary

All together, these elements form a tightly integrated protocol, enabling scalable and expressive applications with low fees and strong security — all without compromising on decentralization or alignment with Ethereum. Now that you've got the big picture, put on your goggles and deep dive into whichever concept or component intrigues you most!

Was this page helpful?

Yes No

Suggest edits Raise issue

Accounts

Next

Ask a question... ⌘I

CRYP2NOVA

Archived copy · cryp2nova.com